

PROGRAMMIERBARE TI-58/58C/59

Individuelles programmieren



TEXAS INSTRUMENTS



TASTENINDEX

Schneller Überblick, auf welcher Seite die Beschreibung jeder Taste zu finden ist.

A' A	V-55 V-55	B' B	V-55 V-55	C' C	V-55 V-55	D' D	V-55 V-55	E' E	V-55 V-55
2nd	V-3	INV	V-3	log lnx	V-16 V-16	CP CE	V-3, 43 V-3	CLR	V-3
Pgm LRN	III-1 V-43	P-R x<t	V-30 V-30	sin x²	V-17 V-20	cos √x	V-17 V-20	tan 1/x	V-17 V-15
Ins SST	V-52 V-48	CMs STO	V-3, 23 V-23	Exc RCL	V-26 V-23	Prd SUM	V-24 V-24	Ind y^x	V-68 V-21
Del BST	V-51 V-48	Eng EE	V-8 V-5	fix (V-8 V-12	Int)	V-20 V-12	 x ÷	V-20 V-10
Pause GTO	V-44 V-56	x=t 7	V-62 V-2	Mem 8	V-51 V-2	Op 9	V-27 V-2	Deg X	V-16 V-10
Lbl SBR	V-55 V-58	x=t 4	V-62 V-2	Σ+ 5	V-32 V-2	Σ 6	V-33 V-2	Rad -	V-16 V-10
St. Off RST	V-65 V-44	W. Off 1	V-65 V-2	D.MS 2	V-30 V-2	π 3	V-2 V-2	Grad +	V-16 V-10
Write R/S	VII-2 V-43	Dsz 0	V-63 V-2	Adv .	VI-3 V-2	Prt +/-	VI-2 V-2	List ≡	VI-4 V-10

WICHTIG

Tragen Sie hier das Kaufdatum und die Seriennummer ein, die am Bodenteil des Rechners angebracht ist. Die Seriennummer ist durch die Worte "SERIAL NO" gekennzeichnet. Geben Sie diese Informationen bei jeder Korrespondenz an.

PROGRAMMIERBARER
RECHNER TI-

Modell-Nummer

Serien-Nummer

Kaufdatum

INHALTSVERZEICHNIS

Abschnitt

Seite

I. GRUNDINFORMATIONEN ZUM KENNENLERNEN

Einführung	I- 1
Einschalten ON/OFF	I- 1
Operationsarten	I- 2
Benutzung der Software	I- 3
Berechnungen über die Tastatur	I- 4
Erstellen eigener Programme – ein Beispiel	I- 4
Druckmöglichkeiten	I- 5
Rechneraufbau	I- 6

II. DIE TASTATUR – EINE ÜBERSICHT

–Ein Blick auf Eigenschaften und Funktionen	II- 1
Haupteigenschaften der Tastatur	II- 2
Löschen der Eingabe – CE , CLR	II- 2
Dateneingabe-Tasten – 0 bis 9 , . , +/- , π	II- 2
Tasten für die Grundrechenarten – + , - , \times , \div , =	II- 2
Die AOS-Eingabemethode	II- 3
Klammertasten – (,)	II- 4
Doppelfunktions-Tasten – 2nd , INV	II- 5
Speichertasten – CMs , STO , RCL , Exc	II- 6
Speicherarithmetik-Tasten – SUM , Prd	II- 7
Anzeigenkontrolle	II- 8
Standardanzeige	II- 8
Exponentialform-Taste – EE	II- 8
Technische Notation-Taste – Eng	II- 9
Festkommaeinstellung – Fix	II- 9
Algebraische Funktionen	II-10
Quadrat, Quadratwurzel, Reziprokwert-Taste – x^2 , \sqrt{x} , $1/x$	II-10
Potenzen und Wurzeln – y^x	II-10
Logarithmen – ln , log	II-11
Winkelmodus-Tasten – Deg , Rad , Grad	II-12
Tasten für trigonometrische Funktionen – sin , cos , tan	II-12
Umrechnungen	II-13
Altgrad-Umformungen – DMS	II-13
Polar/Rechtwinklige Umrechnungen – P\rightarrowR	II-14
Tasten für statistische Funktionen	II-16
Mittelwert, Varianz und Standardabweichung	II-16
Lineare Regression	II-17

III. ANWENDUNG DER „EINGEBAUTEN“ PROBLEMLÖSUNGEN

ZUGRIFF AUF DIE SOLID-STATE-SOFTWAREPROGRAMME	III- 1
Programmsammlungen	III- 1
Der Software-Modul	III- 1
Ablauf der Softwareprogramme	III- 3
Analyse von Softwareprogrammen (Programmübernahme)	III- 4

INHALTSVERZEICHNIS (Fortsetzung)

Abschnitt

Seite

IV. DIE PROGRAMMIERUNG	IV- 1
Der Begriff Programmierung	IV- 1
Grundelemente der Programmierung	IV- 2
Einbringen einer Variablen in ein Programm	IV- 2
Technik der Programmierung	IV-10
Die Anwendung der Programmadress-Tasten (Labels)	IV-11
Kurzformadressierung	IV-15
Eintasten Ihres Programms	IV-16
Anzeige des Programms	IV-17
Zeitdifferenz-Programm	IV-18
Redigieren von Programmen	IV-21
Verbesserung des Zeitdifferenz-Programms	IV-22
Redigieren mit kombinierten Codes	IV-26
Typische Anwendungsbeispiele für die Programmierung	IV-27
Programmieren ist individuell	IV-27
Zinseszins-Programm	IV-27
Preisberechnungs-Programm	IV-32
Programm zur Umrechnung von Kugelkoordinaten	IV-38
Fortgeschrittenes Programmieren	IV-43
Mehr über Labels	IV-43
Verzweigungsbefehle	IV-43
Unbedingte Verzweigungen	IV-44
Der GO-TO-Befehl	IV-44
Unterprogramme	IV-46
Der Unterprogrammbefehl — SBR	IV-46
Aufruf von Unterprogrammen	IV-48
Wichtige Faktoren bei Unterprogrammen	IV-49
Softwareprogramme als Unterprogramme	IV-52
Biorythmus-Programm	IV-53
Bedingte Verzweigungen (Entscheidungsbefehle)	IV-57
Anzeigeregister-T-Register-Vergleich	IV-57
Quadratwurzel-Beispiel	IV-59
Flag-Operation	IV-61
Sonderfunktionen von Flags	IV-65
Metrisches Umrechnungsprogramm	IV-65
Datenregisterverzweigungen — DSZ	IV-68
Programmierung von Schleifen	IV-68
Unbedingte Schleifenbildung	IV-68
Bedingte Schleifenbildung	IV-70
Schleifenbildung mit dem bedingten Verzweigungsbefehl DSZ	IV-71
XI-Programm	IV-72
Mehr über die Anwendungsgebiete	IV-75
Kurswert von festverzinslichen Wertpapieren (Prg)	IV-75
Programm zur Lösung von quadratischen Gleichungen	IV-79
Weitere Programmiertechniken	IV-84
Programmieren von indirekten Befehlen	IV-84
Indirekter Zugriff auf Datenregister	IV-84
Indirekte Verzweigungsbefehle	IV-86
Andere Eigenschaften	IV-87

INHALTSVERZEICHNIS (Fortsetzung)

Abschnitt

Seite

Programmoptimierung	IV-89
Programmiertechniken für vereinfachte Anwendung	IV-89
Programmiertechniken zur Reduzierung von Schritten	IV-89
Programm zur Berechnung von Bearbeitungsgebühren	IV-93
Programmiertechniken für kürzere Verarbeitungszeiten	IV-98
Kodebrecher (Zahlenratespiel-Programm)	IV-101

V. DIE DETAILS – eine eingehende Analyse der Eigenschaften und Funktionen	V- 1
Grundoperationen	V- 1
Standardanzeige	V- 1
Dateneingabe-Tasten	V- 2
Löschoperationen	V- 3
Doppelfunktionstasten (2nd und INV)	V- 3
Anzeigeformen	V- 5
Exponentialform	V- 5
Technische Notation	V- 8
Festkomma-Kontrolle	V- 8
Blinkende Anzeige	V- 9
Arithmetische Berechnungen	V-10
Die Grundfunktionen – + , - , X , ÷ , =	V-10
Algebraisches Operationssystem	V-11
Klammern	V-12
Blindoperationen mit Klammern	V-15
Algebraische Funktionen	V-15
Reziprokwert	V-15
Logarithmen	V-16
Potenzen von 10 und e	V-16
Winkelberechnungen	V-16
Winkelmodi	V-16
Trigonometrische Funktionen	V-17
Arkusfunktionen	V-18
Altgrad-Radian-Neugrad(Gon)-Umrechnungen	V-19
Ganzzahlfunktion und Absolutwert	V-20
Quadrat und Quadratwurzel	V-20
Potenzen und Wurzeln	V-21
Speichermöglichkeiten	V-22
Wahl der Speicherkapazität (Speicherbereichsverteilung)	V-22
Löschen des Datenspeichers	V-23
Speicherung und Aufruf von Daten	V-23
Direkte Registerarithmetik	V-24
Speicher-/Anzeigeaustausch	V-26
Spezielle Steueroperationen	V-27
Druckerfunktionen – Op 00–08	V-28
Analyse eines Softwareprogramms (Programmübernahme) Op 09	V-28
Signum-Funktion – Op 10	V-28
Statistik – Op 11–15	V-28
Speicherbereichsverteilung – Op 16–17	V-29
Testoperationen – Op 18–19	V-29
Inkrement u. Dekrement von Datenregistern – Op 20–29/30–39	V-29
Drucker-Testoperation Op 40 (nur für TI-58C)	V-29

INHALTSVERZEICHNIS (Fortsetzung)

Abschnitt

Seite

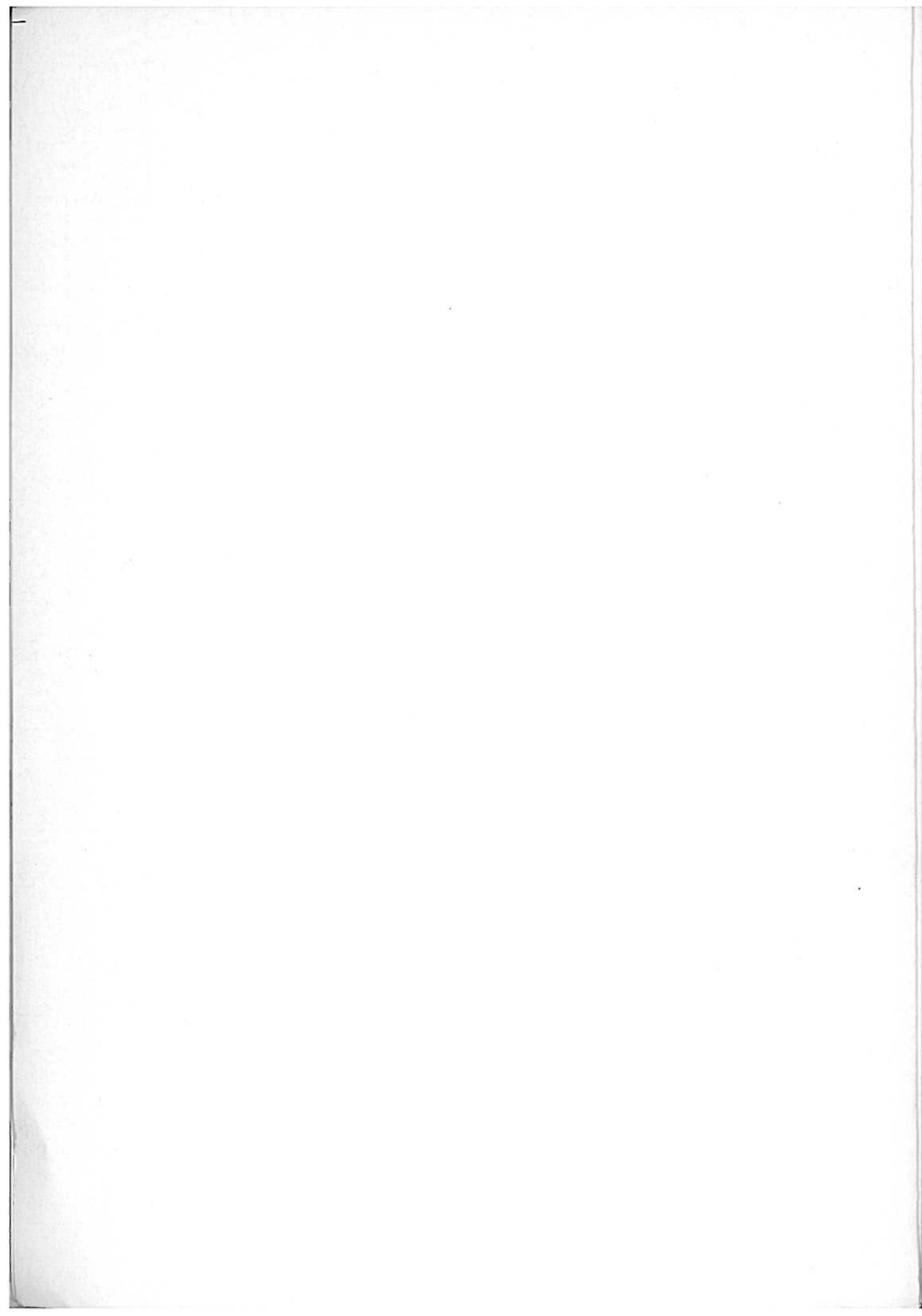
Umrechnungen und Statistik	V-30
Umrechnungen	V-30
Winkelumrechnungen	V-30
Polar-/Rechtwinklige Koordinaten-Umrechnungen	V-30
Statistik	V-32
Dateneingabe	V-32
Mittelwert, Varianz und Standardabweichung	V-33
Lineare Regression	V-36
Trendlinien Analyse	V-39
Statistik in Berechnungen	V-40
Programmieren – allgemein	V-41
Programmierung Ihres Rechners	V-41
Speicherkapazität und Verteilung	V-42
Grundfunktionen für die Programmsteuerung	V-43
Learn-Modus	V-44
Eingabe Ihres Programms	V-45
Ablauf Ihres Programms	V-46
Arbeiten mit Programmen	V-48
Befehlskodes (Tastenkodes)	V-48
Speicherung von Tastenanweisungen	V-51
Redigieren von Programmen	V-51
Ersetzen eines Befehls durch einen anderen	V-52
Löschen eines Befehls	V-52
Einfügen eines Befehls	V-52
Kennzeichnung von Programmteilen	V-55
Programmadressen als Labels	V-55
Allgemeine Labels	V-56
Sprungbefehle	V-56
Unbedingte Sprungbefehle (GTO und SBR)	V-56
GO TO-Befehl	V-56
Unterprogramme	V-58
Softwareprogramme als Unterprogramme	V-60
Bedingte Sprünge (Testbefehle)	V-62
T-Register Vergleiche	V-62
Dekrement und Überspringen bei Null (DSZ)	V-63
Flags	V-65
Flags und Fehlerbedingungen	V-67
Indirekte Adressierung	V-68
VI. DRUCKERSTEUERUNG	VI- 1
Wahlweiser Ausdruck nach Bedarf	VI- 2
Auflisten Ihrer Programme	VI- 4
Auflisten der Datenregister	VI- 4
Protokollieren Ihrer Berechnungen (Parallelbetrieb)	VI- 5
Spezielle Steueroperationen zum Drucken	VI- 7
Alphanumerisches Drucken	VI- 7
Aufzeichnen von Daten (Plotten)	VI-10
Auflisten der im Programm verwendeten Labels	VI-11
Druckkopfreinigung	VI-12

INHALTSVERZEICHNIS (Fortsetzung)

Abschnitt

Seite

VII. MAGNETKARTEN (NUR FÜR DEN PROGRAMMIERBAREN RECHNER TI-59)	VII- 1
Aufzeichnung (Schreiben) auf Magnetkarten	VII- 2
Datenschutz für Programme	VII- 4
Lesen von Magnetkarten	VII- 5
Programmierter Lesebefehl	VII- 5
Pflege der Magnetkarten	VII- 7
Die Behandlung der Magnetkarten	VII- 7
Reinigung der Magnetkarten	VII- 8
Beschriftung der Magnetkarten	VII- 8
Anwendung der Magnetkopf-Reinigungskarte	VII- 8
Anwendung der Reinigungskarte für die Antriebsrolle	VII- 8
Anwendung der Rechner-Diagnosekarte	VII- 9
ANHANG A – WARTUNGS- UND SERVICE-INFORMATIONEN	A- 1
Batterie- und Netzbetrieb	A- 1
Abhilfe bei Störungen	A- 3
Wenn Sie Fragen haben oder Unterstützung brauchen	A- 6
ANHANG B – FEHLERBEDINGUNGEN	B- 1
Fehler während des Programmablaufs	B- 2
ANHANG C – ANGEZEIGTE ERGEBNISSE UND GENAUIGKEIT	C- 1
ANHANG D – STÖRUNGSSUCHE IN PROGRAMMEN	D- 1
Grundsätzliche Überlegungen	D- 1
Programm-Diagnose	D- 4





EINFÜHRUNG

Die heutigen Taschenrechner können die Anwendung der Mathematik bei der Lösung von Problemen in vielen Bereichen leichter machen. Eine neue Arbeitsgeschwindigkeit, Genauigkeit und Sicherheit gehören jetzt zum alltäglichen Umgang mit Zahlen und Mathematik, angefangen bei Routineaufgaben bis hin zu komplexen Problemen.

Ursprünglich war die Rechenkapazität auf die Grundrechenarten - Addition, Subtraktion, Multiplikation und Division - beschränkt, eine bahnbrechende Entwicklung zu dieser Zeit. Der nächste Schritt waren leistungsfähigere Rechner mit mehr mathematischen Funktionen, wie Quadraten, Quadratwurzeln, Logarithmen, trigonometrischen Funktionen etc. Damit erübrigten sich nicht nur Bände von Tabellen und Diagrammen, auch Genauigkeit und Rechengeschwindigkeit nahmen bei der Problemlösung in allen technischen Disziplinen erheblich zu.

Und heute - eine neue Dimension. Die Programmierbarkeit der Taschenrechner eröffnet völlig neue Lösungsbereiche - die bis zu diesem Zeitpunkt dem Computer vorbehalten waren. Dieses Handbuch wurde so konzipiert, daß Sie sofort mit dem Programmieren beginnen können. Sie werden sehen, wie einfach es in Wahrheit ist, Zugang zu der Leistungsfähigkeit Ihres programmierbaren Rechners zu finden.

Das vorliegende Handbuch bezieht sich auf die programmierbaren Rechner TI-58, TI-58C und TI-59. Diese Geräte weisen folgende Unterschiede auf :

Programmierbarer Rechner	Verfügbare Register für		Besondere Merkmale
	Datenspeicherung	Programmspeicherung	
TI-58	bis 60	bis 480	
TI-58C	bis 60	bis 480	Constant Memory™
TI-59	bis 100	bis 960	Magnetkarten

Die Rechner TI-58C und TI-59 haben die Möglichkeit, Datenregisterinhalte und Programminformationen über einen längeren Zeitraum zu speichern. Der TI-59 benutzt dazu Magnetkarten, auf denen Informationen aufgeschrieben und zu einem späteren Zeitpunkt wieder in den Rechner eingelesen werden. Der TI-58C besitzt die Constant Memory™ -Eigenschaft, die vor dem Verlust von Informationen schützt, wenn der Rechner abgeschaltet ist. Diese Eigenschaft erlaubt ferner, das Batteriepaket zu entfernen, um es entweder auszutauschen oder den Rechner an einen Drucker anzuschließen, ohne dabei das Programm oder die Datenregisterinhalte zu verlieren. Weitere Informationen finden Sie in den Abschnitten VI, VII und im Anhang A.

Alle anderen Rechnerfunktionen und Operationen sind identisch. Wenn die Rechneroperationen unterschiedlich sind, finden Sie entsprechende Anmerkungen im Text. Das Buch ist wie folgt gegliedert:

- Die Einleitung enthält einige kurze Erläuterungen der einfachen Anwendung und Programmierbarkeit Ihres Gerätes.
- Es folgt eine Beschreibung der Tastatur und der Funktionen.
- Anschließend finden Sie eine schrittweise Abhandlung über das Programmieren.
- Später werden einige der fortgeschrittenen Programmiereigenschaften aus verschiedenen Bereichen erläutert.
- Der letzte Abschnitt des Handbuches enthält eine detaillierte und vollständige Analyse aller Tasten einschließlich der Kapazitätsgrenzen des Geräts in den verschiedenen Rechensituationen. (Wenn Sie bereits mit Rechnern und ihrer Programmierung vertraut sind, und sofort alle Fakten und Einzelheiten wissen wollen, können Sie die anderen Abschnitte übergehen und die technischen Details Ihres Rechners hier direkt überblicken.)






BETRIEBSHINWEISE

Das mitgelieferte Batteriepaket wurde beim Hersteller vor dem Versand aufgeladen. Vor Inbetriebnahme des Rechners kann jedoch wegen der Selbstentladung, die bei allen Batterien stattfindet, ein erneuter Ladevorgang notwendig werden. Wenn Sie erstmals mit Ihrem Gerät arbeiten, und die Anzeige wird dabei dunkel oder leuchtet unkontrolliert auf, muß das Batteriepaket aufgeladen werden. Schalten Sie den Rechner aus, schließen Sie das Ladegerät an und warten einige Minuten. Dann können Sie fortfahren. Auch während des Ladevorgangs kann mit dem Rechner gearbeitet werden.

Der programmierbare Rechner TI-58C besitzt die Constant Memory™ -Eigenschaft, durch die das Programm und die Inhalte sämtlicher Datenspeicher auch dann erhalten bleiben, wenn der Rechner sofort abgeschaltet wird oder das Batteriepaket für kurze Zeit entfernt wird (siehe dazu Anhang A).

Schieben Sie den Ein/Aus-Schalter (ON/OFF) auf ON. In der Anzeige muß nun eine Null erscheinen, ein Hinweis, das Batteriepaket ist aufgeladen und der Rechner betriebsbereit. Mit dem Einschalten wird der programmierbare TI-59 automatisch komplett gelöscht. Die Constant Memory - Eigenschaft des programmierbaren Rechners TI-58C hält dagegen den Inhalt des Programm- und des Datenspeichers, die Speicherbereichsverteilung und den Festkomma-Status, der beim letzten Ausschalten eingestellt war.

Um die Anzeige des Rechners zu prüfen, drückt man die Tasten für das Dezimalkomma und den Vorzeichenwechsel,  und , und dann wiederholt die Zahl 8, damit alle Segmente in jeder Ziffernposition in der Anzeige aufleuchten. Beachten Sie, daß sich mit jeder 8 Dezimalkomma und Minuszeichen nach links verschieben. Bis zu 10 Ziffern nacheinander können eingegeben werden, unabhängig, ob die Zahlen positiv oder negativ sind. Alle Zifferneingaben nach der zehnten werden ignoriert. Zum leichteren Ablesen bleibt das Minuszeichen in der Anzeige immer unmittelbar links vor der negativen Zahl.

Die Anzeige blinkt, wenn ihre Grenzen überschritten werden, oder wenn eine Operation gefordert wird, die der Rechner nicht durchführen kann. Das Blinken wird abgestellt, wenn man die Eingabelösch taste  drückt.

Sie erhalten eine ausführliche Anleitung zu diesem Gerät, aber bedenken Sie, daß dies kein Ersatz für eigenes Prüfen und das Sammeln von Erfahrungen sein kann. Ihre Eigeninitiative ist eine der besten Möglichkeiten, die Vielseitigkeit und Leistung des Rechners kennzulernen. Je mehr Sie über die weitreichenden Fähigkeiten erfahren, desto besser werden Sie den Rechner für Ihren Bedarf einsetzen können.

OPERATIONSARTEN

Grundsätzlich kann der Rechner in drei verschiedenen Operationsarten betrieben werden:

Man kann problemlos eines der vielen „Solid State Software“ - Programme verwenden, die direkt in das Gerät eingesetzt werden, und damit komplexe Probleme mit nur einigen Tastenbetätigungen bearbeiten - ODER -

Man kann dem Rechner eigene Anweisungen zur Problemlösung erteilen, und er behält diese Lösungsmethode und kann sie ausführen, wann immer Sie es wünschen - ODER -

Ihr Gerät steht immer als leistungsfähiger Rechner zur Verfügung, der manuell über die Tastatur bedient wird - bereit, jederzeit allgemeine mathematische Probleme zu lösen oder komplizierte Berechnungen durchzuführen.



Benutzung der Software

Auch ohne eigene Kenntnisse in der Rechnerprogrammierung können Sie viele nützliche Programme ablaufen lassen. Ein Modul, der bereits in der Rückseite des Rechners eingesetzt ist, enthält eine Grundsammlung gespeicherter „Solid State Software“-Programme. Dieser auswechselbare Modul (andere Module sind lieferbar) enthält eine Vielzahl von Standardprogrammen, die in dem Handbuch für die Standard-Software-Programme beschrieben sind. Mit der Programmtaste **Pgm** kann jedes Programm aufgerufen und entsprechend der Beschreibung durchgeführt werden. Mit dem „Hi-Lo“-Spiel soll gezeigt werden, wie einfach die Anwendung dieser Programme ist.

Ihre Aufgabe in dem Spiel ist es, eine Geheimzahl mit möglichst wenig Versuchen zu erraten. Der Rechner wählt eine Zahl im Bereich 1 bis 1023, und antwortet auf jeden Versuch mit „zu groß“, „zu klein“ oder „richtig“. Ihre Punktzahl (Anzahl der Versuche) wird vom Rechner registriert. Beginnen Sie zu spielen und beachten Sie dabei die Programm-Instruktionen.

PROGRAMM-INSTRUKTIONEN				
Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Auswahl des Programms		2nd Pgm 21	
2	Eingabe eines Dezimalkommata und einer Reihe beliebiger Ziffern	Ihre Zahl	A	Ihre Zahl
3	Erzeugen der Geheimzahl		B	0.
4	Eingabe Ihres Versuchs (1 bis 1023) Anhaltspunkt: -1 : zu niedrig 1 : zu hoch blinkende 0 : richtig	Versuch	C	Anhaltspunkt
5	Wiederholung von Schritt 4 so oft wie nötig			
6	Anzeige der Punktzahl (Anzahl der Versuche)		D	Punktzahl

Die meisten gespeicherten Programme können ebenfalls so problemlos verwendet werden. Tatsächlich wurden Dutzende von Programmschritten durchgeführt, aber automatisch. Sie brauchen nur die Zahlen einzugeben, mit denen Sie arbeiten wollen, und das Programm zu starten.

Ein wichtiger Punkt – der Schlüssel zu allen „Solid State Software“-Programmen ist das Handbuch zur jeweiligen Programmsammlung. Alle Eingaben, Ausgangsinformationen und die wesentlichen Details, die Sie für die optimale Nutzung des Software-Programms benötigen, sind in diesem Handbuch enthalten. Achten Sie also auf diese Anleitungen, wann immer Sie eines der gespeicherten Programme verwenden. Es empfiehlt sich, das Handbuch zu den Standard-Software-Programmen sofort durchzusehen. Sie werden dann mit den Programmen vertraut, die Ihnen bereits zur Verfügung stehen – und zur Bearbeitung Ihrer Probleme sofort beitragen können.



Berechnungen über die Tastatur

Der zeitgemäße Rechner von Texas Instruments arbeitet bei der Eingabe der Probleme mit dem AOSTTM-System, eine der natürlichsten Eingabemethoden, die bisher entwickelt wurden. Die Aufgaben werden ohne Schwierigkeiten durch direkte Eingabe in der Reihenfolge wie beim schriftlichen Ansatz von links nach rechts gelöst. Um zum Beispiel 100°C , 36°C und -4°C in Fahrenheit umzurechnen, müssen Sie den Celsius-Wert mit $\frac{9}{5}$ multiplizieren und 32 addieren.

$$^{\circ}\text{F} = ^{\circ}\text{C} \times \frac{9}{5} + 32$$

Taste	Anzeige
100 \times	100.
9 \div	900.
5 $+$	180.
32 $=$	212.

Diese Folge kann wiederholt werden, um die Ergebnisse $36^{\circ}\text{C} = 96.8^{\circ}\text{F}$ und $-4^{\circ}\text{C} = 24.8^{\circ}\text{F}$ zu erhalten. (Ausführliche Informationen über die AOSTM-Eingabemethode und die daraus resultierende Rechenleistung finden Sie in einem späteren Abschnitt dieses Buches.)

Erstellen eigener Programme — ein Beispiel

Sobald eine Rechenfolge festliegt, und Sie mehrere Werte haben, die mit dieser Folge verarbeitet werden, können Sie die Taste **[LRN]** (learn) drücken und diese Folge in den Rechner eingeben. Für das obige Beispiel drückt man **[LRN]** und dann die nachstehenden Tasten:

\times
9
 \div
5
 $+$
3 2
 $=$
[R/S] (Programmstop und Ergebnisanzeige)

Nach dieser Folge drücken Sie **[LRN]** erneut - der Rechner wird damit angewiesen, das „Erlernen“ der einzelnen Eingaben zu beenden. Er „erinnert“ sich jetzt an diese Folge und ist bereit, diese Reihe von Operationen für jede eingegebene Zahl durchzuführen, (in diesem Fall für jeden Celsius-Wert).

ANMERKUNG: TM = Schutzmarke



Sie können jetzt also veranlassen, daß der Rechner die Celsius-Fahrenheit-Umrechnungen für jede eingegebene Zahl löst.

- Zuerst wird der Celsius-Wert eingetastet.
- Dann wird der Rechner mit der Taste **RST** (reset) angewiesen, am Programmfang zu starten.
- Schließlich drückt man **R/S** (run/stop), um mit der Durchführung des Programms zu beginnen.

Taste	Anzeige
100 RST R/S	212.
36 RST R/S	96.8
4 +/- RST R/S	24.8

Dies ist alle Arbeit, die erforderlich ist, um jeden Celsius-Wert in seinen entsprechenden Fahrenheit-Wert umzuwandeln. So einfach kann also das Erstellen eines eigenen Programms sein.

Diese Möglichkeit, ein eigenes Programm auszuführen, stellt einen wesentlichen Teil der Leistungsfähigkeit des Rechners dar. Sobald ein Programm gespeichert und auf Richtigkeit und Genauigkeit überprüft ist, können Sie es immer wieder verwenden, einfach mit einem Tastendruck.

DRUCKMÖGLICHKEITEN

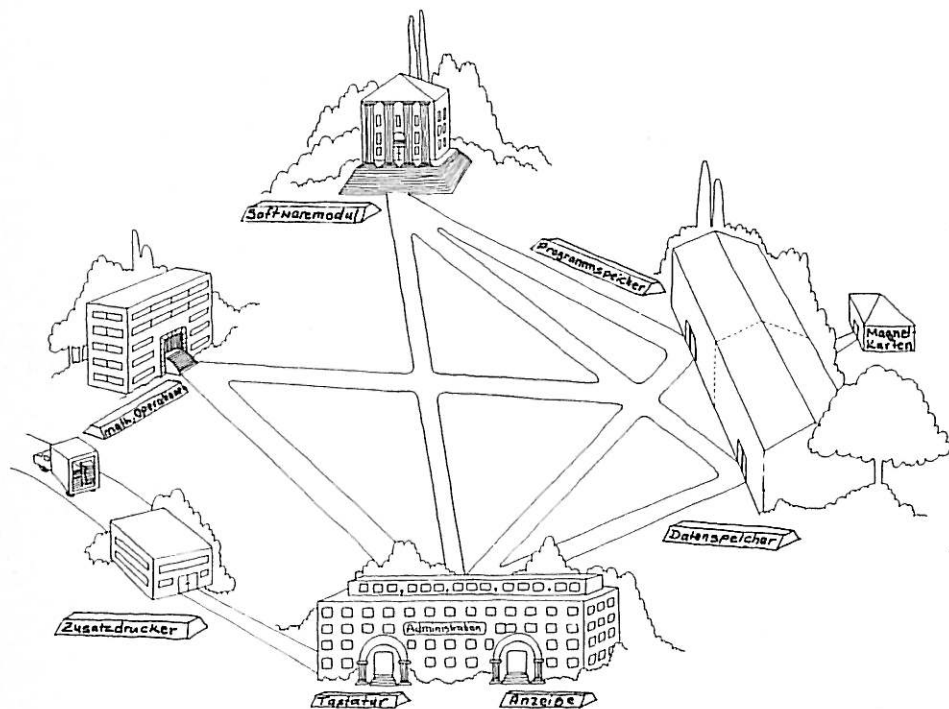
Ihr Rechner ist mit den Druckern der Typen PC-100A, B oder C kompatibel. Der Drucker kann den Anzeigewert auf Papier aufzeichnen, wenn Sie die entsprechende Anweisung geben. Wenn die Probleme direkt über die Tastatur berechnet werden, kann man wahlweise einige oder alle gewünschten Zwischenergebnisse ausdrucken lassen oder auch eine vollständige Auflistung eines gespeicherten Programms anfertigen. Wenn ein Programm abläuft, bewirken die Druckbefehle im Programm, daß die Größe im Anzeigeregister automatisch ausgedruckt wird. Diese Druckeigenschaften erlauben einen Programmablauf mit gleichzeitiger Aufzeichnung einer Reihe von Ergebnissen. Mit dem TRACE-Modus (Parallelbetrieb) auf dem Drucker werden alle durchgeführten Schritte und die entsprechenden numerischen Ergebnisse ausgedruckt.

Durch Anwendung der speziellen Steueroperationen können Sie Nachrichten zusammenstellen und ausdrucken lassen, um Programmsegmente in der Auflistung oder Programmtitel zu kennzeichnen. Pro Zeile können 20 Schriftzeichen gedruckt werden, die Sie aus einer Auswahl von insgesamt 64 Schriftzeichen zusammenstellen.



RECHNERAUFBAU

Ihr neuer Rechner verfügt über viele leistungsfähige Baugruppen, die in ihrem Zusammenwirken ein Problemlösungssystem bilden, das auf einfache und problemlose Anwendung abgestellt ist. In diesem Abschnitt erhielten Sie einen kurzen Abriss von einigen dieser Eigenschaften in Aktion. Der nächste Abschnitt enthält einen Überblick über diese Funktionseinheit mit einigen weiteren Einzelheiten, und dann können Sie sich der Arbeit und der Freude des Programmierens zuwenden.





Ehe Sie sich mit den fortgeschritteneren Eigenschaften Ihres Gerätes beschäftigen, ist eine kurze Übersicht über die wesentlichen Kennzeichen und Funktionen der Tastatur zweckmäßig. Dies trifft vor allem für den Fall zu, daß Sie Ihre ersten Erfahrungen mit einem hochentwickelten Rechner machen. Viele Eigentümer von Rechnern konnten die tatsächliche Leistung ihrer Geräte nie voll nutzen - einfach weil sie sich nie die Zeit nahmen, die Wirkung jeder einzelnen Taste zu beobachten. Dieser Abschnitt enthält einen kurzen Bericht über die Tastatur - und Ihre Zeit wird nicht länger als 10 - 15 Minuten beansprucht. Sie lernen damit weitgehend die Hauptmerkmale der Tastatur kennen - und wenn Sie dann mit dem Programmieren beginnen, können Sie alle Vorteile aus der Leistungsfähigkeit des Rechners ziehen.

Eine Anmerkung für die verschiedenen Anwender:

Wenn Sie die fortschrittlichen Rechner mit der AQS-Eingabemethode bereits kennen, werden Sie diese Beschreibung vielleicht übergehen wollen, um sich im Abschnitt IV sofort mit der Programmierung zu beschäftigen.

Eine gesonderte und ausführliche Beschreibung aller Operationen und Fähigkeiten des Rechners finden Sie im Abschnitt V, wo jede Taste und jede Eigenschaft genau erklärt wird.

Wenn Sie nun mit dieser Übersicht beginnen, halten Sie auch den Rechner griffbereit. Testen Sie selbst jede Taste und jedes Merkmal anhand der Beschreibung. Der beste Weg, Ihren Rechner kennenzulernen, ist, ihn zu benutzen.



HAUPTEIGENSCHAFTEN DER TASTATUR

Löschen der Anzeige — **CE**, **CLR**

Diese beiden Vorgänge ermöglichen das Löschen des Anzeigerregisters, abhängig von Ihren Erfordernissen bei der Bearbeitung eines Problems.

CE LÖSCHEN DER EINGABE (clear entry) — die Eingabelöschstaste löscht die letzte Zahl, die in die Anzeige eingegeben wurde (vorausgesetzt, daß keine Funktions- oder Operationstaste gedrückt wurde). Diese Taste hat keinen Einfluß auf laufende Berechnungen. (Wenn Sie also irrtümlich 5 statt 6 mitten in einer Zahleneingabe drücken, korrigieren Sie mit der Taste **CE** und tasten dann die vollständige, richtige Zahl ein.) Mit der **CE**-Taste kann auch das Blinken der Anzeige, das durch eine Fehlerbedingung verursacht wurde, wieder abgestellt werden.

CLR LÖSCHEN (clear) — mit der Löschstaste werden der Inhalt des Anzeigerregisters sowie alle laufenden Berechnungen gelöscht. Wenn beim Drücken dieser Tasten gleichzeitig eine Fehlerbedingung existiert, wird auch diese gelöscht.

DATENEINGABETASTEN — **0** bis **9**, **.**, **+/-**, **π**

Die Eingabe der Zahlen erfolgt über die Dateneingabe-Tasten **0** bis **9**, **.** und **+/-**. Wenn Sie eine Zahl eingeben, bleibt das Dezimalkomma immer rechts von Ihrer Eingabe, bis die Dezimalkomma-Taste selbst gedrückt wird. Der Bruchteil der Zahl wird dann eingetastet und das Komma verschiebt sich entsprechend mit ihm nach links. Um das Erscheinen einer Zahl in der Anzeige zu ändern, drückt man die Vorzeichenwechsellaste **+/-** einmal. (Drückt man **+/-** ein zweites Mal, wird das Vorzeichen wiederum gewechselt).

Mit der Folge **2nd** **π** erscheinen die ersten 10 Stellen der Zahl π in der Anzeige (3,141592654). Im internen Anzeigerregister ist die Zahl mit 13 Stellen gespeichert (3,141592653590). Diese Eingabe kann nicht mit **CE** gelöscht werden.

TASTEN FÜR DIE GRUNDRECHENARTEN — **+**, **-**, **\times** , **\div** , **=**

Die arithmetischen Aufgaben werden mit den 5 Operationstasten **+**, **-**, **\times** , **\div** und **=** berechnet. Der Rechner arbeitet mit der leistungsfähigen AOS-Eingabemethode, die Lösungen mit diesen Tasten wesentlich vereinfacht. Grundsätzlich tastet man die Aufgabe wie einen schriftlichen Ansatz ein, drückt **=** und erhält das Ergebnis. Das Erstaunliche an der AOS-Eingabe ist, daß die gemischten Operationen damit automatisch für Sie sortiert werden, und die korrekte Verarbeitungsfolge bei der Berechnung des Ergebnisses eingehalten wird. (Mehr über die AOS-Eingabemethode finden Sie auf der nächsten Seite.)

Mit der **=**-Taste werden alle unvollständigen Operationen (die Vorgänge, die intern noch ablaufen müssen), abgeschlossen. Sie erhalten das Ergebnis, der Rechner wird gelöscht — und eine neue Aufgabe kann begonnen werden.

Beispiel: $15 + 7 \times 31 - 4 = ?$

Tasten: **15** **+** **7** **\times** **31** **-** **4** **=** Anzeige: 228

ANMERKUNG: Beachten Sie, daß mit der AOS-Eingabemethode der Rechner angewiesen wird, den Ausdruck als $15 + (7 \times 31) - 4$ zu interpretieren, wobei zuerst 7×31 berechnet, dann 15 addiert und davon schließlich 4 subtrahiert wird.



DIE AOS-EINGABEMETHODE

Mathematik ist eine Wissenschaft, die auf einer klar definierten Serie von Gesetzen aufbaut. Eines dieser Gesetze besagt, daß es für ein- und dieselbe Folge von Operationen niemals zwei verschiedene Antworten geben kann. Wegen dieser Forderung – nur eine Lösung für eine Rechnung – wurde eine Reihe von allgemeingültigen Regeln für die Fälle aufgestellt, wo in einer Rechenaufgabe gemischte Operationen verwendet werden. Zum Beispiel hat die Aufgabe

$$3 + 10 - 2 \times 14 \div 7 = ?$$

nur eine richtige Lösung! (Das Ergebnis ist 9)

Sie können das Problem direkt von links nach rechts eintasten und erhalten die korrekte Antwort. Die algebraische Hierarchie des Rechners sortiert die eingegebenen Operationen, wendet sie in der richtigen Reihenfolge an und gestattet Ihnen die ständige Kontrolle über den Verarbeitungsstand. Der Rechner führt die Operationen, die Sie angewiesen haben, in folgender allgemeingültigen Reihenfolge aus:

- 1) Sonderfunktionen mit einer Variablen – sie wirken unmittelbar auf die angezeigte Zahl, sobald Sie die entsprechende Taste drücken. (Dazu gehören alle Tasten für die trigonometrischen und logarithmischen Funktionen, deren Umkehrfunktionen, sowie Quadrate und Quadratwurzeln, Reziprokwerte und die Umrechnungen – mehr über diese Tasten folgt in einem späteren Teil dieses Abschnitts.)
- 2) Potenzen und Wurzeln (y^x und $\sqrt[x]{y}$) werden anschließend berechnet (und ebenfalls später in diesem Abschnitt näher erklärt).
- 3) Dann werden Multiplikationen und Divisionen abgeschlossen, gefolgt von
- 4) Additionen und Subtraktionen.

Diese algebraische Hierarchie gilt für jedes Klammerpaar.

Schließlich werden mit der [=]-Taste alle Operationen angeschlossen.

Es gibt bei der Lösung von Problemen natürlich Fälle, wo Sie selbst die Verarbeitungsreihenfolge für einen Ausdruck bestimmen wollen. In diesen Fällen können Sie die Reihenfolge mit Hilfe der Klammertasten { }, (), [], [] kontrollieren (siehe nächste Seite). Klammern erfordern besondere Aufmerksamkeit in der Mathematik – und dementsprechend wurden sie auch beim Rechner berücksichtigt.



KLAMMERTASTEN – (,)

In einer Vielzahl von Problemen wollen Sie selbst die exakte Reihenfolge festlegen, in der ein Ausdruck berechnet werden soll, oder Sie müssen die Gruppierung von Zahlen bestimmen, wenn ein Problem gelöst wird. Klammern geben Ihnen die Möglichkeit, Zahlen und Operationen zu Gruppen zusammenzufassen. Wenn eine Reihe von Zahlen und Operationen in Klammern gesetzt wird, weisen Sie damit den Rechner an, diese Teilaufgabe zuerst zu lösen – auf einen einzigen Zahlenwert zu reduzieren – und dieses Ergebnis dann im Rest der Berechnung einzusetzen. Innerhalb jedes Klammerpaares arbeitet der Rechner entsprechend den Regeln der algebraischen Hierarchie. Klammern empfehlen sich immer auch dann, wenn Sie Zweifel haben, wie der Rechner einen Ausdruck verarbeiten wird. Der Rechner ist so ausgelegt, daß maximal 9 Klammern mit maximal 8 unvollständigen Operationen gleichzeitig geöffnet sein können.

$$(((2 \times (2 \times (2 \times (2 \times (2 + 2y \cdot (2 + .2)) - (2 + 2)))))) \div 2) \div 2)$$

Beim Eintasten dieser Folge ist zu beachten, daß keine Berechnungen stattfinden, bis nicht die erste rechte Klammer eingegeben wird. Der Rechner hat alle Anweisungen behalten und führt sie zur rechten Zeit aus.

Beachten Sie noch einen wichtigen Punkt, wenn Sie Klammern verwenden. Sie sehen sicher oft Gleichungen oder Ausdrücke, die mit Klammern geschrieben sind, um auf eine Multiplikation hinzuweisen: $(2 + 1)(3 + 2) = 15$. Der Rechner führt keine so angedeuteten Multiplikationen durch. Sie müssen in jedem Fall auch die Operation zwischen den Klammern eingeben:

$$(2 + 1) \times (3 + 2) = 15.$$

Hier ein Beispiel für die Anwendung von Klammern

Berechnen Sie: $\frac{8 \times (4 + 9) + 1}{(3 + 6 \div 2) \times 7}$



In Aufgaben dieser Art soll der Rechner zuerst den gesamten Zähler ermitteln und dann die Division durch den gesamten Nenner durchführen. Um sicher zu sein, daß die Berechnung auf diese Weise erfolgt, klammert man zusätzlich noch beim Eintasten der Aufgabe Zähler und Nenner ein. (Die Schreibweise mit Bruchstrich ersetzt diese beiden Klammerpaare).

Taste	Anzeige	Bemerkungen
CLR	0	Löschen von möglichen laufenden Berechnungen
(8 X (4 + 9)	13.	$(4 + 9)$ wird berechnet
+	104.	$8 \times (4 + 9)$ wird berechnet
1)	105.	Zahlenwert des Zählers
÷ ((3 + 6 ÷ 2)	6.	$(3 + 6 \div 2)$ wird berechnet
X 7)	42.	Zahlenwert des Nenners
=	2.5	Ergebnis



Doppelfunktions-Tasten -- **2nd**, **INV**

Ihr Rechner ist mit zahlreichen Funktionen ausgestattet, die Ihnen Zeit sparen und die Genauigkeit Ihrer Berechnungen erhöhen. Um Zugang zu der gesamten Leistung zu haben, ohne das Gerät mit Tasten zu überladen, kann mit den meisten Tasten mehr als eine Funktion ausgeführt werden. Die Erstfunktion ist direkt auf der Taste selbst angegeben. Um die Erstfunktion einer Taste zu verwenden, drücken Sie einfach die entsprechende Taste. Die Zweitfunktion (deren Symbol unmittelbar oberhalb einer Taste aufgedruckt ist) wird ausgeführt, wenn man die Taste **2nd** und dann sofort die Taste unter der gewünschten Funktion drückt.

Um zum Beispiel den natürlichen Logarithmus einer Zahl zu ermitteln, drücken Sie **ln**. Mit den Tasten **2nd** **ln** berechnen Sie dagegen den dekadischen Logarithmus einer Zahl. Um die Zweitfunktion kenntlich zu machen, wurde für dieses Handbuch die Darstellung **2nd** **log** gewählt. Erstfunktionen werden also durch das schwarze Symbol auf weißem Grund (), und Zweitfunktionen durch **2nd** und das weiße Symbol auf schwarzem Grund () gekennzeichnet.

Auch mit der Umkehrfunktions-Taste **INV** (inverse) erhält der Rechner zusätzliche Funktionen, ohne daß die Anzahl der Tasten erweitert wird. Stellt man die **INV**-Taste einer anderen Funktion oder Taste voran, wird der Sinn dieser Funktion oder Taste umgekehrt. Die **INV**-Taste wird nur zusammen mit einigen Tasten Ihres Rechners verwendet, um zusätzliche Funktionen zu erhalten oder um eine Operation umzukehren.

Mit den Tasten **2nd** und **INV** können 108 verschiedene Operationen mit nur 45 Tasten durchgeführt werden. Weitere Informationen über die Anwendung mit speziellen Tasten siehe Abschnitt V, Doppelfunktionstasten.



Speichertasten - **CME**, **STO**, **RCL**, **EXC**

Jedesmal, wenn Sie Ihren programmierbaren TI-59 einschalten, stehen Ihnen 60 Datenregister zur Verfügung (30 beim TI-58). Die Anzahl der bei Ihrem programmierbaren TI-58C verfügbaren Datenregister hängt von der zuletzt gewählten Einstellung ab, selbst wenn der Rechner zwischenzeitlich ausgeschaltet wurde. Tatsächlich kann die Anzahl der Datenregister gegenüber der Kapazität des Programmspeichers variieren. (Siehe Wahl der Speichergröße in SPEICHERMÖGLICHKEITEN von Abschnitt V).

Die Datenregister sind spezielle Speicherplätze im Rechner, wo Sie die Werte speichern können, die Sie vielleicht später verwenden müssen. Der programmierbare TI-58C hält die Zahlen auch dann in den Registern gespeichert, wenn der Rechner ausgeschaltet wird.

Da im allgemeinen mehr als ein Datenregister verfügbar ist, muß das zu verwendende Register durch seine zweistellige Adresse XX noch näher bezeichnet werden. Zum Beispiel **STO** 08.

Die Tasten **CE** und **CLR** wirken nicht auf die Speicherinhalte; mit der Folge **2nd** **CME** werden jedoch alle Datenregister gleichzeitig gelöscht (d.h., die Register werden jeweils mit einer 0 belegt).

STO XX - SPEICHERN (store) - Mit diesem Befehl wird die Zahl im Anzeigeregister im Datenregister XX (00 - 99) abgespeichert, wobei der Inhalt des Anzeigeregisters nicht beeinflußt wird. (Jede früher im Register XX gespeicherte Zahl wird vorher gelöscht.)

RCL XX - AUFRUF (recall) - Dieser Befehl bringt den Inhalt des Datenregisters XX wieder in die Anzeige. Auch in diesem Fall bleibt der Inhalt des Datenregisters XX erhalten.

2nd **EXC** XX - SPEICHERAUSTAUSCH (memory exchange) - Mit dieser Folge wird der Inhalt des Datenregisters XX mit dem Inhalt des Anzeigeregisters vertauscht. (Der Wert des Anzeigeregisters wird im Register XX gespeichert, während die gespeicherte Zahl in das Anzeigeregister aufgerufen wird.) Diese Taste erweist sich in vielen Situationen als nützlich, weil mit ihr der Speicherinhalt kurz geprüft oder benutzt werden kann, ohne den Wert im Anzeigeregister zu verlieren.

Beispiel: Speicherung und Aufruf der Zahl 3.21

Taste	Anzeige	Bemerkungen
3.21 STO 08	3.21	3.21 wird im Register 08 gespeichert
CLR	0	Löschen der Anzeige
RCL 08	3.21	Der Inhalt von Register 08 wird aufgerufen



Beispiel: Berechnen Sie: $(A + 2) + A(A + 2)$ für $A = 9.3069128$

Taste	Anzeige	Bemerkungen
CLR	0	Löschen möglicher laufender Berechnungen
9.3069128 STO 12	9.3069128	A wird im Register 12 gespeichert
+ 2 +	11.3069128	Berechnung von $A + 2$
2nd EXC 12	9.3069128	$A + 2$ wird im Register 12 gespeichert und A in das Anzeigeregister aufgerufen
X RCL 12	11.3069128	Aufruf von $A + 2$ in das Anzeigeregister. (Beachten Sie daß zwischen A und $A + 2$ das X eingefügt sein muß)
=	116.5393643	Alle unvollständigen Operationen werden abgeschlossen; das Ergebnis wird angezeigt.

Der vielziffrige Wert von A mußte nur einmal eingegeben werden; auf diese Weise sparte man Zeit und vermied mögliche Fehler beim Eintasten. Die Austausch taste vereinigt in sich Speicherung und Aufruf und verkürzt damit den Rechen vorgang.

Speicherarithmetik-Tasten -- **SUM**, **Prd**

Eine Reihe von Tastenfolgen erlaubt das Rechnen mit gespeicherten Zahlen ohne Einfluß auf andere laufende Berechnungen.

SUM XX - SPEICHERADDITION (memory sum) — Mit dieser Folge wird der Inhalt des Anzeigeregisters direkt zu dem im Register XX gespeicherten Wert addiert. Das Ergebnis der Addition wird in dem benannten Register ohne Einfluß auf das Anzeigeregister gespeichert. Entsprechend wirkt die Folge **INV** **SUM** XX, wo der Wert im Anzeigeregister vom Inhalt des Registers XX subtrahiert wird.

2nd **Prd** XX - SPEICHERMULTIPLIKATION (memory product) — Mit dieser Folge wird der Inhalt des Registers XX mit dem Wert im Anzeigeregister multipliziert. Mit **INV** **2nd** **Prd** XX dividiert man den Wert im Register XX durch die Zahl im Anzeigeregister. Das Ergebnis wird wiederum gespeichert, und das Anzeigeregister bleibt unbeeinflusst.

Die Durchführung dieser Befehle ist vergleichbar mit den arithmetischen Funktionen der Tastatur, mit dem Unterschied, daß die Ergebnisse nicht ins Anzeigeregister, sondern in ein Datenregister übertragen werden.

Beispiel: Ermitteln Sie die Gesamtkosten für die Einzelpositionen 28 und 6.60 mit 5% Steuer.

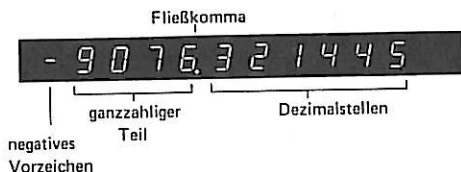
Taste	Anzeige	Bemerkungen
28 STO 01	28.	28 wird im Datenregister 1 gespeichert.
6.6 SUM 01	6.6.	6.6 wird zum Datenregister 1 addiert.
1.05 2nd Prd 01	1.05	Der Inhalt des Datenregisters 1 wird mit 1.05 multipliziert.
RCL 01	36.33	Gesamtkosten



ANZEIGEKONTROLLE

Standardanzeige

Die Anzeige liefert vollständige numerische Informationen einschließlich Minuszeichen und Dezimalkomma und blinkt bei Kapazitätsüberlauf und -unterlauf oder bei Fehlerbedingungen. (Anhang B enthält eine detaillierte Auflistung der Fehlerbedingungen.) Eine Eingabe kann maximal 10-stellig sein. Alle Zifferneingaben nach der 10. Stelle werden ignoriert.



Die Begriffe Anzeige und Anzeigeregister sind keine Synonyme. Das Wort Anzeige bezieht sich auf den bloßen optischen Anzeigebereich. Das Anzeigeregister ist das interne Register, das mit Ergebnissen bis zu 13 Stellen belegt ist.

Ist eine Zahl für die Anzeige im Standardformat zu groß oder zu klein, weist der Rechner diese Zahl automatisch in der Exponentialform aus. Wenn zum Beispiel 400 000 mit 2 000 000 multipliziert wird, ist das Ergebnis 800 000 000 000, eine zu große Zahl für die 10-stellige Anzeige. Sie wird daher als 8. 11 angezeigt ($8.11 \hat{=} 8 \times 10^{11}$).



Exponentialform - Taste **EE**

In vielen Bereichen, insbesondere in Wissenschaft und Technik, müssen Sie oft mit sehr großen oder sehr kleinen Zahlen rechnen. Solche Zahlen können mit Hilfe der Exponentialform ohne Schwierigkeiten verarbeitet werden. Eine Zahl in Exponentialform ist als das Produkt aus einer Grundzahl (Mantisse) und einer Zehnerpotenz (Exponent) ausgedrückt.

$$\text{Zahl} = \text{Mantisse} \times 10^{\text{Exponent}}$$

Eingabe einer Zahl in Exponentialform:

Eingabe der Mantisse mit bis zu 8 Stellen - (dann drückt man **+/-**, wenn die Mantisse negativ ist);

Taste **EE** (Eingabe des Exponenten) - rechts in der Anzeige erscheint „00“;

Eingabe der Zehnerpotenz (und Taste **+/-** bei negativer Zehnerpotenz);

Eine Zahl wie $-3.8901448 \times 10^{-32}$ erscheint wie unten abgebildet in der Anzeige.



In der Exponentialform läßt sich durch die Zehnerpotenz feststellen, wo das Komma stehen müßte, wenn die Zahl ausgeschrieben wird.



Ein positiver Exponent weist darauf hin, um wieviele Stellen das Komma nach rechts verschoben werden müsste, ein negativer Exponent gibt die Verschiebung des Kommas nach links an.

Beispiel: $2.9979 \times 10^{11} = 299\,790\,000\,000$
(das Komma wird um 11 Stellen nach rechts gerückt und soviel Nullen wie nötig werden angefügt.)

$1.6021 \times 10^{-9} = 0.0000000016021$
(das Komma wird um 9 Stellen nach links gerückt und soviel Nullen wie nötig werden eingesetzt.)

Wenn Sie einmal die Exponentialform gewählt haben, wird sie so lange beibehalten, bis sie mit Absicht wieder aufgehoben wird. Drückt man **INV** **EE**, geht der Rechner wieder zur Standardanzeige über, wenn der Wert in der Anzeige im anzeigbaren Bereich liegt. **CLR** hebt diese Form auf, wenn die Anzeige gelöscht wird.

Technische Notation · Taste **Eng**

Die technische Notation ist eine Abwandlung der Exponentialform. Die Potenz (Exponent) ist dabei immer als ein Vielfaches von drei eingestellt (10^{12} , 10^{-6} , etc.). Also kann die Mantisse eine, zwei oder drei Stellen links vom Komma haben. Mit dieser Eigenschaft ist es möglich, daß der Rechner die Ergebnisse in Maßeinheiten anzeigen kann, die der Wissenschaftler, Ingenieur oder Techniker gerne als Arbeitserleichterung verwendet, (wie zum Beispiel 10^{-12} für Picofarad, 10^{-3} für Millimeter, 10^3 für Kilogramm oder 10^{-6} für Mikrosekunden).

Die Anzeige kann jederzeit mit den Tasten **2nd** **Eng** auf technische Notation geschaltet werden. Diese Anzeigeform wird mit der Folge **INV** **2nd** **Eng** wieder aufgehoben.

Beispiel: Berechnung von $8 \times 98 \times 30$ mit technischer Notation.

Taste	Anzeige
CLR 2nd Eng	0. 00
8 X 98 X	784. 00
30 =	23.52 03
INV 2nd Eng	23520.

Festkomma-Einstellung — **Fix**

Diese vorteilhafte Eigenschaft ermöglicht die Wahl der Stellenzahl, die während Ihrer Berechnungen rechts vom Komma in der Anzeige ausgewiesen werden soll. Drücken Sie **2nd** **Fix** und dann die gewünschte Anzahl der Dezimalstellen (0 bis 8). Der Rechner rundet daraufhin alle folgenden Ergebnisse auf diese Dezimalstellenzahl, aber nur für die Anzeige. Eingaben können jedoch weiterhin mit soviel Stellen erfolgen, wie es gerade erwünscht ist, da der Rechner seine eigene interne (13-stellige) Genauigkeit beibehält. Die Festkomma-Einstellung wird mit den Tasten **INV** **2nd** **Fix** aufgehoben. Der programmierbare TI-58C hält die Festkomma-Einstellung auch bei abgeschaltetem Rechner gespeichert.

Taste	Anzeige
CLR	0.
2 ÷ 3 =	.666666667
2nd Fix 6	0.666667
2nd Fix 2	0.67
2nd Fix 0	1.
INV 2nd Fix	.666666667



ALGEBRAISCHE FUNKTIONEN

Tasten für Quadrat, Quadratwurzel und Reziprokwert — x^2 , \sqrt{x} , $1/x$

Diese drei Tasten, gut erreichbar angeordnet, sind sehr wichtig für die schnelle Lösung verschiedener Gleichungssysteme. Alle drei Tasten wirken unmittelbar auf die Zahl im Anzeigeregister, ohne andere laufende Berechnungen zu beeinflussen.

x^2 — QUADRAT — Berechnung der Quadratwurzel der Zahl x im Anzeigeregister.

\sqrt{x} — QUADRATWURZEL — Berechnung der Quadratwurzel der Zahl x im Anzeigeregister.

$1/x$ — REZIPROKWERT — Division von 1 durch den Anzeigeregister-Wert x.

Nachstehend ein Beispiel für alle drei Funktionen: $\sqrt{4 \div (\frac{1}{5})^2} = 50$

Taste	Anzeige	Bemerkungen
CLR	0	mögliche laufende Berechnungen werden gelöscht
4 \sqrt{x}	2.	$\sqrt{4}$
\div 5 $1/x$	0.2	1/5
x^2	0.04	$(1/5)^2$
$=$	50.	Ergebnis

Potenzen und Wurzeln — y^x

Mit dieser Taste kann jede positive Zahl in eine Potenz erhoben und darüberhinaus die Wurzel einer positiven Zahl ermittelt werden.

Potenzen (y^x)

- Eingabe der Zahl (y), die in eine Potenz erhoben wird
- Taste y^x
- Eingabe der Potenz (x)
- Taste $=$ (oder eine beliebige andere Operationstaste)

Beispiel: Berechnung von 2^6

Taste	Anzeige
CLR	0
2 y^x 6 $=$	64.

Wurzeln ($\sqrt[x]{y}$)

- Eingabe der Zahl (y), deren Wurzel berechnet werden soll
- Tasten INV y^x
- Eingabe der Wurzel (x)
- Taste $=$ (oder eine beliebige andere Operationstaste)

Beispiel: Berechnung von $\sqrt[6]{64}$

Taste	Anzeige
CLR	0
64 INV y^x 6 $=$	2.

ANMERKUNG: Für y dürfen nur positive Werte eingegeben werden. Bei negativen Eingaben blinkt die Anzeige.



Logarithmen -- $\ln x$, \log

Logarithmen sind mathematische Funktionen, die in einer Vielzahl technischer und theoretischer Berechnungen Eingang finden. Grundsätzlich gilt, wenn $x = y^2$, dann ist $\ln x$ (zur Basis y) = 2. Die unten beschriebenen Tasten erlauben den direkten Zugang zu den Logarithmen einer beliebigen positiven Zahl - ohne Einfluß auf laufende Berechnungen - und ohne Nachschlagen in unhandlichen Logarithmentafeln.

$\ln x$ – NATÜRLICHER LOGARITHMUS – Berechnung des natürlichen Logarithmus (Basis $e = 2.718281828459$) der Zahl im Anzeigeregister. (Ist diese Zahl negativ oder Null, blinkt die Anzeige). Der Antilogarithmus (Numerus) des natürlichen Logarithmus (e^x) wird mit der Folge INV $\ln x$ berechnet.

2^{nd} \log – ZEHNERLOGARITHMUS – Berechnung des Zehnerlogarithmus (Basis 10) des Wertes im Anzeigeregister. (Wieder muß der Wert in der Anzeige positiv sein.) Der Antilogarithmus (Numerus) des Zehnerlogarithmus (10^x) wird mit den Tasten INV 2^{nd} \log ermittelt.

Beispiel: Berechnung des natürlichen Logarithmus von ($e^{2.7} + 10^{1.2}$)

Taste	Anzeige	Bemerkungen
CLR	0	Löschen möglicher laufender Berechnungen
$($ 2.7 INV $\ln x$	14.87973172	Berechnung von $e^{2.7}$
$+$ 1.2 INV 2^{nd} \log	15.84893192	Berechnung von $10^{1.2}$
$)$	30.72866365	die unvollständige Addition wird abgeschlossen
$\ln x$	3.425195888	Ergebnis



Winkelmodus – Tasten **Deg**, **Rad**, **Grad**

Der Rechner ist so ausgelegt, daß auch eine Reihe von Winkelberechnungen durchgeführt werden können – in der Hauptsache trigonometrische Funktionen und polar-/rechtwinklige Umrechnungen. Winkel können in Altgrad, im Bogenmaß oder in Neugrad gemessen werden. Beim Einschalten ist der Rechner grundsätzlich auf Altgrad eingestellt. Sie können jedoch mit dem nachstehenden Tastenfolgen nach Belieben eine der drei gebräuchlichen Maßeinheiten für Winkel wählen:

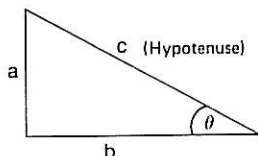
2nd Deg – WINKLEINHEIT ALTGRAD – In diesem Modus werden alle Winkel in Altgrad eingegeben oder berechnet, bis eine andere Einheit gewählt wird. (1 Altgrad = 1/360 eines Kreises – ein rechter Winkel entspricht 90°.)

2nd Rad – WINKLEINHEIT IM BOGENMASS (RADIANT) – In diesem Modus werden alle Winkel in Radiant gemessen. (Ein Radiant entspricht 1/2 π eines Kreises – ein rechter Winkel entspricht $\pi/2$ rad.)

2nd Grad – WINKLEINHEIT NEUGRAD (GON) – In diesem Modus werden alle Winkel in Neugrad angegeben. (ein Neugrad entspricht 1/400 eines Kreises – ein rechter Winkel entspricht 100 Neugrad.)

Tasten für trigonometrische Funktionen – **sin**, **cos**, **tan**

Mit diesen Funktionen werden direkt der Sinus, der Kosinus und der Tangens des Winkels im Anzeigeregister berechnet. Der Winkel wird in der Maßeinheit des gewählten Winkelmodus angegeben.



$$\sin \theta = \frac{a}{c}$$

$$\cos \theta = \frac{b}{c}$$

$$\tan \theta = \frac{a}{b}$$

wobei a, b und c die Seitenlängen sind.

Die Folgen **INV sin**, **INV cos** und **INV tan** dienen zur Berechnung der Arkusfunktionen. Die daraus resultierenden Winkel werden wiederum in den entsprechend vorgewählten Maßeinheiten angezeigt.

Wenn als Winkleinheit Altgrad gewählt wurde, werden alle Winkel in Dezimalform wiedergegeben. (siehe auch Abschnitt V)



UMRECHNUNGEN

Altgrad-Umformungen — **D.MS**

Es gibt zwei verschiedene Darstellungsweisen für Winkel in Altgrad.

Eine Methode ist die Darstellung in Dezimalgrad, GGG.dd. Hier steht GGG für den ganzzahligen Teil des Winkels, während der Dezimalbruchteil als .dd bezeichnet wird. (Sie können bis zu 10 Stellen verwenden.)

Die zweite Methode ist die Darstellungsform in Grad, Minuten und Sekunden, GGG.MMSSsss. Wieder steht GGG für den ganzzahligen Teil des Winkels. MM bezeichnet die Minuten und SS die Sekunden. Wenn eine noch größere Genauigkeit erwünscht ist, können in der Position sss Sekundenbruchteile eingegeben werden. Beachten Sie, daß das Dezimalkomma Grad und Minuten trennt.

Um Grad-Minuten-Sekunden in Dezimalgrad umzurechnen, gibt man den Winkel ein (GGG.MMSSsss) und drückt **2nd** **D.MS** . Mit **INV** **2nd** **D.MS** werden Dezimalgrad in Grad-Minuten-Sekunden umgeformt.

Für die Minuten und für die Sekunden sollten jeweils zwei Stellen eingesetzt werden, da beim Bruchteil der Eingabe der Rechner immer zwei Stellen gleichzeitig beachtet. Endungs-Nullen müssen nicht eingegeben werden. Sehen Sie auch das Beispiel an:

Beispiel: Formen Sie $54^{\circ} 02' 09.6''$ in die Dezimalentsprechung um, und errechnen Sie aus dem Ergebnis wieder Grad-Minuten-Sekunden.

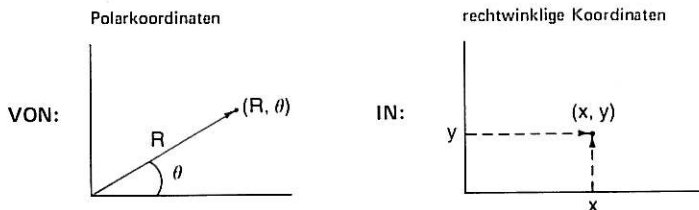
Taste	Anzeige	Bemerkungen
54.02096 2nd D.MS	54.036	GG.ddd
INV 2nd D.MS	54.02096	GG.MMSSs

Dasselbe Verfahren läßt sich auch auf die Umrechnung von Stunden-Minuten-Sekunden in Dezimalstunden übertragen.



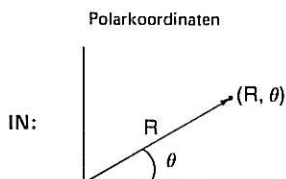
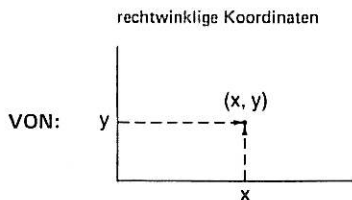
Polar/Rechtwinklige Umrechnungen — **P↔R**

Diese Möglichkeit Ihres Rechners ist von besonderem Vorteil in wissenschaftlichen und technischen Bereichen. Im Zusammenwirken mit der Taste **↔t** ist es ohne viel Zeitaufwand und Mühe möglich, Polarkoordinaten in rechtwinklige Koordinaten umzuwandeln und umgekehrt. Beachten Sie nur die unten angegebenen Tastenfolgen.



Umrechnung von Polarkoordinaten in rechtwinklige Koordinaten:

- Eingabe des Wertes für „R“
- Taste **↔t**
- Eingabe des „θ“-Wertes (Achten Sie auf die richtige Winkleinheit)
- Taste **2nd** **P↔R** zur Anzeige von „y“
- Taste **↔t** zur Anzeige von „x“

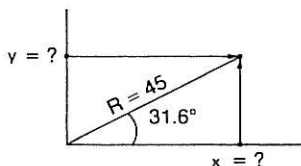


Umrechnung von rechtwinkligen Koordinaten in Polarkoordinaten:

- Eingabe des „x“-Wertes
- Taste $\boxed{x \leftrightarrow t}$
- Eingabe des „y“-Wertes
- Tastenfolge $\boxed{INV} \boxed{2nd} \boxed{P \rightarrow R}$ zur Anzeige von θ in der gewählten Winkereinheit
- Taste $\boxed{x \leftrightarrow t}$ zur Anzeige von „R“

„R“ wird jetzt ausgewiesen.

Beispiel:



Rechnen Sie $R = 45$ Meter, $\theta = 31,6^\circ$
in rechtwinklige Koordinaten um.

Taste	Anzeige	Bemerkungen
$\boxed{CLR} \boxed{2nd} \boxed{Def}$	0	Löschen möglicher laufender Berechnungen und Wahl der Winkereinheit in Altgrad
45 $\boxed{x \leftrightarrow t}$	0.	R wird im T-Register gespeichert*
31.6	31.6	Eingabe des Winkels
$\boxed{2nd} \boxed{P \rightarrow R}$	23.57936577	Umrechnung in rechtwinklige Koordinaten und Anzeige von y
$\boxed{x \leftrightarrow t}$	38.32771204	Anzeige von x (y ist jetzt im T-Register)

ANMERKUNG: In dieser Umrechnung verwendet man ein besonderes Register, das T-Register, das über die Taste $\boxed{x \leftrightarrow t}$ (x-t-Austauschtaste) zugänglich ist. Die Anwendungsmöglichkeiten dieses Registers sind in den verschiedenen Programmierabschnitten erklärt.



TASTEN FÜR STATISTISCHE FUNKTIONEN

Mittelwert, Varianz und Standardabweichung

Vielleicht müssen Sie oft große Gruppen von Datenpunkten bearbeiten, die einen bestimmten Faktor oder einen Parameter einer Anzahl von Einzelpositionen darstellen. (Diese Daten könnten zum Beispiel Verkaufsziffern oder Testergebnisse sein.) Die gebräuchlichsten statistischen Berechnungen, mit denen solche Daten auf einige repräsentative Zahlen reduziert werden, sind Mittelwert, Varianz und Standardabweichung. Der Mittelwert ist der Durchschnitt Ihrer Daten – eine Maßzahl für den zentralen Wert. Varianz und Standardabweichung geben Aufschluß darüber, wie variabel die Daten sind – eine Maßzahl für die Differenz der Daten vom Mittelwert.

Vollständige Informationen über die Anwendung dieser Funktionen finden Sie im Kapitel Statistik im Abschnitt V.

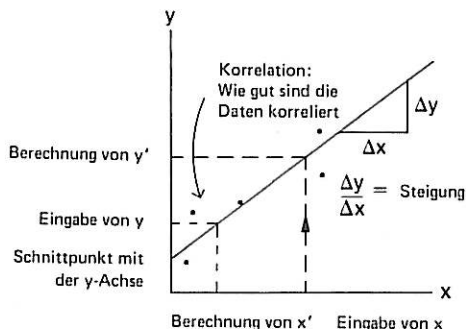


Lineare Regression

Der Begriff lineare Regression mag einen hochwissenschaftlichen Klang haben – aber das Verfahren wird mit Ihrem Rechner äußerst einfach. Und – es geht dabei um eines der ältesten Anliegen überhaupt – um die Vorhersage künftiger Ereignisse.

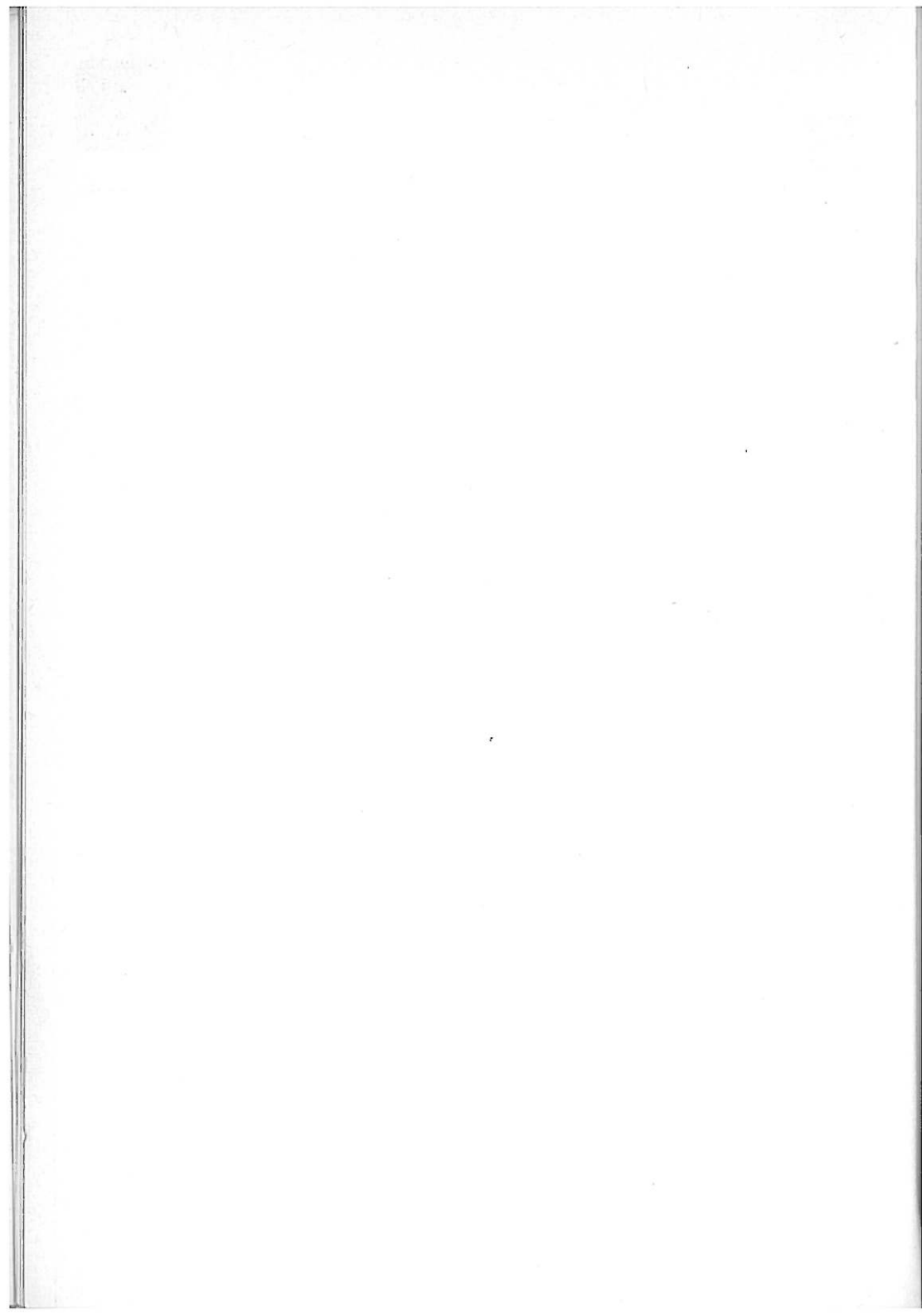
Bei der linearen Regression sind die Daten gewöhnlich als Variablenpaare ausgedrückt, die auf ein Diagramm aufgetragen werden könnten. Ein Punktepaar wird im allgemeinen mit den Buchstaben x, y bezeichnet, (wobei x zum Beispiel die Werbeausgaben und y die Umsatzziffern sein können, oder x ist ein Testergebnis und y eine Leistungsspitze in dem Gebiet, etc.). Sie wollen eine Voraussage für einen x -Wert, den Sie auswählen – was geschieht mit y (oder umgekehrt)? Der Rechner löst für Sie die Aufgabe durch mathematische Ermittlung der best-angepaßten Geraden, die durch Ihre Datenpunkte verläuft. Die Gerade kann dann für Vorhersagen verwendet werden.

Sobald die Daten eingegeben sind, kann der Rechner die beste Anpassungsgerade durch diese Punkte ziehen und folgende Information ausgeben:



Statistische Berechnungen mit nicht linearen Anpassungskurven, zum Beispiel mit Exponentialkurven für Grundgesamtheiten, können ebenfalls mit diesem Rechner durchgeführt werden.

Diese und andere Merkmale sind im Kapitel Statistik, Abschnitt V, ausführlich beschrieben.





ANWENDUNG DER "EINGEBAUTEN" PROBLEMLÖSUNGEN

ZUGRIFF AUF DIE SOLID-STATE-SOFTWAREPROGRAMME

Ob Ihnen der Begriff "Software" vertraut ist oder nicht – er hat tatsächlich eine Vielzahl von Definitionen. Grundsätzlich versteht man unter Software Befehle und Programme – Dinge, die im allgemeinen schriftlich formuliert werden können – die den Computer oder Rechner anweisen, was auszuführen ist, und die für Sie Informationen über die Anwendungsweise enthalten. Der Rechner verfügt über eine Auswahl zweckmäßiger, aber einfach anzuwendender Programme, die in den Rechner eingesetzt und mit einem Tastendruck genutzt werden können. Diese Programme – sie sind für eine Vielzahl von Fachgebieten sehr anwender-orientiert geschrieben – sind in einem speziellen Programm-Modul im Rückteil des Rechners gespeichert. Dieser Modul kann einfach gegen einen anderen ausgetauscht werden. Die Programminformationen sind ausschließlich in einem winzigen Solid-State Silizium-Chip enthalten – der ähnlich aufgebaut ist wie der integrierte Schaltkreis auf Siliziumbasis, der die zentrale Steuerung des Rechners darstellt. Daher der Begriff "Solid-State-Software"-Programm. Eine Menge von Software-Programmen sind für Sie im Solid-State-Softwaremodul gespeichert. Und sie haben Eigenschaften, die die Anwendung einfach machen. Die Vorteile:

- große Programmkapazität auf engstem Raum – leicht zu transportieren und anzuwenden.
- der Zugriff auf die Software-Programme ist jederzeit über die Tastatur möglich.
- die Software-Programme sind so geschrieben, daß auch der Anfänger sie ohne Schwierigkeiten anwenden kann.

Um die Effektivität jedes Moduls vollständig zu machen, gibt es ein entsprechendes Handbuch. Alle programmbezogenen Informationen finden Sie in diesen handlichen Anleitungen.

PROGRAMMSAMMLUNGEN

Im Lieferumfang ist eine Programmsammlung mit einer Standardauswahl zweckmäßiger Programme enthalten. Andere fachorientierte Sammlungen erhalten Sie dort, wo Sie Ihren Rechner gekauft haben oder direkt auf Bestellung von Texas Instruments. Jede Sammlung enthält eine Programmauswahl, die es leicht macht, einige der mathematischen Techniken in den verschiedenen Fachgebieten zu nutzen. Eine Programmsammlung besteht aus dem Modul, dem Handbuch mit detaillierten Informationen zur Anwendung jedes Programms, einem Behälter zum Aufbewahren, und einem Satz Programmlabel-Karten. Jede Software-Programmsammlung, die für die Typen TI-58/59 entwickelt wurde, ist auch für den TI-58C voll kompatibel.

DER SOFTWARE-MODUL

Die Programme jeder Sammlung sind in den Software-Modulen gespeichert, ein Modul pro Sammlung. Der Modul kann im Rückteil des Rechners eingesetzt und sofort verwendet werden. Module sind abnutzungsfeste Betriebssysteme, die aber im Interesse einer langen Lebensdauer sorgfältig behandelt werden müssen.

VORSICHT

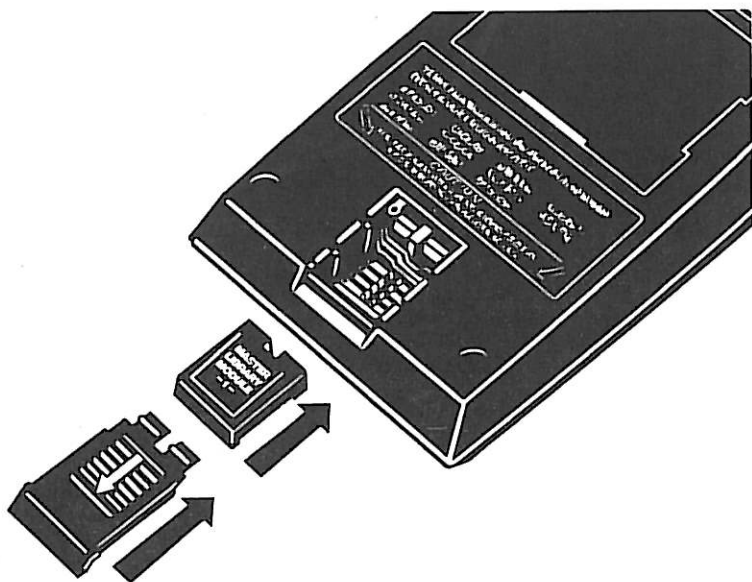
Ihr Körper darf nicht statisch aufgeladen sein, wenn Sie einen Modul auswechseln.

Dies trifft insbesondere dann zu, wenn das Ladegerät angeschlossen ist, weil es den Rechner erdet. Berühren Sie einfach einen Metallgegenstand, um eine eventuelle statische Aufladung Ihres Körpers abzubauen. Die Informationen im Modul können durch statische Entladungen stark beschädigt werden. Mehr über die Wartung der Module siehe Anhang A.



Der Modul für die Standardprogramme wird noch vom Hersteller in den Rechner eingebaut, er kann aber leicht gegen einen anderen ausgetauscht werden. Es empfiehlt sich, den Modul im Rechner zu lassen, es sei denn, Sie wollen ihn austauschen. Befolgen Sie nachstehende Anweisungen, wenn Sie den Modul herausnehmen oder austauschen:

- 1) Schalten Sie den Rechner aus. Beim Einsetzen oder beim Ausbau eines Moduls können die Kontakte kurzgeschlossen werden, und den Modul und/oder den Rechner schwer beschädigen, wenn der Rechner bei dieser Tätigkeit eingeschaltet ist.
- 2) Schieben Sie die kleine Abdeckplatte des Modulfachs im rückwärtigen Teil des Rechnerbodens heraus (Siehe Skizze). Wiederum müssen Sie jede statische Aufladung Ihres Körpers abbauen, wenn Sie mit dem Modul arbeiten.
- 3) Nehmen Sie den Modul heraus, Sie können dazu den Rechner umdrehen und den Modul in Ihre Hand fallen lassen.
- 4) Schieben Sie den Modul mit dem eingekerbten Ende zuerst und mit der Beschriftung oben in das Fach. Der Modul muß ohne Kraftaufwand in die richtige Position gleiten.
- 5) Schieben Sie die Abdeckplatte wieder auf das Fach, damit der Modul einwandfreien Kontakt hat.



Vermeiden Sie jede Aktion, die die Kontakte verbiegen, verunreinigen oder auf andere Weise beschädigen könnte.



ABLAUF DER SOFTWAREPROGRAMME

Wenn ein Software-Modul in den Rechner eingesetzt ist, kann ein bestimmtes Programm einfach durch Angabe seiner Nummer aufgerufen werden. (Jedes Programm einer Sammlung hat eine eigene Nummer.) Die Tastenfolge ist **[2nd] [Pgm] mm**, wobei mm die zweistellige Nummer des ausgewählten Programms ist.

Befolgen Sie genau die Anweisungen im Programmhandbuch, wenn Sie ein Programm anwenden. Für jedes Programm ist eine Programmlabel-Karte beigelegt. Diese nicht-magnetische Karte gibt die Zuordnungen der Programmadress-tasten an, und kann in das Sichtfenster über diesen Tasten eingelegt werden, wenn man sie vom Labelbogen abtrennt.

Beispiel: Welchen künftigen Wert hat eine Einlage von DM 1000 nach 20 Jahren bei einem Jahreszinssatz von 8% ?

Mit dem Zinseszinsprogramm in der Standard-Programmsammlung können Sie diese Aufgabe sofort lösen.

Taste	Anzeige	Bemerkungen
[CLR] [2nd] [Pgm] 18	0.	Der Rechner geht auf das Zinseszinsprogramm (PGM 18)
[2nd] [E']	0.00	Einleitung
20 [A]	20.00	Eingabe der Anzahl der Perioden
8 [B]	8.00	Eingabe des Zinssatzes
1000 [C]	1000.00	Eingabe des Barwertes
0 [D]	4660.96	Berechnung des künftigen Wertes

Sie können das Programm immer wieder ohne Wiederholung des PGM-Befehls ablaufen lassen, sobald es einmal ausgewählt ist. Drücken Sie **[RST]** oder **[2nd] [Pgm] 00**, um auf den Programmspeicher und auf Tastaturoperation zurückzugehen, oder lassen Sie ein anderes Softwareprogramm ablaufen, das Sie über die Taste **[Pgm]** wählen.

Softwareprogramme können auch durch andere Programme aufgerufen werden (siehe Unterprogramme in Abschnitt IV). Diese Eigenschaften stellt eine beträchtliche Erhöhung der Programmierkapazität Ihres Rechners dar.

Wenn Sie zu einem beliebigen Zeitpunkt wissen wollen, welcher Softwaremodul gerade eingesetzt ist, drücken Sie **[2nd] [Pgm] 1 [SBR] [2nd] [Wtr]** (oder beim TI-58/58C die Folge **[2nd] [Pgm] 1 [SBR] [2nd] [R/S]**). Damit wird die Nummer des Moduls angezeigt (und zusammen mit der Modulbezeichnung ausgedruckt, wenn der Rechner an einen PC-100A, B oder C angeschlossen ist).

Die beigelegten Programmlabel-Karten dienen zur Bezeichnung der Programmadress-Tasten, wenn mit den programmierbaren Rechnern TI-58/58C ein Programm erstellt wird.



ANALYSE VON SOFTWAREPROGRAMMEN (PROGRAMMÜBERNAHME)

Im allgemeinen sind die Softwareprogramme für den unmittelbaren Zugriff auf ihre Module begrenzt. Bei Anwendung eines dieser Programme stellt sich der Verarbeitungsfluß direkt auf den Modul ein und führt die Aufgaben durch. Der Zugang zu einem Softwareprogramm ist jedoch möglich, wenn man es in den Programmspeicher einbringt. Damit können Sie alle Programmierstechniken des Rechners einsetzen, um die einzelnen Schritte zu analysieren und das Programm nach individuellem Bedarf zu ändern. Tatsächlich wird aber nur eine Kopie des Programms in den Programmspeicher übernommen, der Informationsgehalt des Moduls kann nie geändert werden. Das Verfahren der Programmübernahme ist einfach.

- 1) Kontrollieren Sie, ob genügend Programmspeicherkapazität für die Übernahme des Programms verfügbar ist. Siehe Speicherbereichsverteilung im nächsten Abschnitt.
- 2) Mit der Folge **2nd** **Pgm** mm bestimmen Sie, welches Programm zu übernehmen ist.
- 3) Drücken Sie **2nd** **Op** 09, um das Programm in den Programmspeicher zu kopieren.

Dieses Verfahren bringt das gewählte Programm, beginnend mit Speicherplatz 000, in den Programmspeicher. Das übernommene Programm überschreibt alle zuvor in diesem Teil des Programmspeichers enthaltenen Befehle. Aus diesem Grund darf die Anweisung zur Übernahme eines Softwareprogrammes in den Programmspeicher nicht Bestandteil eines Programms sein.

Sobald das Programm in den Programmspeicher übernommen ist, können Sie es für Ihre Zwecke verändern. Das modifizierte Programm können Sie jedoch nicht in den Software-Modul übertragen. Wenn das neue Programm erhalten bleiben soll, besteht die Möglichkeit, jeden Schritt in der Kodeform einzutragen, das Programm auf Magnetkarte (oder Karten) aufzuzeichnen (nur TI-59) oder mit dem Drucker eine Auflistung zu erstellen. Der programmierbare TI-58C hält ein Programm solange gespeichert, bis Sie es ändern oder bis die Batterieversorgung für längere Zeit ausfällt.

Wenn ein Software-Modul rechtlich geschützte Programme enthält, können diese Programme ebenfalls gegen die Übernahme in den Programmspeicher geschützt werden. Bei der Anweisung, eines dieser Programme zu übertragen, blinkt die Anzeige.

Bei Störungen des Software-Moduls siehe Anhang A.



DER BEGRIFF PROGRAMMIERUNG

Der Einfluß der Computer auf das tägliche Leben ist so groß, daß Begriffe wie „Programmierer“, „Computer-Programmierung“, „Programmiersprache“ oder nur „Programmierung“ jedem geläufig sind. Manche Leute verbinden mit diesen Begriffen die Vorstellung von hochintellektuellen Personen, die mit komplizierten Aufgaben betraut sind, und allein der Gedanke, Programmierer zu werden, scheint ohne intensives Training nicht im Bereich der Möglichkeit zu liegen.

Nicht so — die Zeit für eigenes Programmieren ist hier und jetzt angebrochen. Rechnerprogrammierung ist einfach. Und es ist verblüffend, daß wirklich jedermann nach ein paar leichten Lektionen einen Rechner programmieren kann. Die programmierbaren Rechner von Texas Instruments machen die Programmierung einfach und problemlos. Ihr Rechner ist vielseitig genug, um Ihnen die Freude an der Rechengeschwindigkeit und Leistung durch Programmieren zu geben — ob Sie nur einfache Arithmetik anwenden oder als Flugingenieur äußerst komplexe mathematische Probleme bearbeiten müssen. Die Rechenleistung ist für jeden da, aber nutzen Sie das, was für Ihren Bedarf gefordert ist. Sie werden erstaunt sein, wie schnell und einfach zeitraubende Probleme mit reiner Arithmetik und simplen Programmen zu lösen sind.

Programmieren heißt logisch denken. In einfachste Worte gekleidet, ist ein Programm eine Gruppe von Befehlen, mit denen ein Gerät oder eine Person angewiesen wird, wie etwas getan werden muß. Ein Rechnerprogramm weist demnach einen Rechner an, wie er etwas bearbeiten muß, insbesondere, wie Berechnungen durchzuführen sind. Wenn der Rechner für Sie eine Aufgabe übernehmen soll, müssen Sie nur genaue Anweisungen geben, was getan und wie es getan werden muß. Ein Programm ist eine Reihe präziser Befehle, die in bestimmter Reihenfolge genau und zuverlässig durchgeführt werden müssen.

Eine Sprache ist das bloße Kommunikationsmittel mit dem Rechner. Selbst für die Kommunikation mit dem einfachsten Vierfunktionsrechner gibt es eine Sprache. Angewandt auf die Programmierung ist Sprache das notwendige Mittel, um dem Rechner Ihr Programm mitzuteilen.

Eine Rechnersprache ist primär eine Sprache der logischen Vernunft und der Arithmetik. Wenn Sie also Erfahrungen mit arithmetischen Berechnungen haben, ob mit Papier und Bleistift oder mit einem Rechner, kennen Sie schon einen Großteil der Programmiersprache Ihres Rechners. Die Funktionen, die in diesem Buch für Tastaturoperationen erklärt sind, lassen sich ebenso in Programmen anwenden.

Ein Rechner führt wie ein Computer mit höchster Zuverlässigkeit die und nur die Befehle aus, die gegeben werden. Diese Eigenschaft führt bei der Arbeit mit diesen Geräten zu unterschiedlichen Erfahrungswerten. Die Folge ist, daß der Programmierer streng darauf achten muß, welche Anweisungen der Rechner erhält und in welcher Reihenfolge die Befehle abgeschlossen werden müssen. Der Rechner führt genau das aus, was Sie angewiesen haben, gleich, ob Sie die Verarbeitung in dieser Form wünschen oder nicht. Die hier erläuterten Techniken geben Ihnen Einblick in die hohe Leistungsfähigkeit Ihres Rechners und lassen Sie teilhaben an der Ära der eigenen individuellen Programmierung.



GRUNDELEMENTE DER PROGRAMMIERUNG

Einbringen einer Variablen in ein Programm

Betrachten Sie folgenden einfachen Ausdruck:

$$A + B = C$$

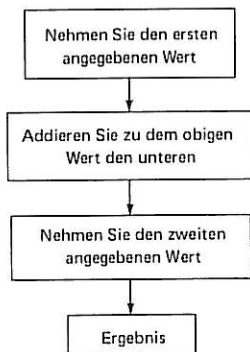
Mit einem Vierfunktionsrechner können die Werte für A und B nicht später bestimmt werden. Sie müssen bereits beim Eintasten des Ausdrucks bekannt sein. Nachdem der erste Ausdruck eingetastet und ein Ergebnis ausgewiesen ist, müssen Sie den gesamten Ausdruck neu eingeben, wenn einer oder beide Werte geändert werden. Mit dem programmierbaren Rechner können Sie die Befehle eintasten, ohne zunächst die Werte zu bestimmen, und Sie erhalten später durch bloße Eingabe der geänderten Werte ein Ergebnis.

In dem obigen einfachen arithmetischen Ausdruck kann die Rechengeschwindigkeit des Vierfunktionsrechners der des programmierbaren Rechners gleichkommen. Die wirklichen Vorteile des programmierbaren Rechners werden in einem Ausdruck wie unten deutlich. Voraussetzung sei, daß Sie ein Ergebnis für 10 verschiedene A-Werte brauchen, und B und C sollen unverändert bleiben.

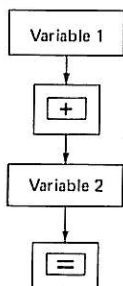
$$A \times (B \div (1 + A)^{-C}) = \text{ERGEBNIS}$$

Sie wollen die Gleichung einmal eingeben und dann nur den A-Wert für jede Berechnung ändern — eine leichte Aufgabe mit Ihrem programmierbaren Rechner.

Noch einmal zurück zu dem ersten Ausdruck: Überlegen Sie, wie man dem Rechner Befehle geben kann. Schreiben Sie zuerst die Befehle so auf, als würden sie einer anderen Person gelten, und setzen Sie diese Anweisungen dann in Rechnerbefehle um.



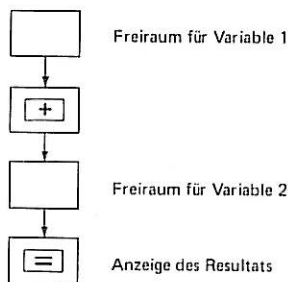
A und B sind beliebige Werte — sie können variieren. Derartige Werte werden auch als Variable bezeichnet.



Was den Rechner betrifft, so müssen zwei Punkte erledigt werden, wenn Sie die Variable nicht als Teil des Programms eingeben:

1. An der richtigen Stelle in Ihrer Befehlsfolge muß ein Freiraum gelassen werden, wo Sie die Variable später einbringen können.
2. Sie müssen den Rechner anweisen, wo der Wert der Variablen bei Bedarf aufgefunden wird. Dieser Befehl kann beinhalten, daß der Rechner eine Variable entweder der Anzeige oder einem der Datenregister entnehmen muß.

Nachstehend ist die Befehlsfolge neu skizziert, wobei die Freiräume für die Variablen berücksichtigt werden.

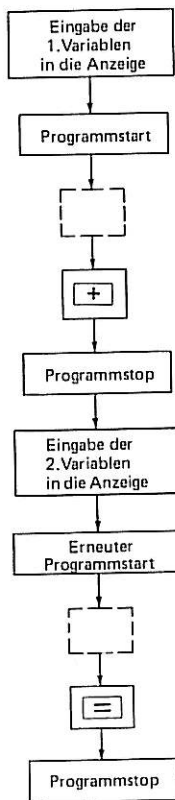


Jetzt ist der Freiraum geschaffen, wenn der Rechner eine angezeigte Zahl als Variable benötigt. Beginnt der Rechner dann einen Programmdurchlauf, wird der jeweilige Anzeigewert in den Freiraum eingebracht. Auch der Wert für die zweite Variable muß in der Anzeige aufgefunden werden, also müssen Sie das Programm unterbrechen, ehe die Variable gebraucht wird und die Variable in die Anzeige eingeben. Wenn dann der Programmablauf wieder gestartet wird, nimmt der Rechner den Anzeigewert auf und setzt das Programm fort.

Ein Verfahren, in Ihrem Programm Freiräume für neue Eingaben oder Daten zu schaffen, ist die Programmunterbrechung an dieser Stelle mit der Taste [R/S] (Run/Stop). Sie weisen das Gerät an, „alle Informationen zu halten“, sodaß Sie den nächsten benötigten Wert in die Anzeige eintasten können. (Man kann von einem scheinbaren Freiraum sprechen, da zwischen den Programmbefehlen in Wirklichkeit keine Lücke offen ist. Durch die Unterbrechung des Ablaufs an einer Stelle im Programm wird angedeutet, daß hier etwas durchgeführt werden soll – in diesem Fall eine Dateneingabe.)



IV



manuell über die Tastatur

manuell über die Tastatur

Die erste Variable wird aus der Anzeige in den scheinbaren Freiraum aufgenommen und hier eingefügt.

Programmbefehl

Programmbefehl-Unterbrechung für die Eingabe der nächsten Variablen.

manuell über die Tastatur

manuell über die Tastatur

Die zweite Variable wird aus der Anzeige in den scheinbaren Freiraum aufgenommen und hier eingefügt.

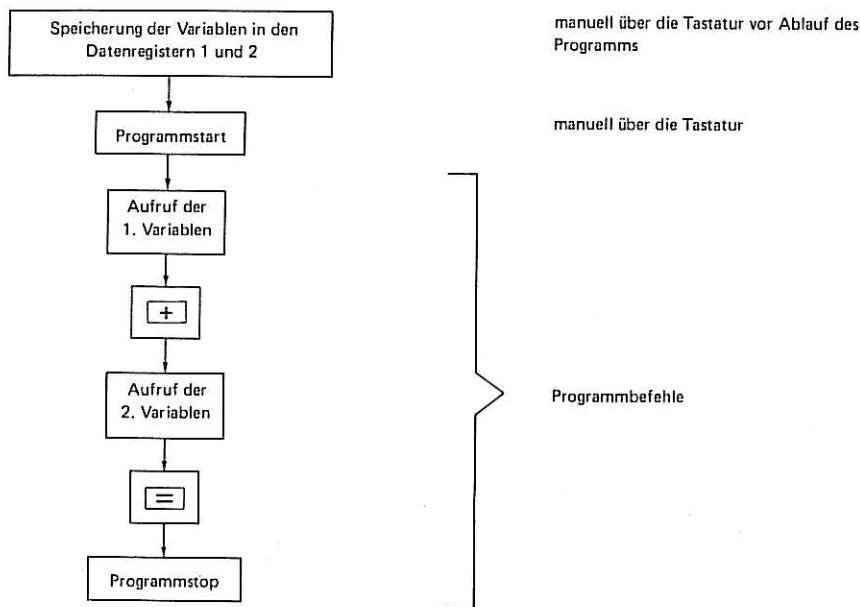
Programmbefehl

Anzeige des Resultats

Variable in der Anzeige — Ablaufdiagramm

Die obige Methode (Programmunterbrechung zur Dateneingabe) ist ideal, wenn für jeden Programmlauf eine vollständig neue Variablengruppe eingesetzt werden muß. Sie bevorzugen vielleicht ein anderes Verfahren, wenn nur ein Wert zu ändern ist. Bei diesem Vorgang verwenden Sie die Datenregister des Rechners, um die Variablen zu speichern.

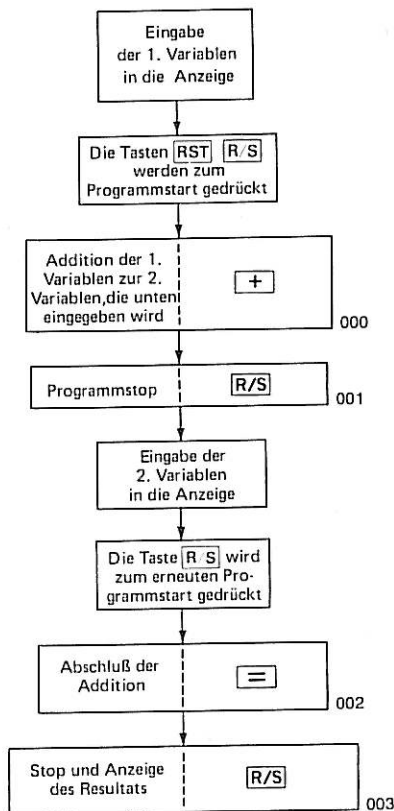
Wenn der Rechner eine Variable im Speicher auffinden soll, müssen Sie einen Befehl programmieren, mit dem die Variable aus dem entsprechenden Datenregister in das Programm eingebracht wird. Beispiel: Der Aufruf einer im Datenregister 1 gespeicherten Variablen wird mit der Folge **RCL 01** ausgeführt.



Variable im Datenspeicher – Ablaufdiagramm

Zunächst eine kurze Zusammenfassung der bisherigen Ausführungen: Zuerst wurde die Aufgabe definiert. Dann folgten Überlegungen zu den Eingabemethoden für die Variablen (mit oder ohne Speicher). Drittens wurde für jede Methode ein einfaches Ablaufdiagramm entwickelt. Beachten Sie, daß ein Ablaufdiagramm durch graphische Aufteilung des Problems in Einzelschritte oder -aktionen entsteht, die das Problem lösen, wenn man sie von oben nach unten durchführt.

Der nächste wichtige Schritt ist die Anwendung des Ablaufdiagramms als Hilfsmittel, mit dem die Tastenbefehle bestimmt werden, die der Rechner zur Lösung der Aufgabe benötigt. Beachten Sie, daß in den folgenden Beispielen vor Ablauf des Programms die Taste **RST** gedrückt werden muß. Damit wird sichergestellt, daß das Programm tatsächlich mit dem Befehl im Speicherplatz 000 beginnt, wenn Sie das Programm mit der Taste **R/S** starten. Das folgende Beispiel zeigt die nötigen Tastenbetätigungen, mit denen der Rechner angewiesen wird, die Variablen dem Anzeigeregister zu entnehmen.



Programm mit den Variablen in der Anzeige

Die mittleren Rechtecke in jedem Ablaufdiagramm zeigen auf, was Sie ausführen müssen, damit das Programm abläuft, nachdem die Befehle in den Programmspeicher eingegeben sind. Diese Befehle oder Tastenbetätigungen sehen Sie aus den rechten Hälften der durch gepunktete Linien geteilten Rechtecke. Die Zahlen außerhalb der Rechtecke sind die Nummern der Befehle entsprechend den Tastenbetätigungen innerhalb des Rechtecks. Diese Tasten- oder Programmbefehle können in den Programmspeicher eingegeben werden, wenn Sie den Rechner in den Learn-Modus schalten.



Im folgenden ein Verfahren für die Eingabe von Programmen:

1. Drücken Sie die Tasten **[2nd]** **[CP]** um den Programmspeicher zu löschen.
2. Schalten Sie den Rechner mit der Taste **[LRN]** in den Learn-Modus. Dieser Modus ist durch ein besonderes Anzeigeformat, 000 00, erkennbar.
3. Drücken Sie der Reihe nach von oben nach unten jede Taste, die im Ablaufdiagramm angegeben ist. Drücken Sie nur die dargestellten Tasten. Wenn Sie einen Fehler machen drücken Sie **[LRN]** und beginnen Sie wieder mit Schritt 1. Veränderungen in der Anzeige sind später erklärt.
4. Drücken Sie die Taste **[LRN]** erneut, um den Learn-Modus abzustellen. Damit wird auch das Sonderformat der Anzeige aufgehoben, und nur eine einzelne Null wird ausgewiesen. Jetzt können Sie das Programm ablaufen lassen.

Die drei linken Stellen in der Anzeige müssen sich bei Eingabe eines Programms ändern. Diese drei Stellen zeigen, auf welchen Programmspeicherplatz (Befehlsnummer) der Programmzeiger gerichtet ist. Der Programmzeiger ist ein internes Betriebssystem des Rechners, um beim Programmablauf zu bestimmen, welcher Befehl als nächster auszuführen ist. Im Learn-Modus weist der Programmzeiger auf den nächsten, nichtbelegten Speicherplatz im Programmspeicher.

Mit diesen Kenntnissen können Sie jetzt mit Ihrem Rechner ein Programm versuchen, bei dem die Variablen über die Anzeige eingebracht werden.

1. Schalten Sie den Rechner ein und drücken Sie **[2nd]** **[CP]**.
2. Taste **[LRN]**, um den Rechner in den Learn-Modus zu schalten.
3. Eingabe des Programms durch Eintasten der oben angegebenen Folge.
[+] **[R/S]** **[=]** **[R/S]**
4. Taste **[LRN]**, um den Learn-Modus aufzuheben.

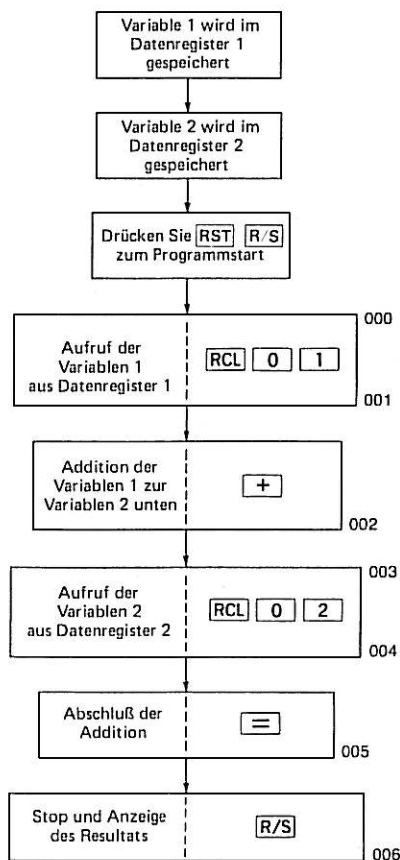
Damit ist der Rechner programmiert. Lösen Sie jetzt mit dem Programm die Aufgabe $227 + 34 = ?$

1. Drücken Sie **[CLR]**, um mögliche laufende Berechnungen zu löschen.
2. Drücken Sie **[RST]**, und geben Sie für die 1. Variable 227 ein.
3. Drücken Sie **[R/S]**. Die Zahl 227 bleibt in der Anzeige.
4. Geben Sie für die 2. Variable 34 ein und drücken Sie erneut **[R/S]**. Das Ergebnis 261 wird angezeigt.

Um falsche Ergebnisse zu vermeiden, empfiehlt es sich, vor Ablauf eines Programms mit der Taste **[CLR]** mögliche unvollständige Berechnungen zu löschen. Jedesmal, wenn man diese Tasten drückt, werden die Werte in der Anzeige addiert und das Ergebnis ausgewiesen. Wiederholen Sie das Programm einmal mit eigenen Zahlen.



Jetzt sollen die Variablen in Datenregistern gespeichert werden. Schreiben Sie dazu ein neues Programm:



Programm mit den Variablen im Datenspeicher



Gehen Sie wie folgt vor und geben Sie das Programm in den Rechner ein:

1. Schalten Sie den Rechner ein und drücken Sie **2nd** **CP**.
2. Drücken Sie **LRN**, um den Rechner in den Learn-Modus zu schalten.
3. Geben Sie das Programm durch Eintasten der nachstehenden Folge ein:

RCL
0
1
+
RCL
0
2
=
R/S

4. Drücken Sie **LRN** erneut, um den Learn-Modus abzustellen.

Dieses Programm entnimmt die Variable den Datenregistern 01 und 02. Speichern Sie also 227 in Register 01 und 34 in Register 02:

- | | |
|----------------------|---------------------|
| 1. 227 in Speicher 1 | 2. 34 in Speicher 2 |
| Eingabe 227 | Eingabe 34 |
| Taste STO 01 | Taste STO 02 |

Wie bereits im Ablaufdiagramm angemerkt wird, ist die einzige erforderliche Tastaturoperation **RST** **R/S**, um das Programm durchzuführen. Wenn diese Tasten gedrückt wurden, erscheint 261 in der Anzeige und das Problem ist gelöst.

Beachten Sie, daß bei Eingabe der Variablen über die Tastatur in die Anzeige weniger programmierte Befehle nötig waren. Folglich wurde weniger Programmspeicherraum beansprucht. Bei Anwendung der Datenregister zur Variablenspeicherung wurden mehr Befehle im Programmspeicher platziert, aber der Rechner ermittelte das Ergebnis ohne Unterbrechungen. Die Wahl der Methode ist eine Frage Ihres persönlichen Bedarfs.

Bedenken Sie den Nutzen der Ablaufdiagramme, vor allem für die Organisation und Darstellung des Lösungswegs für ein bestimmtes Problem. Das Ablaufdiagramm zeigt die Arbeitsgänge, während das Programm abläuft, und umfaßt nicht nur die Befehle oder das Programm im Programmspeicher, sondern veranschaulicht auch die manuellen Tastaturoperationen, die für die Ausführung des Programms erforderlich sind, wie Programmstart und Variableneingabe. Die dargestellten Tastenbetätigungen sind die Befehle, die der Rechner erkennt und nach denen er vorgeht. Sie werden im Learn-Modus im Programmspeicher gespeichert; sie sind im wesentlichen das Programm.



Technik der Programmierung

Die vielseitige arithmetische Sprache erlaubt einfache, aber auch hoch-komplexe Programmierung. Einfache Programme können ohne viel Mühe und ohne Probleme eingegeben, getestet und durchgeführt werden. Aber obwohl die Sprache so direkt und einfach wie möglich aufgebaut ist, erfordern komplexe Programme Voraussicht und Planung.

Wenn Sie wenig Programmiererfahrung haben, sollen Ihnen die folgenden Anregungen helfen. Sind Sie mit der Programmierung bereits vertraut, dienen diese Anregungen der Auffrischung Ihrer Kenntnisse und Ihrer Orientierung auf das Rechnerprogrammieren. Betrachten Sie die folgenden Punkte nur als eine Liste von Vorschlägen, denn Sie werden ohne Zweifel einen eigenen Programmierstil entwickeln.

1. **Definieren Sie die Aufgabe klar und sorgfältig.** Bestimmen Sie die Formeln, die Variablen und die gewünschten Resultate. Was ist bekannt? Was muß berechnet werden? In welcher Beziehung stehen bekannte und unbekannte Werte zueinander?
2. **Entwicklung einer Lösungsmethode (Algorithmus).** Definition der Operationsfolge für die gewünschte Zahlenwertbestimmung mit Rücksicht auf die verfügbare Rechen- und Programmierkapazität. (Bedenken Sie, daß nicht der Rechner, sondern Sie die Aufgaben lösen. Der Rechner führt Ihre Lösungen genau nach Ihren Anweisungen aus.)
3. **Entwicklung eines Ablaufdiagramms.** Oft ist es vorteilhaft, mit Hilfe von Skizzen eine visuelle Vorstellung vom Verarbeitungsfluß zu bekommen. Hier kann man das Zusammenwirken zwischen den verschiedenen Lösungsanteilen darstellen und vielleicht sogar die Struktur des Programms vereinfachen.
4. **Beginn mit Datenregisterzuordnungen.** Ordnen Sie den Zahlen, mit denen Sie arbeiten müssen, Datenregister zu. Diese Aufgabe setzen Sie während der gesamten Programmierung fort. Es empfiehlt sich, niemals eine Größe zu speichern, ohne parallel dazu Notizen zu machen, welches Datenregister mit welchem Wert belegt ist.
5. **Umsetzen des Ablaufdiagramms in eine Tastenfolge.** Die Kodeformen sollen Ihre Arbeit erleichtern. Es ist vorteilhaft, alle Labels und Speicherregister in die vorgesehenen Felder auf der Kodeform einzutragen. Nutzen Sie die Spalte für die Anmerkungen zur einfachen Bezugnahme auf die verschiedenen Programmsegmente.
6. **Eingabe des Programms.** Drücken Sie **2nd** **CP** **LRN** und tasten Sie das komplette Programm von der Kodeform ein. Nach Abschluß der Eingabe drücken Sie **LRN** erneut, um den Learn-Modus abzustellen.
7. **Programmtest.** Überprüfen Sie das Programm mit Hilfe von Testaufgaben, mit denen so viele Fälle wie zweckmäßig berücksichtigt werden.
8. **Fehlerkorrektur.** Korrigieren Sie in der Kodeform die Fehler, die Sie beim Programmtest festgestellt haben.
9. **Redigieren des Programms.** Schalten Sie den Rechner in den Learn-Modus, und nehmen Sie die erforderlichen Korrekturen vor. Drücken Sie **LRN** erneut, und gehen Sie auf Tastaturoperation zurück.
10. **Erneuter Programmtest.** Wiederholen Sie die Schritte 7 bis 9 nach Bedarf.
11. **Aufzeichnung des Programms.** Wenn der Rechner mit Magnetkarten arbeiten kann, zeichnen Sie das Programm auf.
12. **Dokumentation der Programm-Instruktionen.** Es ist empfehlenswert, die Anweisungen, wie das Programm anzuwenden ist, Schritt für Schritt sorgfältig niederzuschreiben. Selbst die leistungsfähigsten Programme sind nutzlos, wenn Sie nicht mehr wissen, wie sie gesteuert werden. Tragen Sie auf dem Bogen für die Programminstruktionen detaillierte Informationen ein, wie man das Programm ablaufen lassen kann.



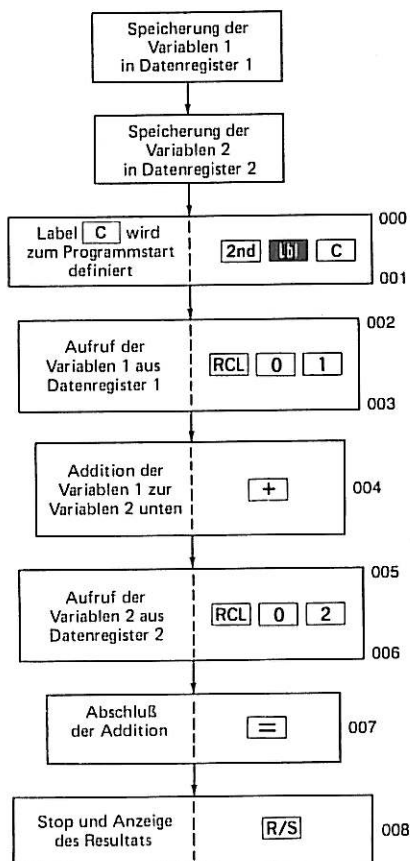
Die Anwendung der Programmadress-Tasten (User-defined labels)

Beim Ablauf der letzten Programmbeispiele wurden jedesmal die Tasten **RST** und **R/S** verwendet. Da **RST** den Programmzeiger wieder auf den Befehl 000 einstellt, haben Sie vielleicht den Schluß gezogen, daß jedes Programm mit dem Anfang des Programmspeichers beginnen muß. Wenn Sie mehr Programmiererfahrung gewinnen, werden Sie feststellen, daß dies nicht immer zweckmäßig ist. Ihr Rechner hat Programmadress-Tasten, die als Labels den leichten Zugriff auf jeden Speicherplatz im Programm ermöglichen.

Bringt man eine Programmadress-Taste in das Programm ein, richtet sich der Programmzeiger auf das Label, sobald man diese Taste drückt. Der Rechner beginnt dann automatisch mit der Durchführung des Programms, wobei der Ausgangspunkt für die Berechnungen der erste Befehl nach dem Label ist. Es handelt sich um die Tasten **A** bis **E** und deren Zweitfunktionen **2nd A** bis **2nd E**; sie erlauben die Kennzeichnung und den Zugriff auf bis zu 10 verschiedene Bezugspunkte (Programme oder Teile von Programmen). Mit einem kleinen Zusatz zum ersten Programmbeispiel kann die Tastenfolge **RST R/S**, die zum Programmstart verwendet wurde, einfach durch **C** oder eine andere Programmadress-Taste ersetzt werden. Da die Funktion der Taste **C** vom Anwender definiert wird, besteht der Programmzusatz einfach in der bloßen Kennzeichnung des Programmstarts mit **C** über die Labeltaste **2nd Lbl**, wie auch im folgenden Diagramm gezeigt wird.

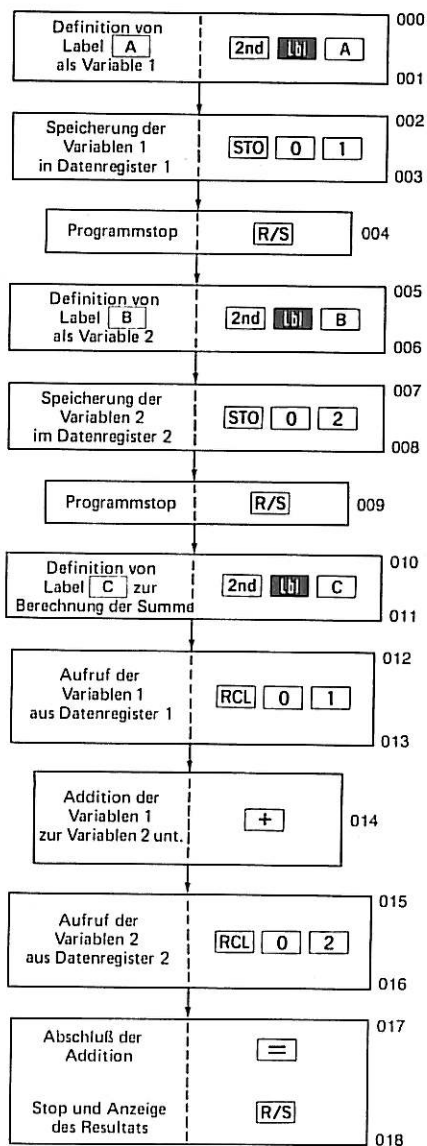


IV



Schalten Sie wie zuvor in den Learn-Modus und geben Sie die dargestellte Tastenfolge ein. Anschließend wird der Learn-Modus aufgehoben und Sie speichern 227 in Datenregister 1 und 34 in Datenregister 2. Jetzt drücken Sie die Taste **C**. Das Ergebnis wird angezeigt, weil mit der Taste **C** der Rechner angewiesen wird, die Position von **Lbl C** im Programmspeicher aufzufinden, und die Befehle nach **Lbl C** durchzuführen.

Um Ihre Vorstellung von der Funktionsweise der Programmdresstasten zu vertiefen, betrachten Sie einen weiteren Zusatz zum vorigen Programm: Damit ist es möglich, durch Drücken der Taste **A** die Variable 1 im Datenregister 1 und mit der Taste **B** die Variable 2 im Datenregister 2 zu speichern. **C** wird wiederum verwendet, das Ergebnis zu erhalten.





IV

Schalten Sie den Rechner in den Learn-Modus und geben Sie die Programmbefehle ein. Drücken Sie **LRN** erneut, um den Learn-Modus aufzuheben, und erproben Sie das Programm. Beachten Sie, daß die Variablen 1 und 2 in beliebiger Reihenfolge eingegeben werden können.

Obwohl die Änderung die Größe des Programms erhöht, ist es jetzt viel einfacher anzuwenden. Die folgende Gegenüberstellung dieser drei Programme vermittelt einen Eindruck, wie die Programmadress-Tasten die Gebrauchseignung eines Programms verbessern können.

Erste Version	Zweite Version	Dritte Version
Eingabe von 227	Eingabe von 227	Eingabe von 227
Tastenfolge STO 01	Tastenfolge STO 01	Taste A
Eingabe von 34	Eingabe von 34	Eingabe von 34
Tastenfolge STO 02	Tastenfolge STO 02	Taste B
Tasten RST R/S	Taste C	Taste C
Anzeige: 261	Anzeige: 261	Anzeige :261

Labels können beliebig in ein Programm in einer Befehlsfolge eingebracht werden, ohne Sinnänderung dieser Folge. Sie werden im Verarbeitungsablauf der Befehle einfach ignoriert, und haben nur den Zweck, einen gewünschten Punkt im Programmspeicher aufzufinden. Sie beeinflussen keine unvollständigen Operationen. Diese Aussage bedeutet nicht, daß ein Label eine Folge wie **RCL** 14 unterbrechen darf, bei der für die Definition einer einzelnen Verarbeitungseinheit mehr als ein Speicherplatz beansprucht wird.

Labels sollten schon von Beginn an in die Planung des Programms einbezogen und nicht erst nachträglich eingefügt werden. Dann tasten Sie Ihre Labels zusammen mit dem restlichen Code wie die anderen Befehle ein.

Natürlich machen Labels die Anwendung eines Programms, das nur zwei Zahlen addiert, nicht zweckmäßiger, denn die Anzahl der Tastenbetätigungen nimmt zu, anstatt sich zu reduzieren. Man muß jedoch auf die Funktion der Labels als wertvolle Programmierhilfen in größeren und komplizierteren Programmen deutlich hinweisen.



Kurzformadressierung

Bisher wurden immer zweistellige Adressen für den Zugriff auf Datenregister verwendet. Das heißt, eine Variable, die zum Beispiel im Datenregister 1 gespeichert ist, wurde mit der Folge **RCL 01** aufgerufen. In einigen Fällen erübrigen sich jedoch die führenden Nullen bei Operationen mit den Datenregistern 0 bis 9. Diese Art der Adressierung ist die sogenannte Kurzform-Adressierung, die immer dann möglich ist, wenn unmittelbar nach der Registeradresse eine nichtnumerische Taste folgt.

Beispiel: Speichern Sie 227 im Datenregister 1 und 34 in Register 2. Rufen Sie dann diese Werte auf und berechnen Sie ihre Summe.

Taste	Anzeige	Bemerkungen
227 STO 01	227.	Da die nächste Eingabe mit einer numerischen Taste erfolgt, muß die volle Adresse angegeben werden.
34 STO 2	34.	Bei den letzten 3 Vorgängen ist Kurzformadressierung möglich, weil nach jeder Adresse eine nichtnumerische Taste folgt.
RCL 1 +	227.	
RCL 2 =	261.	

Beachten Sie, daß bei Kurzformadressierung der Befehl nicht abgeschlossen wird, bis Sie eine nicht-numerische Taste drücken. Das heißt, 227 wird nicht aufgerufen, bis **+** gedrückt wird, und 34 nicht gespeichert, bis die Anweisung **RCL** gegeben wird.



Eintasten Ihres Programms

Programmieren ist die Technik, die Anweisungen an den Rechner zu bestimmen. Sobald Sie diese Befehle vorbereitet haben, müssen Sie auch das Eingabeverfahren beherrschen. Sie kennen bereits den Learn-Modus, der im folgenden eingehend behandelt wird.

Programme werden durch logische Organisation des Problems erstellt. Obwohl Sie zur Entwicklung eines Programms keinen Rechner brauchen, wollen Sie sicher jedes Programm in diesem Buch nachvollziehen. Dieses Kapitel hier soll Sie mit dem Learn-Modus vertraut machen, und Ihnen helfen, die Lücke zwischen der Formulierung eines Programms und dessen Anwendung zu überbrücken.

Der Rechner kann Programmbefehle über die Tastatur nur im Learn-Modus aufnehmen. Umgekehrt wird jeder Tastendruck, der im Learn-Modus erfolgt, vom Rechner als Befehl aufgenommen (eine Ausnahme bilden nur die vier Anweisungen zu Redigier- und Korrekturzwecken, die einige Seiten später erklärt sind). Diese Tatsache ist äußerst wichtig, denn sie bedeutet, daß die Befehle mit Sorgfalt und Konzentration eingegeben werden müssen, und daß Berechnungen über die Tastatur im Learn-Modus nicht durchgeführt werden können.

Wenn Ihnen beim Eintasten eines Befehls ein Fehler unterläuft, müssen Sie nicht neu beginnen und die Eingabe der ganzen Befehlsfolge wiederholen. Der Rechner verfügt über Korrekturtasten, mit denen falsche Eingaben berichtigt, oder Befehle gelöscht und eingefügt werden können. Zu diesen Tasten siehe Kapitel „Redigieren von Programmen“.

Wenn Sie die folgenden einfachen Schritte beachten, können Sie jedes Programm eingeben.

1. Über die Tastatur drücken Sie **2nd** **GP**, um den Programmzeiger auf den Speicherplatz 000 einzustellen, und den gesamten Programmspeicher zu löschen.
2. Drücken Sie **LRN**, um den Rechner in den Learn-Modus zu schalten. (Erklärung des Anzeigeformats siehe „Anzeige des Programms“ auf der nächsten Seite.)
3. Tasten Sie das Programm ein und vergessen Sie dabei nicht die **2nd**-Präfixe.
4. Beachten Sie, daß das Programm nicht die Programmspeicherkapazität überschreitet. Wenn zu viele Befehle eingegeben werden, schaltet der Rechner auf Tastatursteuerung, und das Anzeigeformat des Learn-Modus verschwindet.
5. Drücken Sie **LRN** erneut, um wieder auf Tastatursteuerung zu schalten.
6. Lassen Sie Testprobleme ablaufen und korrigieren oder redigieren Sie Ihr Programm. Siehe hierzu das Kapitel „Redigieren von Programmen“.



Anzeige des Programms

Das Anzeigeformat des Learn-Modus soll die Position des Programmzeigers sowie den augenblicklichen Befehl in diesem Speicherplatz ausweisen. Schalten Sie Ihren Rechner ein und geben Sie mit der Taste **[LRN]** den Learn-Modus ein. Eine Anzeige, bestehend aus zwei Gruppen von Nullen, muß daraufhin erscheinen.



Die Dreiergruppe links zeigt den Speicherplatz, auf den der Programmzeiger im Programmspeicher gerichtet ist. Beim Schreiben eines Programms wird jedem Befehl ein Speicherplatz im Programmspeicher zugewiesen. Damit ist nicht nur eine ständige Kontrolle der Befehle gewährleistet, der Rechner erhält gleichzeitig Anweisungen, in welcher Reihenfolge die Befehle abzuschließen sind.

Da der Rechner nur Zahlen erfassen kann, ist jeder Taste ein zweistelliger Code zugeordnet (Tastencode). Die Zweiergruppe rechts in der Anzeige weist den Tastencode für den Befehl aus, der in dem links angegebenen Programmspeicherplatz gespeichert ist. Die allgemeinen Regeln für die Kodierung der Tasten lauten:

1. Alle numerischen Tasten werden durch die entsprechende Ziffer dargestellt, also wird zum Beispiel die Taste **[7]** als „07“ kodiert.
2. Allen anderen Erstfunktionen sind Tastencodes bezüglich ihrer Lage auf der Tastatur zugewiesen. Die erste Ziffer gibt an, in welcher der neun Reihen (1 bis 9 von oben nach unten) sich die Taste befindet. Die zweite Ziffer ergibt sich aus der Position der Taste in der Spalte (1 bis 5 von links nach rechts). Im Learn-Modus erscheint der Befehl **[√x]** (Reihe 3, Spalte 4), der den Programmspeicherplatz 073 belegt, in der Anzeige wie folgt:



3. Für die Kodierung der Zweitfunktionen addiert man jeweils 5 zur Spaltenposition. **[2nd] [cos]**, oberhalb der Taste **[√x]** (Kode „34“) wird dann als „39“ kodiert. Dagegen erhält **[2nd] [tan]** oberhalb der Taste **[1/x]** (Kode „35“) den Code „30“ und nicht den Code „40“, weil die Nummer der Reihe gleichbleibt.

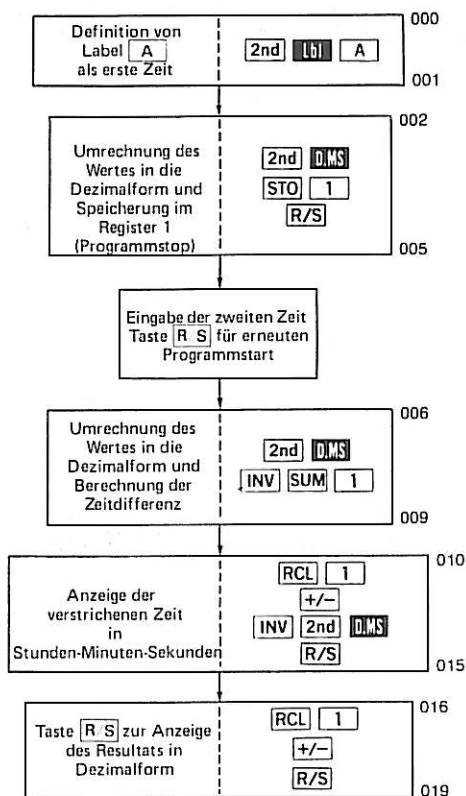
Sie können die mitgelieferte Klarsichtfolie mit den Tastencodes auflegen, um die Codes kennenzulernen. Tastencode-Auflistungen finden Sie auch im Abschnitt V.

In einigen Sonderfällen können die Befehle kombiniert und in einem Speicherplatz gespeichert werden. In diesen Fällen sind auch die Tastencodes kombiniert oder in einem einzelnen 2-stelligen Code zusammengefaßt. **[RCL] 12** belegt zum Beispiel 2 Programmspeicherplätze. **[RCL]** wird im ersten Platz abgespeichert und die Folge **[1] [2]** wird zusammengefaßt und belegt den zweiten Speicherplatz. In diesem Beispiel werden die Tastencodes zu dem 2-stelligen Code „12“ verbunden. Hier wird „12“ nicht als der Programmadressbefehl **[B]** interpretiert, weil jede Speicheroperation den Rechner veranlaßt, den nächsten Befehl in eine Datenregister-Adresse umzusetzen. Andere Sonderfälle werden von Fall zu Fall erläutert.



Zeitdifferenz-Programm

Schreiben Sie ein Programm zur Bestimmung der Zeitdifferenz zwischen zwei angegebenen Zeitpunkten. Sie können die Zeit im Stunden-Minuten-Sekunden-Format eingeben, (d.h., 3:16:03 = 3.1603) und sie für die Berechnung mit **2nd** **DMS** in die Dezimalform umwandeln. Angaben zu dieser Umrechnung siehe „Altgrad-Umformungen“, Abschnitt II.



IV



Führen Sie diese Übung durch und gehen Sie dabei wie folgt vor: Zuerst drückt man **2nd** **CP**, um den Programmspeicher zu löschen und den Programmzeiger auf den Speicherplatz 000 einzustellen. Dann gibt man das Programm nach dem unten angegebenen Verfahren ein.

Taste	Anzeige
LRN	000 00
2nd LbI	001 00
A	002 00
2nd D.MS	003 00
STO	004 00
1	*004 00
R/S	006 00
2nd D.MS	007 00
INV	008 00
SUM	009 00
1	*009 00
RCL	011 00
1	*011 00
+/-	013 00
INV	014 00
2nd D.MS	015 00
R/S	016 00
RCL	017 00
1	*017 00
+/-	019 00
R/S	020 00

Da zu Beginn der Programmspeicher mit **2nd** **CP** gelöscht wurde, werden bei Eingabe des Programms alle Tastenkodes als 00 ausgewiesen. Ursache dafür ist, daß der Programmzeiger sofort nach der Belegung eines Speicherplatzes auf den nächsten Platz vorrückt, und der Tastenkode des hier gespeicherten Befehls angezeigt wird – und nicht der Kode des eben eingegebenen Befehls.

- * Diese Anzeigen scheinen nicht folgerichtig zu sein; beachten Sie jedoch, daß hier die Möglichkeit der Kurzformadressierung genutzt wurde. Tatsächlich erwartet der Rechner den Abschluß der Datenregister-Adresse. Im ersten Fall würde zum Beispiel 005 00 in der Anzeige erscheinen, wenn man **STO** 01 gedrückt hätte. Im obigen Beispiel jedoch wird mit dem nicht-numerischen Befehl **R/S** der Rechner angewiesen, daß die in 004 gespeicherte Adresse vollständig ist und daß der Befehl **R/S** in 005 gespeichert werden muß. Der Zeiger rückt dann automatisch auf 006 vor, sobald der Speicherplatz 005 belegt ist.



IV

Mit nachstehendem Verfahren überprüfen Sie, ob das Programm korrekt eingegeben wurde. Wenn Sie über einen Drucker PC-100A, PC-100B oder PC-100C verfügen, drücken Sie **RST** **2nd** **List**, um eine Programmauflistung zu erhalten. Drücken Sie **R/S** um das Auflisten anzuhalten.

Taste	Anzeige	Entsprechende Befehle
RST LRN	000 76	2nd Lbl
SST	001 11	A
SST	002 88	2nd D.MS
SST	003 42	STO
SST	004 01	1
SST	005 91	R/S
SST	006 88	2nd D.MS
SST	007 22	INV
SST	008 44	SUM
SST	009 01	1
SST	010 43	RCL
SST	011 01	1
SST	012 94	+/-
SST	013 22	INV
SST	014 88	2nd D.MS
SST	015 91	R/S
SST	016 43	RCL
SST	017 01	1
SST	018 94	+/-
SST	019 91	R/S
LRN		

Jetzt, nach fehlerfreier Rechnerprogrammierung lassen Sie das Programm mit folgenden Werten ablaufen: 2:15 für die erste Zeit und 3:42:54 für die zweite Zeit.

Eingabe	Taste	Anzeige	Bemerkungen
2.15	A	2.25	t_1 (H.MMSS) $\rightarrow t_1$ (H.hh)
3.4254	R/S	1.2754	t_2 (H.MMSS) $\rightarrow \Delta t$ (H.MMSS)
	R/S	1.465	$\rightarrow \Delta t$ (H.hh)

Die Zeitdifferenz beträgt eine Stunde, 27 Minuten und 54 Sekunden oder 1.465 Stunden.

Wenn Sie bei der Durchführung dieses Programms Ausgabeinformationen wie etwa 6.396 im Stunden-Minuten-Sekunden-Format erhalten, ist dieses Ergebnis als 6 Stunden, 39 Minuten und 60 Sekunden zu interpretieren, das entspricht 6 Stunden und 40 Minuten.



Redigieren von Programmen

Im Learn-Modus haben Sie folgende Eingriffsmöglichkeiten:

1. Anzeige des gespeicherten Befehls nach Angabe des Speicherplatzes.
2. Ersetzen eines Befehls durch einen anderen.
3. Löschen eines Befehls und Schließen der Lücke.
4. Raum schaffen für einen zusätzlichen Befehl, ohne die zuvor programmierten Befehle zu zerstören.
5. Durchsicht des Programmspeichers in Einzelschritten vor oder zurück ohne Einwirkung auf dessen Inhalt.

Diese Eigenschaften erlauben Überprüfung, Korrektur und Änderung eines Programms, ohne richtige Befehle erneut eingeben zu müssen.

Die vier Tasten, die man im Learn-Modus drücken kann, und die dabei nicht als Programmbefehle interpretiert werden, sind: **SST**, **BST**, **Ins** und **Del**. **SST** und **BST** erlauben die Durchsicht des Programmspeichers in Einzelschritten vor oder zurück und die Überprüfung des Inhaltes der einzelnen Speicherplätze. Im Rechenmodus kann mit **SST** ein Programm schrittweise ausgeführt werden, eine Möglichkeit, die Ergebnisse jeder Operation zu beobachten. Der Befehl **2nd Ins** bewirkt, daß der augenblickliche Befehl sowie alle folgenden um einen Platz im Programmspeicher nach hinten rücken und im augenblicklichen Speicherplatz wird ein Nullbefehl eingefügt. Wenn auch der letzte Programmspeicherplatz mit einem Befehl belegt ist, geht dieser infolge der **Ins**-Anweisung verloren. Mit der Folge **2nd Del** löscht man den Befehl im derzeitigen Programmspeicherplatz und verschiebt alle folgenden Befehle einen Speicherplatz nach vorn. Der letzte Speicherplatz wird dann mit einer Null belegt.

Zwei weitere zweckdienliche Redigiertasten sind **RST** und **GTO** (Vorsicht: **RST** hat mehrere Funktionen. Siehe „Grundfunktionen für die Programmsteuerung“ in Abschnitt V). Gibt man **RST** manuell über die Tastatur ein, wird der Programmzeiger auf 000 eingestellt. **GTO**, gefolgt von einer dreistelligen absoluten Programmadresse oder einem Label, richtet den Programmzeiger auf diesen Programmspeicherplatz. (Führende Nullen können bei Kurzformadressierung unterdrückt werden.) Drückt man **GTO** und läßt eine Label-adresse folgen, wird der Programmzeiger auf den ersten Speicherplatz nach dem Label eingestellt. Wenn man nach einer der oben erwähnten Folgen den Learn-Modus eingibt, können Sie den Inhalt des Programmspeichers am gewünschten Speicherplatz prüfen. **Achtung!** Wenn Sie diese Tasten drücken, während der Rechner in den Learn-Modus geschaltet ist, werden sie als Programmbefehle interpretiert.

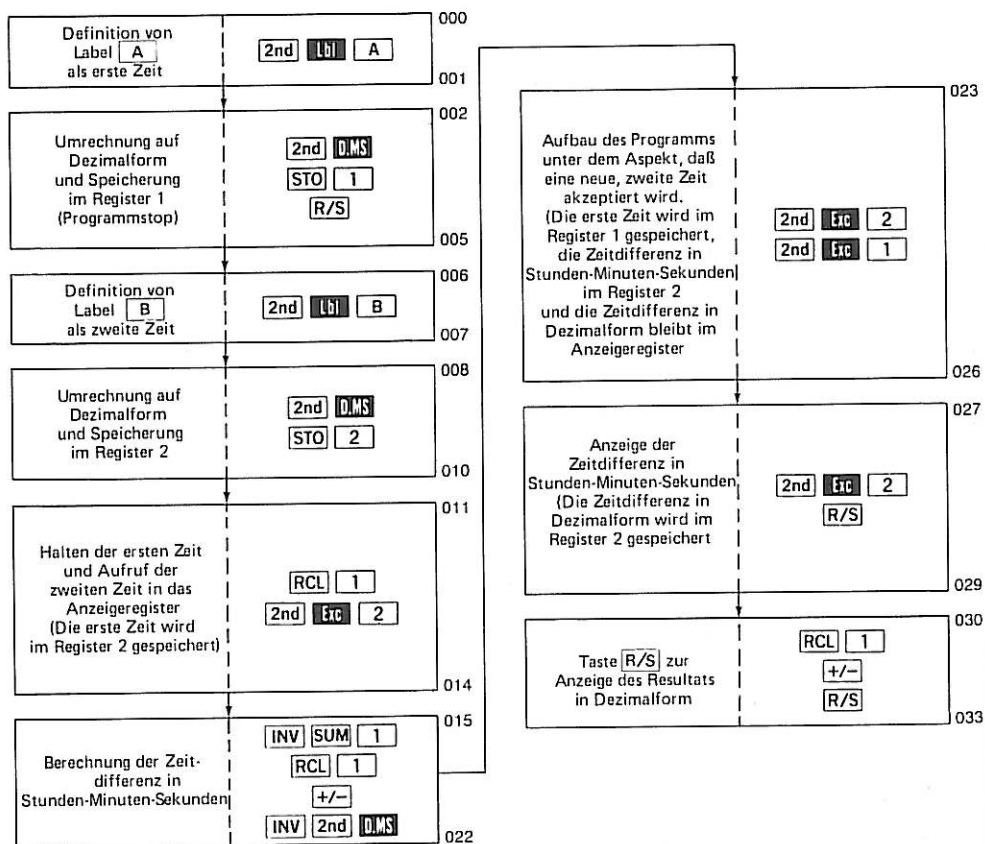
Wenn Sie eine Programmfolge ändern wollen, suchen Sie die Folge mit einer der oben beschriebenen Methoden auf, und überschreiben einfach die alten Befehle mit den neuen, oder Sie addieren und löschen Befehle nach Bedarf.



Verbesserung des Zeitdifferenz-Programms

Das letzte Programm soll so aufbereitet werden, daß man die zweite Zeit ohne Eingabewiederholung der ersten Zeit ändern kann. Nehmen Sie die Modifikationen nach Eingabe des Originalprogramms vor und tasten Sie nicht wieder das ganze Programm ein.

Drei Dinge sind zu erledigen: Erstens verwendet man für die Eingabe der zweiten Zeit ein Label, um den direkten Zugriff auf dieses Programmsegment zu ermöglichen. Zweitens braucht man ein Mittel, um die erste Zeit so zu erhalten, daß die Information nach der Berechnung wiedergewonnen werden kann. Und drittens wird das Programm so aufgebaut, daß nach der Berechnung der Zeitdifferenz die neue zweite Zeit akzeptiert wird.



In der oben angegebenen Tastenfolge ist ein Fehler unterlaufen. Versuchen Sie, den Fehler zu finden, ehe Sie weiterlesen.

IV



Führen Sie die notwendigen Änderungen nach folgendem Verfahren durch.

Taste	Anzeige	Bemerkungen
GTO 6		Einstellung des Programmzeigers auf Speicherplatz 006
LRN	006 88	Eingabe des Learn-Modus
2nd Ins	006 00	Einfügen von Label B
2nd Ins	006 00	
2nd Lbl	007 00	
B	008 88	
SST	009 22	Vorrücken des Programmzeigers um einen Schritt
2nd Ins	009 00	Einfügen der Befehle 009 - 014
2nd Ins	009 00	
STO 2	010 00	
2nd Ins	011 00	
2nd Ins	011 00	
RCL 1	012 00	
2nd Ins	013 00	
2nd Ins	013 00	
2nd Exc 2	014 00	
SST SST SST	018 43	Vorrücken des Programmzeigers um 8 Schritte
SST SST SST	021 22	
SST SST	023 91	
2nd Ins	023 00	Einfügen der Befehle 023 - 028
2nd Ins	023 00	
2nd Exc 2	024 00	
2nd Ins	025 00	
2nd Ins	025 00	
2nd Exc 1	026 00	
2nd Ins	027 00	
2nd Ins	027 00	
2nd Exc 2	028 00	
LRN		Aufheben des Learn-Modus



IV

Sie können mit folgender Methode überprüfen, ob das Programm korrekt geändert wurde.

Taste	Anzeige	Entsprechende Befehle
RST LRN	000 76	2nd LbI
SST	001 11	A
SST	002 88	2nd D.MS
SST	003 42	STO
SST	004 01	1
SST	005 91	R/S
SST	006 76	2nd LbI
SST	007 12	B
SST	008 88	2nd D.MS
SST	009 42	STO
SST	010 02	2
SST	011 43	RCL
SST	012 01	1
SST	013 48	2nd Exc
SST	014 02	2
SST	015 22	INV
SST	016 44	SUM
SST	017 01	1
SST	018 43	RCL
SST	019 01	1
SST	020 94	+/-
SST	021 22	INV
SST	022 88	2nd D.MS
SST	023 48	2nd Exc
SST	024 02	2
SST	025 48	2nd Exc
SST	026 01	1
SST	027 48	2nd Exc
SST	028 02	2
SST	029 91	R/S
SST	030 43	RCL
SST	031 01	1
SST	032 94	+/-
SST	033 91	R/S

IV



Führen Sie das Programm mit folgenden Werten durch: 1:30 für die erste Zeit und 2:13:57 sowie 2:14:24 für die zweite Zeit.

Eingabe	Taste	Anzeige	Bemerkungen
1.3	A	1.5	t_1 (HH.MMSS) $\rightarrow t_1$ (HH.hh)
2.1357	B	0.4357	t_2 (HH.MMSS) $\rightarrow \Delta t$ (HH.MMSS)
	R/S	-1.5	?

Das Beispiel wird hier unterbrochen, weil das Ergebnis offenkundig falsch ist. Die Ausgabeinformation muß die Zeitdifferenz in Dezimalstunden sein; hier wird jedoch der negative Wert der ersten Zeit in seiner Dezimalform angezeigt. Bei einer Überprüfung des Ablaufdiagramms und der entsprechenden Tastenbefehle wird ersichtlich, daß die gewünschte Ausgabeinformation nicht verloren ist. Die Austauschfolgen der Schritte 023 bis 028 haben die Information nur in das Datenregister 2 übertragen. Deshalb läßt sich das Problem beheben, wenn man Schritt 31 ändert. Die Korrektur wird einfach durch Ersetzen des betreffenden Befehls vorgenommen.

Taste	Anzeige
GTO 3 1	
LRN	031 01
2	032 94
LRN	

Jetzt lassen Sie das Programm erneut ablaufen.

Eingabe	Taste	Anzeige	Bemerkungen
1.3	A	1.5	t_1 (HH.MMSS) $\rightarrow t_1$ (HH.hh)
2.1357	B	0.4357	t_2 (HH.MMSS) $\rightarrow \Delta t$ (HH.MMSS)
	R/S	0.7325	$\rightarrow \Delta t$ (HH.hh)
2.1424	B	0.4424	t_2 (HH.MMSS) $\rightarrow \Delta t$ (HH.MMSS)
	R/S	0.74	$\rightarrow \Delta t$ (HH.hh)



Redigieren mit kombinierten Codes

Nehmen Sie an, die Ausgabeinformation im obigen Beispiel wäre in Register 12 und nicht in 2 gespeichert worden. In diesem Fall müßten Sie im Programmspeicherplatz 031 die Zahl 12 speichern. Hier müssen Sie besonders sorgfältig darauf achten, daß dieser Code richtig kombiniert wird. (Siehe „Anzeige des Programms“ einige Seiten zuvor). Nachstehend eine Methode zur Eingabe dieses Codes:

Taste	Anzeige
CLR GTO 30	0
LRN	030 43
RCL	031 01
1 2	032 94
LRN	0

Dieses Verfahren erfordert die erneute Eingabe des **RCL**-Befehls. Der Rechner wird damit angewiesen, die Datenregisteradresse automatisch zusammenzufassen und in einem einzigen Programmspeicherplatz zu speichern.



Typische Anwendungsbeispiele für Programmierung

PROGRAMMIEREN IST INDIVIDUELL

Ehe diese Ausführungen fortgesetzt werden, ist es wichtig, Programmieren auch als eine rein persönliche Angelegenheit zu begreifen. Das heißt, daß zwei Personen, die ein- und dasselbe Problem programmieren, mit unterschiedlichen Programmbefehlen zum gleichen Ergebnis kommen können. Auch in den verschiedenen Arten, wie ein Problem gelöst wird, sind individuelle Unterschiede erkennbar. Organisatorische Abläufe können als Folge unterschiedlicher Ausbildung und beruflicher Laufbahnen differieren. Ein Ingenieur mit umfassendem mathematischem Training wird wahrscheinlich eine Lösung mit komplexen mathematischen Gleichungen benötigen, während ein Fachmann der Geisteswissenschaften mit wenig Erfahrung in Mathematik seine Aufgaben auf andere Weise mit reiner Arithmetik lösen kann. Eine Person mag mit einer Befehlsfolge zufrieden sein, die einen Großteil des Programmspeichers belegt, während andere immer nach Möglichkeiten suchen, das Programm so zu komprimieren, daß es nur das Minimum der Speicherkapazität beansprucht. Jeder will schließlich die ihm vertraute Lösungsmethode wählen.

Ihr Stil entwickelt sich mit dem Verständnis für die Programmiervorgänge. Sie sollten auch diese Lernperiode als erfreuliches Erlebnis empfinden. Seien Sie unbesorgt, wenn Ihnen während des Lernprozesses Fehler unterlaufen – Ihr Rechner nimmt nichts übel. Eine Betriebsstörung bei einem großen Computer kann sehr kostenintensiv sein, deshalb werden angehende Programmierer oft ferngehalten. Bei Ihrem Rechner entstehen jedoch keinerlei Betriebskosten – nutzen Sie also diese gute Gelegenheit und experimentieren Sie mit verschiedenen Lösungswegen, Funktionen, Mustern und mit Ihren sonstigen Ideen.

ZINSESZINSPROGRAMM

Welche Vorteile bieten programmierbare Rechner? Der programmierbare Rechner soll Lösungen schneller ermitteln, und Fehlermöglichkeiten durch ständige Eingabewiederholungen reduzieren. Die Programmierung der folgenden Aufgabe verdeutlicht den zeitsparenden Aspekt.

Jeder hatte irgendwann schon ein Sparkonto, und auf die Einlage wurden Zinsen aufgeschlagen. Bei einem Betrag von 1.000 DM mit 5% jährlicher Verzinsung werden am Ende des Jahres 50 DM an Zinsen zugeschlagen und das Konto steigt auf 1.050 DM. Die Einlage von 1.000 DM ist das Anfangskapital oder der Barwert, da dieser Betrag noch keine Zinsen beinhaltet. Der Wert am Ende des Jahres, der auf 1.050 DM angewachsen ist, wird als künftiger Wert oder als Endkapital bezeichnet. Zinseszinsrechnung bedeutet, daß ein bestimmter Geldbetrag auf ein Konto eingezahlt wird und dort eine oder mehrere Perioden ruht. Die Zinsen werden jeweils dem Kapital zu Anfang der Periode zugeschlagen. Auf diese Weise werden in der nächsten Periode auch die kapitalisierten Zinsen mitverzinst. Die ursprüngliche Einlage entwickelt sich wie folgt:

$$1.000 \text{ DM} + 1.000 \text{ DM} (.05) = 1.050 \text{ DM am Ende des ersten Jahres}$$

$$1.050 \text{ DM} + 1.050 \text{ DM} (.05) = 1.102.50 \text{ DM am Ende des zweiten Jahres}$$



Die Formel für diesen Vorgang ist allgemein bekannt. Sie kann wie folgt ausgedrückt werden:

„Das Endkapital (der künftige Wert des Betrages) ist gleich dem Anfangskapital, das mit dem Wert multipliziert wird, der sich aus eins plus dem Zinssatz, potenziert mit der Anzahl der Verzinsungsperioden ergibt.“

Mathematisch:

$$FV = PV \times (1 + i)^n$$

Ehe Sie das Programm schreiben, müssen Sie logisch planen, was durchgeführt werden muß. Erstens muß der Zinssatz in Dezimalform in die Gleichung eingegeben werden. Überlassen Sie dem Rechner diese Aufgabe, wobei der Zinssatz nach Eingabe durch 100 dividiert wird. Dazu kommt, daß Sparkassen verschiedene Verzinsungsperioden ansetzen (vierteljährlich, täglich etc.) Das Programm gewinnt an Flexibilität, wenn Sie eine Möglichkeit finden, daß der Rechner die Art der Verzinsung erkennen kann. Bei Einsetzen dieser Änderungen kann die Gleichung zur Berechnung des künftigen Wertes wie folgt umgeschrieben werden:

$$FV = PV \times \left[1 + \left(\frac{i}{100} \div c \right) \right]^{cn}$$

Die oben verwendeten Variablen sind:

FV = künftiger Wert der Einlage

PV = Barwert der Einlage

i = jährlicher Zinssatz

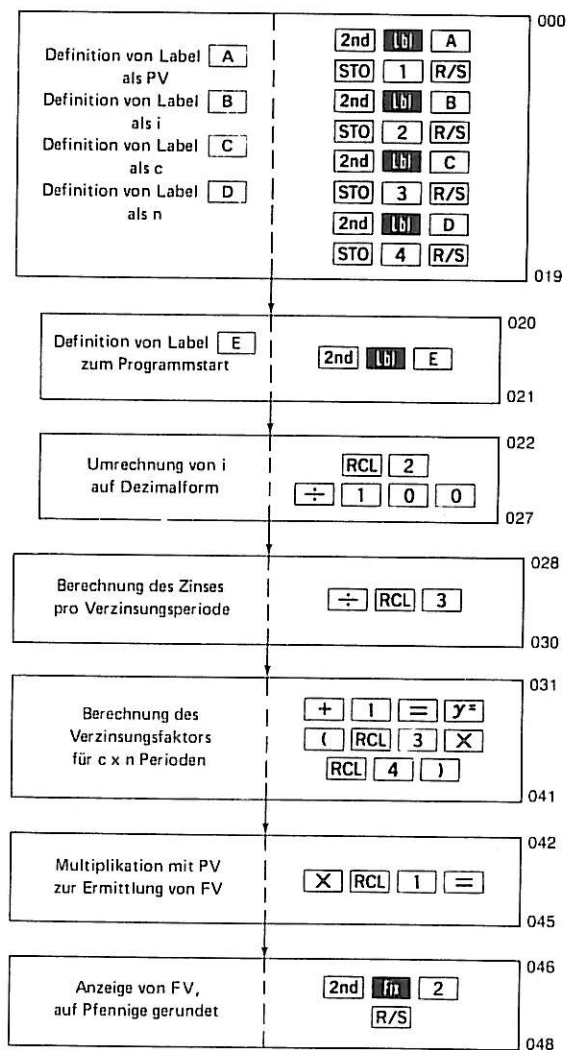
c = Anzahl der Verzinsungsperioden pro Jahr

n = Anzahl der Jahre (gesamte Verzinsungszeit)

ANMERKUNG: In Europa wird bei der Zinseszinsberechnung im allgemeinen der effektive Jahreszins verwendet. Die effektiven Jahreszinsen sind nicht einfache Vielfache der periodischen Zinssätze, sondern werden selbst aus der Zinseszinsgleichung hergeleitet. Der Einfachheit halber wurde hier auf das europäische Beispiel verzichtet.

Jetzt müssen Sie die Entscheidung treffen, ob Sie die Variablen in die Anzeige eingeben, sobald sie benötigt werden, oder ob Sie diese Werte speichern, damit sie bei Bedarf aufgerufen werden können. In diesem Beispiel sind sie gespeichert. Damit kann man die Variablen einzeln eingeben und verschiedene Möglichkeiten leichter berechnen. Wenn der Programmablauf mit früher eingegebenen Daten wiederholt werden soll, müssen Sie dafür sorgen, daß die ursprünglichen Daten erhalten bleiben. Eine derartige Änderung war auch im Zeitdifferenzprogramm erforderlich.

Die Lösungsmethode liegt nun fest, die Struktur der Gleichung zeigt die gewünschten Variablen auf, und die Verarbeitungsweise für die Variablen ist bereits bestimmt. Jetzt können Sie den Programmablaufplan erstellen und das Programm schreiben.



Programm für die Berechnung einer Investition



PROGRAMM-INSTRUKTIONEN				
Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Löschen des Programmspeichers und Rückstellen des Programmzeigers		2nd CP	
2	Eingabe des Learn-Modus		LRN	000 00
3	Eingabe des Zinseszinsprogramms			
4	Aufheben des Learn-Modus		LRN	0
5	Eingabe des Barwertes	PV	A	PV
6	Eingabe des jährlichen Zinssatzes	i	B	i
7	Eingabe der Verzinsungsperioden pro Jahr	c	C	c
8	Eingabe der Anzahl der Jahre	n	D	n
9	Berechnung des Endkapitals Die Eingabereihenfolge für die Variablen ist beliebig — Für neue Aufgaben müssen Variable, die un- verändert bleiben, nicht wiederholt eingegeben werden.		E	FV



Speicherplatz und Tastenkod

000 76	2nd Lbl
001 11	A
002 42	STO
003 01	1
004 91	R/S
005 76	2nd Lbl
006 12	B
007 42	STO
008 02	2
009 91	R/S
010 76	2nd Lbl
011 13	C
012 42	STO
013 03	3
014 91	R/S
015 76	2nd Lbl
016 14	D
017 42	STO
018 04	4
019 91	R/S
020 76	2nd Lbl
021 15	E
022 43	RCL
023 02	2
024 55	÷

Speicherplatz und Tastenkod

025 01	1
026 00	0
027 00	0
028 55	÷
029 43	RCL
030 03	3
031 85	+
032 01	1
033 95	=
034 45	y^x
035 53	(
036 43	RCL
037 03	3
038 65	X
039 43	RCL
040 04	4
041 54)
042 65	X
043 43	RCL
044 01	1
045 95	=
046 58	2nd fix
047 02	2
048 91	R/S

Programm für die Berechnung einer Investition



Berechnen Sie den künftigen Wert einer Einlage von 3.000 DM nach 5 Jahren, wenn der Jahreszinssatz 8% beträgt und täglich, bzw. monatlich verzinst wird.

Eingabe	Taste	Anzeige	Bemerkungen
3000	A	3000.	PV
8	B	8.	i
365	C	365.	c
5	D	5.	n
	E	4475.28	FV
12	C	12.00	c
	E	4469.54	FV

Alle Resultate von hier an werden mit zwei Dezimalstellen ausgewiesen. Zusätzlich bleibt die Einstellung auf zwei Dezimalstellen gespeichert, bis sie wieder geändert wird, selbst wenn Sie den Rechner zwischenzeitlich ausschalten (gilt für TI-58C).

PREISBERECHNUNGS-PROGRAMM

Bisher wurden die Datenregister primär für Speicherung und Aufruf der Variablen verwendet. Der Rechner kann jedoch auch die zuvor in den Registern gespeicherten Variablen addieren, subtrahieren, multiplizieren und dividieren, ohne sie aufzurufen. Die Anwendung des Speichers auf diese Weise wird oft als Speicherarithmetik bezeichnet. Der Vorteil dieser Speicherarithmetik ist am folgenden Musterprogramm demonstriert. Beachten Sie auch, daß mit einfacher Arithmetik ein äußerst zweckmäßiges Programm erstellt werden kann. Ferner ist nachdrücklich zu betonen, daß sich programmierbare Rechner für jede Art von Programm vorzüglich eignen – nicht nur für solche mit komplexen mathematischen Problemen.

Angenommen der normale Auftrag für ein Unternehmen setzt sich aus verschiedenen Artikeln mit unterschiedlichen Verkaufspreisen zusammen. Um die Rechnung an den Kunden zu stellen, multipliziert man die Menge jedes Artikels mit dem Stückpreis, und erhält den Preis für die Bestellmenge des betreffenden Artikels. Zur Berechnung des durchschnittlichen Stückpreises für jede Bestellung, addiert man die einzelnen Bestellmengen und dividiert den gesamten Rechnungsbetrag durch diese Summe. Dieser Vorgang ist nicht schwierig, aber zeitraubend.

Artikel	Bestellmenge	Stückpreis	Gesamtpreis
1	100	DM 0.25	DM 25.00
2	200	0.15	30.00
3	50	0.35	17.50
4	150	0.40	60.00
5	300	0.10	30.00
Gesamte Bestellmenge	800		DM162.50
Mittlerer Stückpreis der Bestellung		DM 0.203125	



Ein Blick auf die Bestellung zeigt, daß Multiplikationen, Additionen und Divisionen durchzuführen sind. Wesentlich ist, wie das Problem organisiert wird und welche Anweisungen der Rechner erhält.

Wenn das Programm eine unbegrenzte Anzahl von Artikeln verarbeiten soll, wählen Sie folgenden Lösungsweg. Zunächst entscheiden Sie, wie die Variablen eingegeben werden. In diesem Beispiel erfolgt die Eingabe der Variablen über die Anzeige, während das Programm abläuft. Sie werden also nicht in den Datenregistern gespeichert und an späterer Stelle im Programm aufgerufen. Wenn Sie mit der Speicherarithmetik die kumulativen Gesamtsummen berechnen, wird jede Datengruppe nur einmal verwendet und muß nicht dauernd in den Datenregistern gespeichert werden. Um Zeitverluste durch Anzeige der Zwischendaten auszugleichen, speichert man die kumulative Bestellmenge in R_1 (Datenregister 1 wird mit R_1 angegeben, Datenregister 2 mit R_2 etc.), den kumulativen Gesamtpreis in R_2 und den jeweiligen durchschnittlichen Stückpreis in R_3 . Das Musterprogramm ist so geschrieben, daß der Gesamtpreis eines Artikels nach Eingabe der Menge und des Stückpreises ausgewiesen wird; andere Ergebnisse können jedoch beliebig aufgerufen werden.

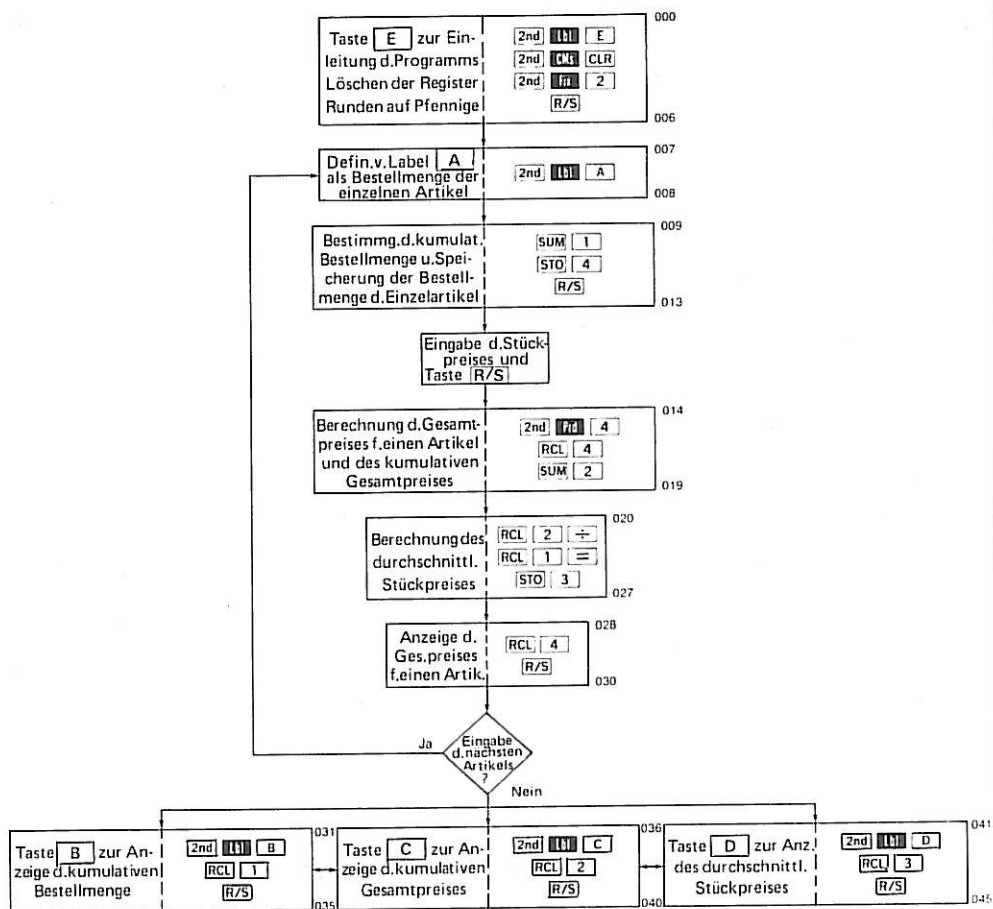
Schließlich ist noch anzumerken, daß das Programm einen einleitenden Programmteil erhalten soll, der diese Datenregister auf Null stellt, weil die Anfangsoperationen für R_1 und R_2 Summierungsbefehle sein müssen.

In diesem Beispiel wird die Wichtigkeit der Organisation des Lösungswegs sehr deutlich. Mit dem aufgezeigten Lösungsweg kann jede Bestellung mit unbegrenzter Anzahl von Einzelposten bearbeitet werden.



IV

Jetzt skizziert man das Ablaufdiagramm und bestimmt die Tastenbefehle für das Programm.



Preiskontroll-Programm



PROGRAMM-INSTRUKTIONEN

Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Löschen des Programmspeichers und Rückstellen des Programmzeigers		2nd CP	
2	Eingabe des Learn-Modus		LRN	000 00
3	Eingabe des Preisberechnungsprogr.			
4	Aufheben des Learn-Modus		LRN	0
5	Programmbeginn		E	00.00
6	Eingabe d. Bestellmenge d. Einzelartik.	Menge	A	Menge
7	Eingabe des Stückpreises	Stückpreis	R/S	Gesamtpreis des bestellten Artikels
	Die Schritte 6 und 7 werden für jeden Artikel wiederholt			
	Jeweils nach Eingabe eines Artikels können folgende Variable angezeigt werden:			
	kumulative Bestellmenge		B	Bestellmenge
	kumulativer Preis		C	Preis
	durchschnittlicher Stückpreis		D	durchschnittlicher Preis



IV

Speicherplatz und Tastenkod

000 76	2nd Lbl
001 15	E
002 47	2nd CMs
003 25	CLR
004 58	2nd fix
005 02	2
006 91	R/S
007 76	2nd Lbl
008 11	A
009 44	SUM
010 01	1
011 42	STO
012 04	4
013 91	R/S
014 49	2nd Prd
015 04	4
016 43	RCL
017 04	4
018 44	SUM
019 02	2
020 43	RCL
021 02	2
022 55	÷
023 43	RCL

Speicherplatz und Tastenkod

024 01	1
025 95	=
026 42	STO
027 03	3
028 43	RCL
029 04	4
030 91	R/S
031 76	2nd Lbl
032 12	B
033 43	RCL
034 01	1
035 91	R/S
036 76	2nd Lbl
037 13	C
038 43	RCL
039 02	2
040 91	R/S
041 76	2nd Lbl
042 14	D
043 43	RCL
044 03	3
045 91	R/S

Preiskontroll-Programm



Durchführung des Programms mit den zu Beginn angegebenen Daten:

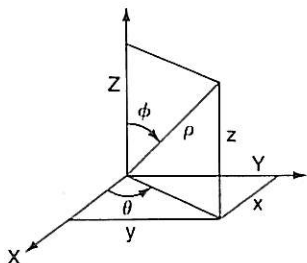
Eingabe	Taste	Anzeige	Bemerkungen
	E	0.00	Einleitung
100	A	100.00	Bestellmenge A
.25	R/S	25.00	Stückpreis A
			Gesamtpreis des bestellten Artikels
200	A	200.00	Bestellmenge B
.15	R/S	30.00	Stückpreis B
			Gesamtpreis des bestellten Artikels
50	A	50.00	Bestellmenge C
.35	R/S	17.50	Stückpreis C
			Gesamtpreis des bestellten Artikels
150	A	150.00	Bestellmenge D
.4	R/S	60.00	Stückpreis D
			Gesamtpreis des bestellten Artikels
300	A	300.00	Bestellmenge E
.1	R/S	30.00	Stückpreis E
			Gesamtpreis des bestellten Artikels
	B	800.00	kumulative Bestellmenge
	C	162.50	kumulativer Gesamtpreis
	D	0.20	durchschnittlicher Stückpreis (gerundet)
	INV 2nd fix	0.203125	durchschnittlicher Stückpreis (ungerundet)



PROGRAMM ZUR UMRECHNUNG VON KUGELKOORDINATEN

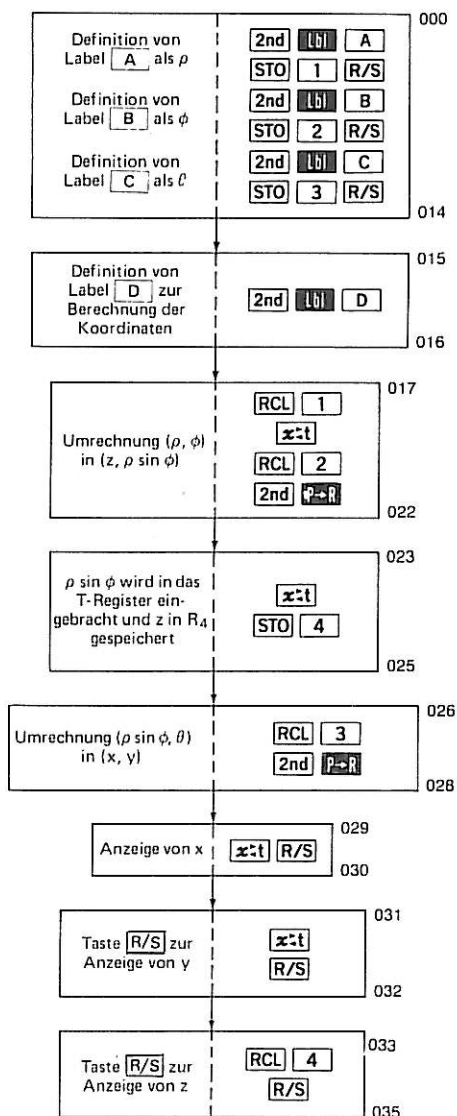
Schreiben Sie ein Programm zur Umrechnung von Kugelkoordinaten in rechtwinklige Koordinaten.

$$\begin{aligned}x &= \rho \sin \phi \cos \theta \\y &= \rho \sin \phi \sin \theta \\z &= \rho \cos \phi\end{aligned}$$



Der Rechner hat eine vorprogrammierte Funktion, $\boxed{2nd} \boxed{P \rightarrow R}$, zur Umrechnung von Polar- in rechtwinklige Koordinaten. (Siehe Abschnitt V.) Diese Funktion ist auch hier von Vorteil.

Die einfachste Möglichkeit, sphärische Koordinaten einzugeben, ist die Speicherung von ρ , ϕ , und θ in den Datenregistern R_1 , R_2 bzw. R_3 . Wenn man den Rechner anweist, ρ in das T-Register und ϕ in das Anzeigeregister einzubringen, kann man z durch die Umformung in rechtwinklige Koordinaten mit der Tastenfolge $\boxed{2nd} \boxed{P \rightarrow R}$ ermitteln. Diese Umrechnung bringt $\rho \sin \phi$ in das Anzeigeregister und $\rho \cos \phi$ ($= z$) in das T-Register. Speichert man z sicher in R_4 und bringt $\rho \sin \phi$ in das T-Register, kann man diese Umrechnung wieder anwenden, um nach Aufruf von θ in das Anzeigeregister x und y zu bestimmen. Das Programm soll so formuliert sein, daß x , y und z in der angegebenen Reihenfolge über die Taste $\boxed{R/S}$ ausgewiesen werden.





IV

PROGRAMM-INSTRUKTIONEN

Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Löschen des Programmspeichers und Rückstellen des Programmzeigers		2nd CP	
2	Eingabe des Learn-Modus		LRN	000 00
3	Programmeingabe			
4	Aufheben des Learn-Modus		LRN	
5	Eingabe ρ	ρ	A	ρ
6	Eingabe ϕ	ϕ	B	ϕ
7	Eingabe θ	θ	C	θ
8A	Berechnung der Koordinaten und Anzeige von x		D	x
8B	Anzeige von y		R/S	y
8C	Anzeige von z		R/S	z

IV



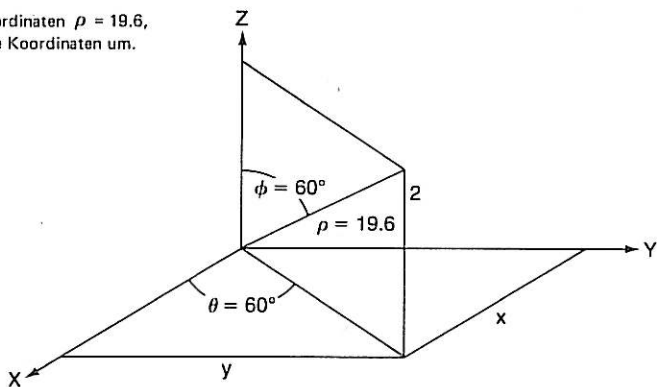
Speicherplatz und Tastenkod	Tastenfolge	Speicherplatz und Tastenkod	Tastenfolge
000 76	2nd Lbl	019 32	x↔t
001 11	A	020 43	RCL
002 42	STO	021 02	2
003 01	1	022 37	2nd P→R
004 91	R/S	023 32	x↔t
005 76	2nd Lbl	024 42	STO
006 12	B	025 04	4
007 42	STO	026 43	RCL
008 02	2	027 03	3
009 91	R/S	028 37	2nd P→R
010 76	2nd Lbl	029 32	x↔t
011 13	C	030 91	R/S
012 42	STO	031 32	x↔t
013 03	3	032 91	R/S
014 91	R/S	033 43	RCL
015 76	2nd Lbl	034 04	4
016 14	D	035 91	R/S
017 43	RCL		
018 01	1		

Programm für Sphärische Koordinaten



IV

Beispiel: Rechnen Sie die Kugelkoordinaten $\rho = 19.6$, $\phi = 60^\circ$, $\theta = 60^\circ$ in rechtwinklige Koordinaten um.



Eingabe	Taste	Anzeige	Bemerkungen
	2nd Deg		Winkelmodus: Altgrad
19.6	A	19.6	ρ
60	B	60	ϕ
60	C	60	θ
	D	8.487048957	x
	R/S	14.7	y
	R/S	9.8	z



FORTGESCHRITTENES PROGRAMMIEREN

Mehr über Labels

Wie Sie bereits wissen, werden die Programmadress-Tasten (A bis E und A' bis E') als Labels definiert und verwendet. Sobald ein Programmsegment mit einer dieser Tasten gekennzeichnet ist, stellt sich der Programmzeiger auf diesen Programmteil ein und der Ablauf wird sofort aufgenommen, wenn eine dieser Tasten manuell gedrückt wird.

Zusätzlich zu diesen Programmadress-Tasten können Sie fast alle Erst- und Zweitfunktionen des Rechners als Labels definieren, **2nd** **sin**, **x²**, **=**, **CLR**, **2nd** **Pgm**, **EE**, **2nd** **fix** und andere können Labels sein, um nur einige Beispiele zu nennen. Diese Labels werden als allgemeine Labels bezeichnet. Die folgenden Tasten bilden eine Ausnahme, sie können nicht als Labels verwendet werden.

Lbl	2nd	SST
Ins	LRN	BST
Del	Ind	Ziffern (0 - 9)

Der Unterschied zwischen den allgemeinen Labels und den Programmadress-Labels besteht darin, daß das allgemeine Label den Programmlauf nicht starten kann, wenn man die als Label definierte Anweisung manuell über die Tastatur eingibt. Auch wenn ein Programmsegment mit **x²** gekennzeichnet ist, wird nur der Anzeigewert quadriert, wenn die Taste **x²** manuell gedrückt wird. Dagegen veranlaßt die Tastenfolge **GTO** **x²** **R/S**, daß der Programmlauf bei Label **x²** beginnt. Dennoch stehen Ihnen jetzt mehr als 60 Labels zur Verfügung.

Ebenso wie die Programmadress-Labels können Sie die allgemeinen Labels an beliebiger Stelle im Programm einsetzen. Offensichtlich zusammengehörige Befehlsgruppen wie **STO** 12 oder **2nd** **fix** 6 oder **INV** **2nd** **sin** dürfen dabei aber nicht getrennt werden. Im restlichen Teil dieses Abschnittes finden Sie wegen der Vielseitigkeit in der Hauptsache Erläuterungen zu den Programmadress-Tasten als Labels.

Verzweigungsbefehle

Einige weitere wesentliche Befehle erhöhen die Programmierkapazität Ihres Rechners. Sie ermöglichen zusätzliche Flexibilität bei der Steuerung der Verarbeitungsreihenfolge der Befehle. Diese neuen Programmsteuerfunktionen sind die Verzweigungsbefehle oder einfach Verzweigungen. Sie können den normalen Hauptfluß eines Programms durch den Sprung zu einem anderen Speicherplatz umlenken. Man unterscheidet zwischen zwei Verzweigungsarten, den unbedingten und bedingten Verzweigungen.

Die unbedingten Verzweigungsbefehle (Sprünge) verzweigen ohne jede Vorbedingung, wo es verlangt wird. Sie sind unabhängig von allen Berechnungen. Ein bedingter Verzweigungsbefehl hingegen testet zunächst einen Wert, und verzweigt nur dann zu einem Speicherplatz außer der Reihe, wenn dieser Wert die Testbedingungen erfüllt.



Unbedingte Verzweigungen

RST, **GTO** und **SBR** sind die unbedingten Verzweigungsbefehle. **RST** stellt den Programmzeiger automatisch auf den Speicherplatz 000 ein. **GTO** und **SBR** richten den Zeiger ohne weitere Bedingung auf den angegebenen Speicherplatz. Beachten Sie, daß **RST** noch andere Funktionen hat. (Siehe Grundfunktionen für die Programmsteuerung auf Seite V-43.)

DER GO-TO-BEFEHL – **GTO**

Im Kapitel „Redigieren von Programmen“ wurde die Funktion des Go-to-Befehls erläutert, wenn er manuell über die Tastatur gegeben wird. Genauso wirkt diese Anweisung während des Programmablaufs. **GTO**, gefolgt von einer absoluten Adresse oder einem Label, richtet den Programmzeiger unmittelbar auf den angegebenen Speicherplatz. Die Verarbeitung wird mit dem neuen Speicherplatz fortgesetzt.

Kurzformadressierung bei absoluten Adressen (Programmspeicheradressen) ist möglich. Im Learn-Modus ist **GTO** 9 dann und nur dann gleichbedeutend mit **GTO** 009, wenn der nächste Tastenbefehl keine Ziffer ist. Geben Sie das folgende kleine Zählprogramm ein:

Taste	Anzeige	Bemerkungen
CLR 2nd CP	0.	Vorbereitung für ein Programm
GTO 9 LRN	009 00	Sprung zum Speicherplatz 009 und Eingabe des Learn-Modus
+	010 00	Programmeingabe
1	011 00	
=	012 00	
2nd Pause	013 00	
GTO	014 00	
9	014 00	Die Speicherplatznummer rückte nicht weiter, weil der Rest der Adresse erwartet wird
LRN	0.	Aufheben des Learn-Modus
LRN	016 00	Der Programmzeiger rückte weiter, als mit dem ersten LRN der Abschluß der eingehenden Adresse signalisiert wurde.

Um dieses Programm durchzuführen, drückt man **GTO** 9 **R S**.

Im Programmspeicher belegen die absoluten Adressen 2 Programmspeicherplätze. Für die Folge **GTO** 136 benötigt man also 3 Programmspeicherplätze. Der erste Speicherplatz enthält den **GTO**-Befehl, die Adresse wird in den beiden nächsten gespeichert. Davon wird die Hunderterstelle der Adresse in den ersten dieser Speicherplätze eingebracht, und die letzten zwei Stellen werden zusammengefaßt und belegen den nächsten Speicherplatz. Die resultierende Tastenfolge ist 61 01 36 im Programmspeicher. Bei dieser kompakten Art der Speicherung spricht man von Kombi-Befehlen, wobei die Kodekombinationen vollautomatisch vom Rechner vorgenommen werden.



Versuchen Sie folgende Übung mit Ihrem Rechner;

Taste	Anzeige	Bemerkungen
2nd CP CLR	0.	Löschen des Programmspeichers und der Anzeige
GTO 136	0.	Einstellung des Programmzeigers auf Speicherplatz 136
LRN	136 00	
R/S	137 00	R/S belegt den Speicherplatz 136
LRN RST LRN	000 00	Rückkehr auf 000
GTO	001 00	GTO belegt Speicherplatz 000
1	001 00	
3	001 00	
6	003 00	Der Programmspeicher erwartet alle drei Stellen, ehe die Adresse gespeichert wird
BST	002 36	
BST	001 01	Die Adresse wird gespeichert, wie bereits erläutert
BST	000 61	
LRN	0.	
R/S	0.	Durchführung des Programms
LRN	137 00	Sofortige Verzweigung zum Speicherplatz 136, wo der hier gespeicherte R/S-Befehl ausgeführt wird, und der Programmzeiger auf Speicherplatz 137 gerichtet bleibt.

Ein Label hätte die gleiche Wirkung. Gibt man zum Beispiel GTO x^2 manuell über die Tastatur ein, stellt sich der Programmzeiger auf Label x^2 ein, und weitere Befehle werden abgewartet. Wenn Sie den Rechner anweisen, ein Label aufzusuchen, das nicht existiert, blinkt der augenblickliche Wert in der Anzeige.

Aus diesen Ausführungen wird die Funktion des GTO-Befehls ersichtlich, wenn man ihn über die Tastatur eingibt und wenn er Bestandteil eines Programms ist. Die anderen Verzweigungen haben weitgehend die gleiche Wirkung. Wenn sie manuell über die Tastatur eingegeben werden oder in einem Programm vorkommen, verzweigen sie zum angegebenen Programmspeicherplatz.



UNTERPROGRAMME

Wenn Sie anfangen, mehr Programme zu schreiben, finden Sie oft Rechenfolgen, die wiederholt durchgeführt werden müssen. Diese Folgen bezeichnet man als Unterprogramme. Unterprogramme geben Ihnen die Möglichkeit, einen „Nebenprozeß“ oder eine Folge von Tastenbefehlen zu definieren, die einen außergewöhnlichen Zweck erfüllen. Man kann sie kennzeichnen und jederzeit von jeder Stelle im Programm auf sie zurückgreifen. Dieser Vorgang ist beinahe so einfach, als wäre der Funktion des Unterprogramms eine Taste auf der Tastatur zugewiesen. Nach Abschluß eines Unterprogramms richtet sich der Programmzeiger automatisch auf den ersten Programmspeicherplatz, der dem Punkt folgt, wo Sie mit dem Unterprogramm anfangen. Wenn Sie ein Unterprogramm verwenden, spricht man oft vom „Aufruf“ dieses Programtteils. Sie weisen damit den Rechner an, eine ganze Schrittfolge mit einem einzigen Unterprogrammbefehl auszuführen.

Es ist eine empfehlenswerte Programmiertechnik, die Programme so zu schreiben, daß man sie als Unterprogramme verwenden kann. Damit können sie ohne Änderungen auch von anderen Programmen benutzt werden. Um dies zu erreichen, gibt man anstelle der **R/S**-Anweisung den Befehl **INV SBR**, wenn der Programmlauf unterbrochen werden soll. Die folgenden Programme in diesem Abschnitt sind auf diese Weise geschrieben.

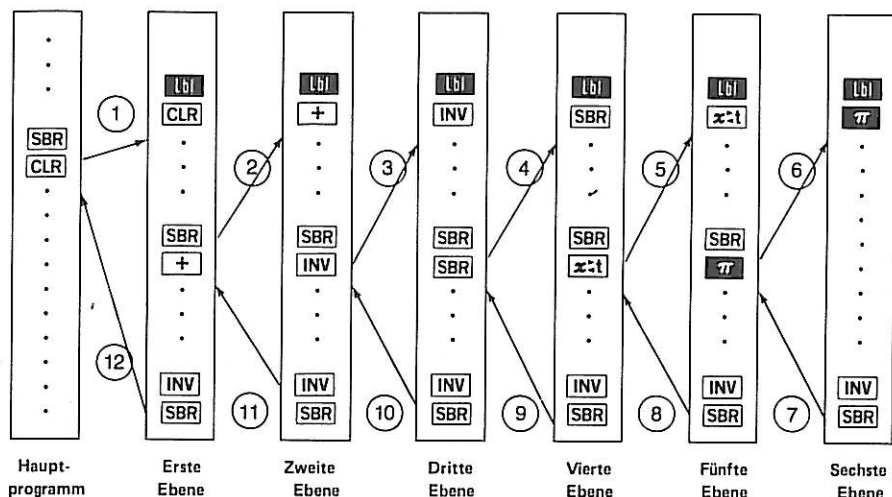
DER UNTERPROGRAMMBEFEHL — **SBR**

Der Unterprogrammbefehl ist ein zweifach modifizierter Sprungbefehl — über die Tastatur kann er den Programmablauf starten und im Programm kontrolliert er, von welcher Stelle aus die Verzweigung erfolgte. Gibt man **SBR 136** manuell über die Tastatur ein, wird der Programmzeiger sofort auf Speicherplatz 136 eingestellt und der Programmablauf beginnt automatisch an dieser Stelle. Die Wirkung ist identisch mit der Tastenfolge **GTO 136 R/S**. Der gleiche Vorgang findet statt, wenn man **SBR ∞** drückt. Der Programmablauf beginnt bei Label x^2 , unabhängig, an welcher Stelle das Label sich befindet.

Hätte man **SBR 136** wie **GTO 136** im vorigen Beispiel in die Speicherplätze 000 bis 002 eingegeben, wäre die Folge genauso durchgeführt worden. Wenn aber das Unterprogramm abläuft, wird die Programmadresse 003 im Unterprogrammrücksprungsregister gespeichert. Wenn also eine Rechenfolge, die mit **INV SBR** endet, bei 136 beginnt, wird bei der Verarbeitung das Unterprogrammrücksprungs-Register kontrolliert, und dort ist 003 gespeichert. Der Verarbeitungsfluß springt auf 003 und wird dort fortgesetzt. Schließen Sie ein Unterprogramm mit **INV SBR** ab, um die Verarbeitung wieder auf die Ausgangsposition zu lenken. Der tatsächliche Vorgang war eine Verzweigung zu einem Unterprogramm bei 136 und eine automatische Rückkehr, als der Befehl **INV SBR** im Programm erreicht wurde.



Das Unterprogrammrücksprung-Register kann bis zu 6 Rücksprungadressen gleichzeitig speichern. Das heißt, ein Unterprogramm kann selbst ein Unterprogramm enthalten und aufrufen, das ebenfalls wieder ein Unterprogramm verwenden kann etc. – bis zu sechsmal. Diese große Kapazität in der Graphik:



Wenn die oben dargestellte Schrittfolge als Teil eines Programms geschrieben wäre, würde die Verarbeitung entsprechend der obigen Numerierung ablaufen. Beachten Sie, daß jedes Unterprogramm mit **INV SBR** abgeschlossen ist. Damit wird der Rechner angewiesen, aus dem Unterprogrammrücksprung-Register die zuletzt gespeicherte Adresse wiederzugewinnen und dorthin zu verzweigen. Im allgemeinen endet der Verarbeitungsfluß wieder im Hauptprogramm – dem Programm, das mit dem Aufruf der Unterprogramme begann. Übrigens wird **INV SBR** im Programmspeicher kombiniert und belegt nur einen Speicherplatz mit dem Tastenkode 92. Dieser Kode wird nicht aus der Reihe-/Spalte-Zuordnung hergeleitet.

Wenn ein Programmteil mit einer Programmadresstaste gekennzeichnet ist, kann dieser Teil über die Tastatur ausgeführt werden, wenn Sie einfach das entsprechende Label drücken. Der Vorgang findet statt, wenn eine dieser Tasten Bestandteil des Programms ist – der Programmzeiger stellt sich auf das Label ein, und die Verarbeitung wird fortgesetzt. Diese Programmadresstasten enthalten einen internen automatischen **SBR**-Befehl. Wenn Sie also ein Programmsegment mit einer Programmadresstaste kennzeichnen, und mit **INV SBR** abschließen, wird dieser Teil so verarbeitet, als wäre er mit dem **SBR**-Befehl aufgerufen worden.

Anmerkung: Die Tasten **A** - **E** definieren bei unbedingter Verzweigung eine Unterprogrammebene. Um Schwierigkeiten beim Programmablauf zu vermeiden, empfiehlt es sich, die Labeltasten **A** - **E** in Verbindung mit der **GTO**-Taste zu programmieren.



AUFRUF VON UNTERPROGRAMMEN

Laut Definition ist ein Unterprogramm ein Programmsegment, das für einen bestimmten Zweck erstellt wurde — es wird einmal geschrieben, aber wiederholt verwendet. Alle Unterprogramme müssen mit **INV** **SBR** enden, um den Verarbeitungsfluß wieder auf die aufrufende Folge zu lenken.

Es gibt drei Methoden, mit denen ein Unterprogramm aufgerufen werden kann:

Absolute Adresse, **SBR** 136

Allgemeines Label, **SBR** x^2

Programmadress-Label, **A**

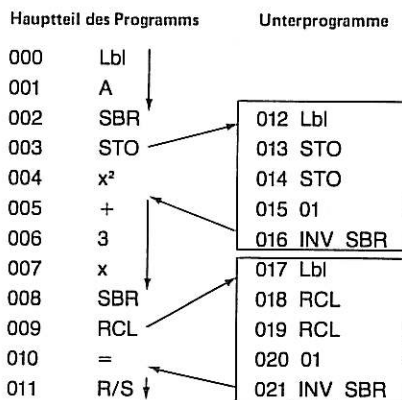
Die Kennzeichnung von Unterprogrammen mit Labels trägt zur Klarheit und Einfachheit der Programmbefehle bei. Man kann sogar Labels verwenden, die bereits den Zweck des Unterprogramms andeuten. Wählen Sie Ihre Labels mit Bedacht und notieren Sie deren Funktion. Gekennzeichnete Unterprogramme können an beliebiger Stelle im Programmspeicher eingebracht werden, weil beim Aufruf das Label unabhängig von seiner Position im Programmspeicher aufgefunden wird. Dazu kommt, daß ein Unterprogramm mit einem Label nicht von **Ins** - oder **Del** -Befehlen beeinflusst wird, die vor dem Unterprogramm im Programmspeicher ausgeführt werden.

Für ein Programm, das $x^2 + 3x$ für eingehende x-Werte berechnet, müssen Sie nachstehende Folge eintasten:

STO
1
 x^2
+
3
X
RCL
1
=
R/S



Zur Erläuterung sollen Unterprogramme auch Speicherung und Aufruf übernehmen. Das Programm hat folgende Form:



Geben Sie einen beliebigen x -Wert ein und drücken Sie **A**. Die Verarbeitungsfolge ist mit Pfeilen dargestellt. Beachten Sie, daß für den Programmstart die einfachste Möglichkeit gewählt wurde – die Anwendung eines Programmadress-Labels. Die Unterprogrammtitel wurden so ausgesucht, daß die dazugehörigen Segmente gleichlauten, andere Labels hätte man jedoch ebenso gut verwenden können.

WICHTIGE FAKTOREN, DIE BEI UNTERPROGRAMMEN ZU BEACHTEN SIND

RST und **=** sind die beiden Befehle, die in Unterprogrammen nur mit Vorsicht anzuwenden sind. Außerdem müssen Sie sicherstellen, daß das Unterprogrammrücksprung-Register vor Beginn einer neuen Aufgabe gelöscht wird.

Der Reset-Befehl löscht neben seinen anderen Funktionen automatisch das Unterprogrammrücksprung-Register. Wenn Sie in einem Unterprogramm auf Speicherplatz 000 zurückgehen müssen (die primäre Funktion der Taste **RST**), verwenden Sie **GTO** 000 oder ein Label, wenn in 000 ein Label gespeichert ist.

Der Gleichheitsbefehl schließt alle unvollständigen Operationen ab. Verwendet man ihn in einem Unterprogramm, schließt er nicht nur die unvollständigen Operationen des Unterprogramms, sondern auch die des Hauptprogramms ab.



IV

Prüfen Sie folgendes Programmsegment zur Berechnung von $4 + (1 + 2) \times 3$.

4
+
SBR x^2
X
3
=
R/S
2nd [Lb] x^2
1
+
2
=
INV SBR

Die Gleichheitsanweisung hier im Unterprogramm $\text{[x}^2\text{]}$ schließt vor der Rückkehr zu $\times 3$ nicht nur das $1 + 2$ ab, sondern auch $4 +$. Man erhält deshalb 21 als Ergebnis, richtig ist aber 13.

Das Programm läßt sich leicht so ändern, daß die Aufgabe richtig bearbeitet wird. Setzen Sie Klammern zur Berechnung des Unterprogramms.

4
+
SBR x^2
X
3
=
R/S
2nd [Lb] x^2
(
1
+
2
)
INV SBR

Diese Folge berechnet das richtige Ergebnis, 13.



Es sollte zur Gewohnheit werden, jedes Unterprogramm mit einer linken Klammer **[** zu beginnen, und die rechte Klammer **]** vor dem Befehl **[INV] [SBR]** zu setzen. Mit Klammern benötigen Sie zwar einen Tastendruck mehr als mit **[=]**, aber Sie können viele falsche Berechnungen vermeiden. Der eigentliche Vorteil besteht darin, daß bei Klammern nur die unvollständigen Operationen innerhalb des Klammerpaares beeinflußt werden.

Wenn Sie den Gleichheitsbefehl **[=]** in solchen Fällen vermeiden, wird damit die Aufgabe keineswegs erschwert. Klammern dienen dazu, derartige Ausdrücke von anderen getrennt zu berechnen. Einige wissenswerte Informationen müssen jedoch noch für die Anwendung des augenblicklichen Anzeigeregisterwertes in Unterprogrammen beachtet werden.

Wenn ein Unterprogramm zur Zeit des Aufrufs wiederholten Zugriff auf den Inhalt des Anzeigeregisters erfordert, sollten Sie vor Ausführung von Berechnungen die Variable in einem Datenregister speichern und bei Bedarf aufrufen. Ist der Inhalt des Anzeigeregisters nur für den Beginn einer Berechnung erforderlich, kann man mit der Taste **[CE]** einen Trick anwenden, um diesen Wert in den Klammerausdruck hineinzuziehen. Dieser Trick wirkt als Teil eines Programms genauso wie bei manueller Anweisung über die Tastatur.

Drücken Sie: 2.18 **[X] [] [CE] [+]** 6 **[] [=]**

Anzeige: 17.8324

In der obigen Folge bringt die Taste **[CE]** den Wert 2.18 in den Klammerausdruck hinein, und ermöglicht dem Rechner, $2.18 \times (2.18 + 6) = 17.8324$ zu berechnen.

Gelegentlich erstellen Sie ein Programm vielleicht in der Weise, daß es innerhalb eines Unterprogramms abgeschlossen wird. Mit anderen Worten, das Ergebnis der Aufgabe wird ohne Rückgabe der Steuerung an das aufrufende Programm ermittelt. In diesen Situationen bleibt die Rückgabe der Steuerung offen, weil das Unterprogrammrücksprung-Register nicht gelöscht wird. Wenn Sie den Rechner nicht ausschalten oder nicht die Taste **[RST]** drücken oder das Rücksprungregister nicht mit dem Befehl **[2nd] [CP]** löschen, können beim Ablauf einer neuen Aufgabe Schwierigkeiten auftreten, weil das Programm zu den früheren Rücksprungadressen falsch verzweigen könnte. Um fehlerhafte Lösungen künftiger Probleme durch solche übriggelassene Rücksprungadressen zu vermeiden, sollten Sie grundsätzlich mit dem **[RST]**-Befehl das Unterprogrammrücksprung-Register löschen. Die Löschung kann manuell erfolgen, vorzugsweise setzen Sie aber den **[RST]**-Befehl an gesignierter Stelle im Programm ein.



SOFTWAREPROGRAMME ALS UNTERPROGRAMME

Es wird oft zweckmäßig sein, die Leistung des eigenen geschriebenen Programms durch die Verbindung mit einem Softwareprogramm zu erweitern. Die gleichen Befehle, mit denen der Zugriff auf diese Programme über die Tastatur möglich ist, können auch in das Programm eingefügt werden. Die Softwareprogramme sind dann einfach Unterprogramme Ihres eigenen Programms im Programmspeicher. In jedem der Software-Module sind Tausende von Befehlen gespeichert, die als Unterprogramme verwendet werden können. Jedes Softwareprogramm selbst ist ein Unterprogramm. Sie haben, mit wenigen Ausnahmen, keine [=] - oder [RST]-Befehle, werden mit Klammern berechnet und enden mit [INV] [SBR] .

Drückt man [2nd] [PgM] mm über die Tastatur, stellt sich der Programmzeiger auf das Softwareprogramm Nummer mm ein. Bis nach Abschluß des Softwareprogrammsegments bleibt der Programmzeiger in diesem Programm.

In einem Programm wirkt die Funktion von [PgM] ähnlich wie der [SBR]-Befehl. Der einzige Unterschied liegt darin, daß [2nd] [PgM] mm den Rechner veranlaßt, das Unterprogramm im Softwareprogramm mm aufzufinden und nicht im Programmspeicher des Rechners selbst. Sobald die Software-Routine abgeschlossen ist, kehrt der Programmzeiger an die Aufrufstelle im Programmspeicher zurück und die Verarbeitung wird normal fortgesetzt. Die zweistellige Programm-Nummer wird kombiniert und belegt einen einzigen Speicherplatz.

Wenn ein Segment der Software-Routine mit einer Programmadress-Taste wie [A] gekennzeichnet ist, muß [2nd] [PgM] mm [A] die Programmfolge sein, mit der diese Routine ausgeführt wird. Ist ein allgemeines Label wie [tan] verwendet, lautet die Folge [2nd] [PgM] mm [SBR] [2nd] [tan] . Achtung: Wenn nach [2nd] [PgM] mm kein [SBR]-Befehl oder keine Programmadress-Taste folgt, ist die Tastenfolge unzulässig und kann zu falschen Resultaten führen.

(Routine = Programm oder Programmteil, jeweils abgeschlossen — Anm. d. U.)



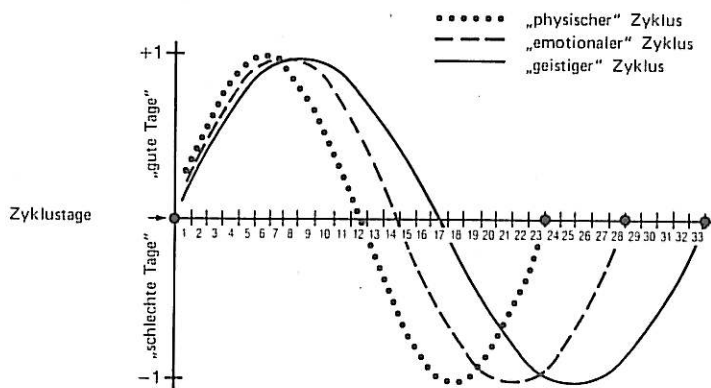
BIORHYTHMUS - PROGRAMM

Prüfen Sie ein typisches Anwendungsbeispiel für Unterprogramme in einer interessanten Programmsituation.

Nach der Theorie der Biorhythmen gibt es drei Zyklen in Ihrem Leben, die mit dem Tag der Geburt begannen:

1. Der physische Zyklus – 23-Tage-Perioden
2. Der emotionale Zyklus – 28-Tage-Perioden
3. Der geistige Zyklus – 33-Tage-Perioden

Man sagt, die erste Hälfte jedes Zyklus enthält Ihre „guten Tage“, während Ihre „schlechten Tage“ in die zweite Hälfte fallen.



Mit der folgenden Gleichung kann man die Amplituden dieser Biorhythmuszyklen als einen Wert zwischen -1 und +1 ausdrücken.

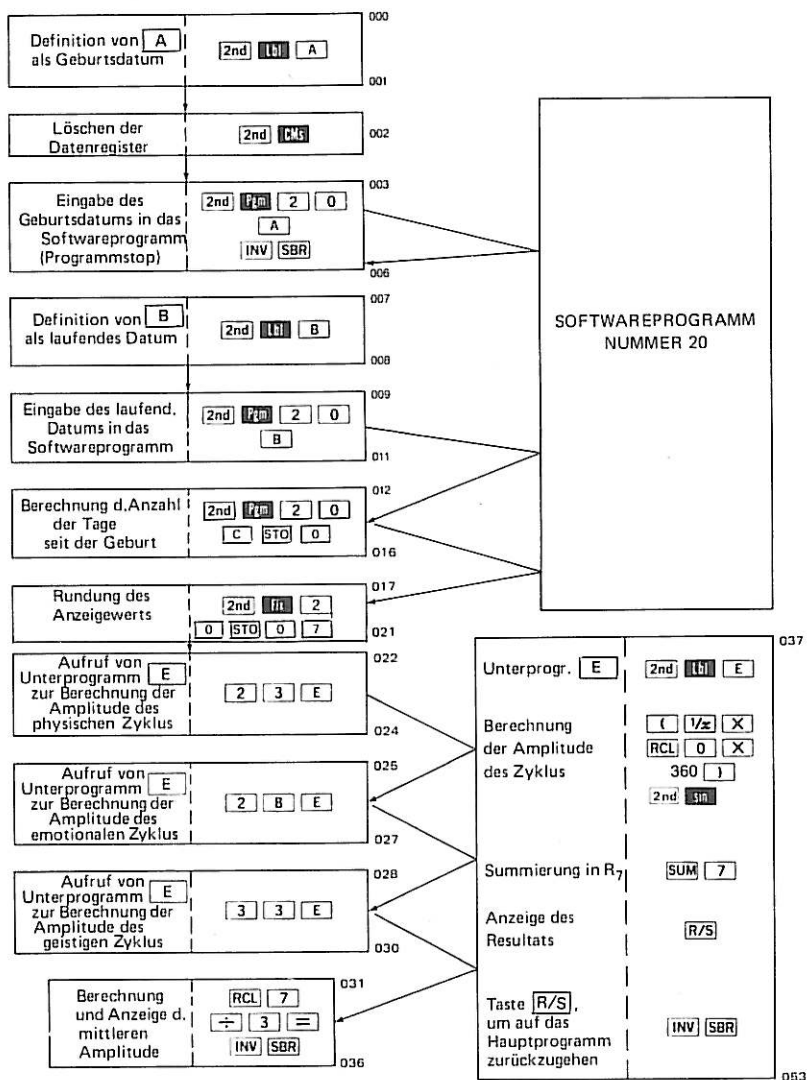
$$\text{Amplitude} = \sin \left(360 \times \frac{\text{Anzahl der Tage seit der Geburt}}{\text{Anzahl der Tage im Zyklus}} \right)$$

Nun soll ein Programm geschrieben werden, mit dem man die Amplituden der biorhythmischen Zyklen einer Person bestimmen kann. Da jeder Zyklus auf der selben Gleichung basiert, kann man zur Vereinfachung des Programms ein Unterprogramm verwenden. Außerdem können Sie auch Programm 20 der Standardsoftware-Programme als Unterprogramm heranziehen, um die Anzahl der Tage seit der Geburt zu berechnen.

Im folgenden Musterprogramm verwendet man das Unterprogramm mit dem Label **E** zur Berechnung der Amplitude für jeden Zyklus, da der einzige Unterschied zwischen den Berechnungen die Anzahl der Zyklostage ist. Man kann das Programm mit **R S** unterbrechen, und die Amplituden des physischen, emotionalen und geistigen Zyklus in dieser verbindlichen Reihenfolge anzeigen. **INV SBR** wird hier nicht verwendet, weil dieser Befehl die Programmsteuerung an das Hauptprogramm zurückgeben und nicht den Ablauf unterbrechen würde. (Siehe „Der Unterprogrammbefehl – **SBR**“ einige Seiten vorher.) Das Endergebnis des Programms ist das Mittel aus diesen drei Werten. Alle Ergebnisse werden für die Anzeige auf zwei Dezimalstellen gerundet.



IV



Biorhythmus-Programm


**Speicherplatz
und Tastenkod**

000 76	2nd [Lb]
001 11	A
002 47	2nd [CMs]
003 36	2nd [Pgm]
004 20	2 0
005 11	A
006 92	INV [SBR]
007 76	2nd [Lb]
008 12	B
009 36	2nd [Pgm]
010 20	2 0
011 12	B
012 36	2nd [Pgm]
013 20	2 0
014 13	C
015 42	[STO]
016 00	0
017 58	2nd [fir]
018 02	2
019 00	0
020 42	[STO]
021 07	0 7
022 02	2
023 03	3
024 15	E
025 02	2
026 08	8

**Speicherplatz
und Tastenkod**

027 15	E
028 03	3
029 03	3
030 15	E
031 43	[RCL]
032 07	7
033 55	÷
034 03	3
035 95	=
036 92	INV [SBR]
037 76	2nd [Lb]
038 15	E
039 53	[]
040 35	[1/x]
041 65	[X]
042 43	[RCL]
043 00	0
044 65	X
045 03	3
046 06	6
047 00	0
048 54	[]
049 38	2nd [sin]
050 44	[SUM]
051 07	7
052 91	[R S]
053 92	INV [SBR]

Biorhythmus-Programm



PROGRAMM-INSTRUKTIONEN				
Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Löschen des Programmspeichers und Rückstellen des Programmzeigers		2nd CP	
2	Eingabe des Learn-Modus		LRN	000 00
3	Eingabe des Biorhythmus-Programms			
4	Aufheben des Learn-Modus		LRN	
5	Eingabe des Geburtsdatums	MMTT.JJJJ	A	0.
6	Eingabe des laufenden Datums und Berechnung der Amplitude des physischen Zyklus	MMTT.JJJJ	B	Amplitude des physischen Zyklus
7	Berechnung der Amplitude des emotionalen Zyklus		R/S	Amplitude des emotionalen Zyklus
8	Berechnung der Amplitude des geistigen Zyklus		R/S	Amplitude des geistigen Zyklus
9	Berechnung der mittleren Amplitude		R/S	mittlere Amplitude

Beispiel: Fred ist am 2. Mai 1944 geboren. Berechnen Sie seinen Biorhythmus für den 1. März 1977. (Es wird vorausgesetzt, daß das Biorhythmus-Programm noch im Speicher ist.)

MMTT.JJJJ = Monat Tag, Jahr

Taste	Anzeige	Bemerkungen
502.1944 A	0.	Eingabe des Geburtsdatums
301.1977 B	0.82	Amplitude des physischen Zyklus
R/S	1.00	Amplitude des emotionalen Zyklus
R/S	0.76	Amplitude des geistigen Zyklus
R/S	0.86	Mittlere Amplitude

Fred scheint für alle seine Unternehmungen in ziemlich guter Verfassung zu sein. Bestimmen Sie mit diesem Programm doch auch, wo Sie in Ihren biorhythmischen Zyklen liegen.

Wenn Sie Softwareprogramme als Unterprogramme in Ihren eigenen Programmen verwenden, achten Sie besonders darauf, welche Datenregister Sie belegen, und welche Register im Softwareprogramm benutzt werden. Der Versuch, in beiden Programmen die gleichen Datenregister für verschiedene Zwecke zu belegen, kann falsche Ergebnisse zur Folge haben.



Bedingte Verzweigungen (Entscheidungsbefehle)

Andere vorteilhafte Eigenschaften für die Problemlösung sind die Befehle, die in Ihrem Programm Entscheidungen treffen können. Die Gruppe der sogenannten bedingten Verzweigungsbefehle (bedingte Sprungbefehle) macht diesen Entscheidungsprozess möglich. Jedesmal, wenn einer dieser Befehle in einem Programm vorkommt, wird zuerst ein Test durchgeführt und abhängig vom Ausgang des Tests die Entscheidung getroffen, ob die Verzweigung stattfindet oder nicht.

Es gibt drei Arten von bedingten Sprungbefehlen, die nach dem Gegenstand des Tests differenziert werden.

1. Vergleich des Anzeigeregister-Inhalts mit dem Inhalt des T-Registers **cc=t** , **cc=t**
2. Test der Inhalte der Datenregister 0 - 9 **0=t**
3. Test des Status der Programmflags **000g**

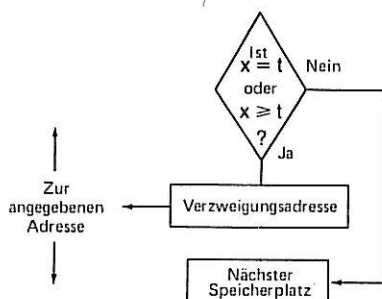
Nach jedem dieser Befehle steht die Verzweigungsadresse. Zu dieser Adresse erfolgt die Verzweigung, wenn die Antwort auf den Test „ja“ lautet, (Test positiv). Ist die Antwort auf den Test „nein“ (Test negativ), wird die Verzweigung übersprungen. Wenn zum Beispiel der Test **2nd cc=t A** positiv ist, verzweigt das Programm zu **A**, als wäre der Befehl **GTO A** vorgekommen. Wenn x nicht genau gleich dem T-Register-Wert ist, findet keine Verzweigung statt.

ANZEIGEREGISTER-T-REGISTER-VERGLEICH

Was ist das T-Register? T steht für „TEST“ und das Register selbst ist eine Art Datenregister, wo Zahlen gespeichert, aufgerufen und mit der angezeigten Zahl verglichen werden können. Mit der x-t-Austausch-Taste **x=t** werden Zahlen in das T-Register eingebracht und aus dem T-Register herausgenommen. Diese Taste vertauscht einfach den Wert x im Anzeigeregister gegen den Wert t im T-Register. Zu Beginn ist das T-Register mit 0 belegt. Tasten Sie 5 ein und drücken Sie **x=t**. Jetzt ist Null in der Anzeige und 5 im T-Register. Drücken Sie **x=t** erneut, und 5 kommt wieder in die Anzeige, während Null in das T-Register zurückgeht.



Zum Vergleich des jeweiligen Anzeigeregisterwertes mit dem Inhalt des T-Registers gibt es mehrere Testbefehle, „x gleich t“ **cc=1** und „x größer oder gleich t“ **cc=1**. Jedem dieser Befehle muß immer eine absolute Programmspeicheradresse oder ein Label folgen. Wenn ein Test durchgeführt wird, zum Beispiel „Ist $x = t$ “, und die Antwort ist „ja“, verzweigt das Programm zu der Adresse nach dem Testbefehl. Bei negativem Test wird die Adresse ignoriert und die Verarbeitung so fortgesetzt, als hätte überhaupt kein Test stattgefunden. Nachstehend der Vorgang in graphischer Darstellung:



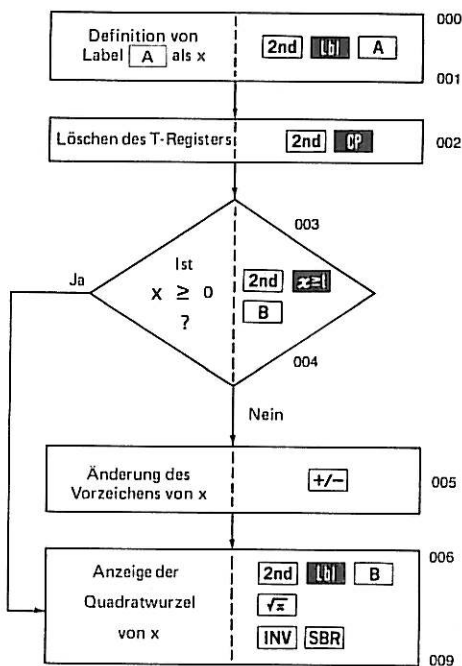
Diese Befehle dienen primär für Anwendung innerhalb des Programms, um den Verarbeitungsfluß zu steuern, sie können aber genauso über die Tastatur durchgeführt werden. Versuchen Sie die folgenden Tastenbefehle.

Taste	Anzeige	Bemerkungen
5 cc=1	0.	5 wird im T-Register gespeichert
6 2nd cc=1 123	6.	Eingabe von 6 in die Anzeige und Anweisung an den Rechner, zum Programmspeicherplatz 123 zu verzweigen, wenn $x \geq t$
LRN	123 00	Kontrolle, daß die Verzweigung stattfand, weil 6 größer als 5 ist
LRN 4 2nd cc=1 111	4.	Zurück auf Tastatursteuerung und Test für $x = 4$
LRN	123 00	Immer noch Speicherplatz 123, die Verzweigung fand nicht statt, weil 4 nicht größer als 5 ist
LRN 5 2nd cc=1 111	5.	Zurück auf Tastatursteuerung und Test für $x = 5$
LRN	111 00	Jetzt fand die Verzweigung statt, weil die Anzeige gleich dem T-Registerwert 5 ist.



QUADRATWURZEL – BEISPIEL

Aufgabe: Berechnen Sie die Quadratwurzel von jeder Zahl x , die in die Anzeige eingegeben wird. Ist die Zahl negativ, ändern Sie das Vorzeichen und berechnen dann die Quadratwurzel.





IV

Speicherplatz und Tastenfolge

000 76
001 11
002 29
003 77
004 12
005 94
006 76
007 12
008 34
009 92

Tastenfolge

2nd Lbl
A
2nd CP
2nd x=I
B
+/-
2nd Lbl
B
√
INV SBR

Jetzt können Sie das Programm durchführen. Geben Sie zum Beispiel 4 ein und drücken Sie **A**. Das Ergebnis ist 2. Gibt man -4 ein und drückt dann **A**, erhalten Sie dasselbe Ergebnis, 2.

Diese Testbefehle können auch mit der Taste **INV** verwendet werden, um die Verzweigungsbedingungen umzukehren. Siehe folgende Darstellung:

Befehlsfolge

2nd x=I

INV 2nd x=I

2nd x≥I

INV 2nd x≥I

Fragestellung (Test)

Ist der Wert im Anzeigeregister gleich dem Wert im T-Register?

Ist der Wert im Anzeigeregister ungleich dem Wert im T-Register?

Ist der Wert im Anzeigeregister größer oder gleich dem Wert im T-Register?

Ist der Wert im Anzeigeregister kleiner als der Wert im T-Register?

Wird eine der obigen Fragen mit „ja“ beantwortet, verzweigt der Verarbeitungsfluß zu der Adresse, die unmittelbar nach diesem Befehl angegeben ist. Ist die Antwort „nein“, wird die beigelegte Adresse einfach ignoriert und die Verarbeitung geht zum nächsten Speicherplatz des Programmspeichers über.



FLAG-OPERATION

Was ist ein Programmflag und welche Funktion hat es im Programm? Ein Flag ist ein internes Schaltsystem in "EIN"- oder "AUS"-Position. (Bildhaft ausgedrückt, ist das Programmflag entweder aufgerichtet oder gesenkt.) Ein Flag kann man an einer Stelle im Programm setzen und zu einem späteren Zeitpunkt testen. Dieses Setzen und Rücksetzen (bzw. Errichten und Senken) und Testen von Flags erfolgt unabhängig vom Anzeigeregister und vom Datenspeicher.

Wann sind Flags angebracht? Flags haben eine Vielzahl von Verwendungszwecken, drei davon sind nachstehend aufgeführt.

- Steuerung von Programmoptionen manuell über die Tastatur vor dem Programmablauf.
- Die Programmbedingungen setzen ein Flag für den späteren Test.
- Kontrolle der Entwicklung des Programmablaufs – welcher Weg durch das Programm führte zur augenblicklichen Position?

10 einzelne Flags, numeriert von 0 bis 9, stehen Ihnen zur Verfügung. Folglich müssen Sie mit jedem Flagbefehl angeben, auf welches Flag Sie sich beziehen.

Nachstehend eine Definition der Befehle, die die Flags steuern:

- **2nd** **STng** y – Flag y setzen.
- **INV** **2nd** **STng** y – Flag y rücksetzen (Herstellung des definierten Ausgangszustandes).
- **2nd** **ITng** y – Test von Flag y und Verzweigung, wenn es gesetzt ist. Dieser Befehl muß wie die früher behandelten Testbefehle mit einer Verzweigungsadresse vervollständigt werden.
- **INV** **2nd** **ITng** y, gefolgt von einer Verzweigungsadresse – Test von Flag y und Verzweigung, wenn es nicht gesetzt ist.



IV

Diese Befehle können manuell über die Tastatur gegeben werden oder Bestandteil eines Programms sein. Tasten Sie folgendes ein und beobachten Sie die Wirkung der Flags.

Taste	Anzeige	Bemerkungen
2nd CP	0.	Löschen des Programmspeichers und der Anzeige. Mit diesem Befehl werden auch alle Programmflags rückgesetzt und das T-Register gelöscht.
2nd ST/IF 4	0.	Flag 4 wird gesetzt.
2nd IF/IF 4 136	0.	Test von Flag 4, Wenn es gesetzt ist, Sprung zum Speicherplatz 136.
LRN	136 00	Die Verzweigung zu 136 fand statt, weil das Flag gesetzt ist.
LRN 2nd IF/IF 5 111	0.	Test von Flag 5. Wenn es gesetzt ist, Sprung zum Speicherplatz 111.
LRN	136 00	Die Verzweigung fand nicht statt, weil das Flag nicht gesetzt ist.
LRN INV 2nd ST/IF 4	0.	Flag 4 wird rückgesetzt.
2nd IF/IF 4 222	0.	Test von Flag 4. Wenn es gesetzt ist, Sprung zu Speicherplatz 222.
LRN	136 00	Das Flag ist nicht gesetzt, also findet keine Verzweigung statt.
LRN INV 2nd IF/IF 4 222	0.	Test von Flag 4 – Wenn es nicht gesetzt ist, Sprung zu 222.
LRN	222 00	Das Flag ist nicht gesetzt, also verzweigt das Programm zu 222.

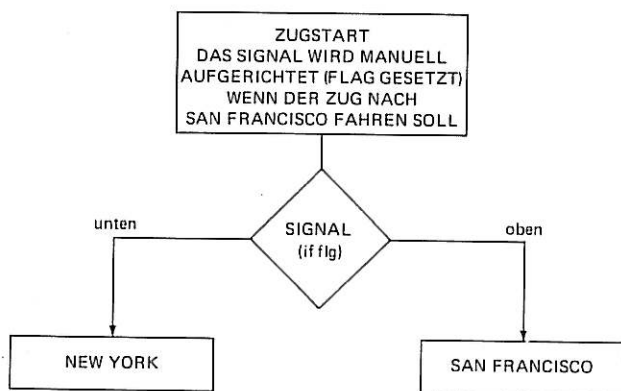
Der Flagtest-Befehl verhält sich ähnlich wie die T-Register-Tests. Der Unterschied ist, daß diese Befehle den Zustand der Flags testen, während mit den T-Register-Tests die Inhalte von Anzeige und T-Register verglichen werden. Bedenken Sie, daß die Verzweigungsadresse nach dem Befehl eine absolute Adresse sein kann wie in der obigen Übung, aber auch ein Label jedes Typs (Programmadress-Label oder allgemeines Label).

Das Setzen eines bereits gesetzten Flags, das Rücksetzen eines bereits rückgesetzten Flags und das Testen eines Flags haben keinen Einfluß auf den Flagstatus noch auf Berechnungen. Alle Flags können sofort mit **RST** oder **2nd** **CP** rückgesetzt werden.

Beachten Sie außerdem, daß Sie in der Anzeige nicht direkt (wie beim T-Register oder einem anderen Datenspeicher) verfolgen können, ob ein Flag gesetzt oder rückgesetzt ist. Der Flagstatus kann erst durch den Test sichtbar gemacht werden.



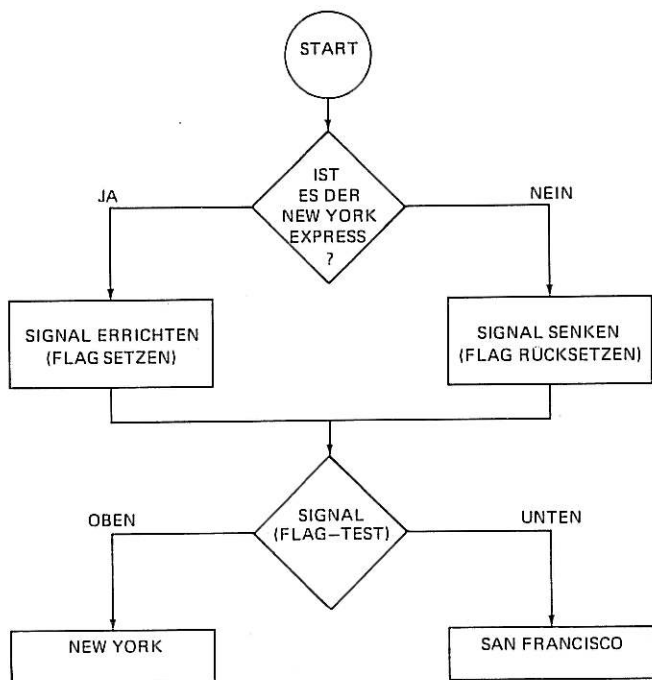
Die folgende Situation veranschaulicht den zuerst genannten Anwendungszweck für Flags, wie sie manuell über die Tastatur benutzt werden können. Angenommen, Sie sind Fahrdienstleiter in einem Stellwerk. Ein Zug fährt vorbei und trifft an einer Weiche auf ein Signal. Wenn das Signal oben ist, wird der Zug auf das Gleis nach San Francisco gelenkt. Ist das Signal unten, fährt der Zug auf das Gleis nach New York. Als Fahrdienstleiter müssen Sie das Signal bedienen, das wiederum die Richtung des Zuges steuert. Genauso können Sie Flags manuell setzen und rücksetzen, um zu bestimmen, welcher Teil des Programms durchzuführen ist.



Manuelle Flag-Operation (über die Tastatur)

Wenn Sie ein Programm verarbeiten, können Sie die Flags manuell setzen, um die Programmoperation zu kontrollieren. Sie haben zum Beispiel ein Kostenkontrollprogramm im Rechner, mit dem eine Menge von Gutschriften und Lastschriften verarbeitet werden müssen. Da man die Lastschriften anders als die Gutschriften behandeln muß, setzt man ein Flag und schreibt das Programm so, daß das Flag getestet und die eingehenden Werte entsprechend verarbeitet werden.

An der folgenden Modifizierung des Zugbeispiels läßt sich zeigen, wie die Züge selbst Signale errichten oder senken können. Diese Situation demonstriert das Prinzip, wo durch die Programmbedingungen Flags gesetzt werden. Vorausgesetzt sei, daß die Schnellzüge New York und San Francisco gesondert zu ihren Bestimmungsorten geleitet werden.



Automatische (Programm-)Flag-Operation

Das Steuersystem fragt bei jedem Zug: „Ist es ein New York Express?“ Wenn die Antwort „ja“ lautet, wird das Signal errichtet. Ist die Antwort „nein“, wird das Signal gesenkt. Dieses Signal (Flag) wird für die spätere Steuerung getestet – ist es aufgerichtet, fährt der Zug nach New York, ist es gesenkt, wird der Zug nach San Francisco gelenkt. Ähnlich kann man im Programm die Eigenschaften des zuletzt berechneten Wertes abfragen: „Ist der Wert negativ?“ oder „Ist der Wert größer als 1.000?“ und viele andere Fragen mehr. Kann man die Frage mit „ja“ beantworten, wird ein Flag gesetzt und später getestet, wenn man zwischen alternativen Verarbeitungsabläufen wählen muß.

Der dritte Anwendungszweck von Flags gibt dem Programm eine Möglichkeit, nachträglich festzustellen, wie ein gegebener Punkt erreicht wurde. Diese Kontrolle ist in den Situationen notwendig, wo die weitere Verarbeitung vom bisherigen Programmablauf abhängig ist. Sie erinnern sich, daß der Programmzeiger nur seine augenblickliche Einstellung kennt, aber nicht nachvollziehen kann, welcher Weg zu dieser Position führte. Diese Erinnerungsfähigkeit ist jedoch in einigen Fällen erforderlich, und die Programmflags bieten hierzu eine vorteilhafte Möglichkeit. Programmieren Sie **2nd** **STO** y in den einen Verarbeitungsweg, und **INV** **2nd** **STI** y in den anderen. Es ist nicht angebracht, diesen anderen Weg frei zu lassen, da spätere Programmläufe Fehler verursachen können, wenn das Flag nicht rückgesetzt ist. Später können Sie dann mit **2nd** **IF** y leicht feststellen, welcher Verarbeitungsweg eingeschlagen wurde. Ein Flag kann zum Beispiel angeben, welcher von zwei möglichen Zinssätzen im Programm verwendet wurde, oder ob das Vorzeichen einer Zahl geändert werden muß, ehe sie verarbeitet werden kann etc..... – die Reihe der Alternativen läßt sich beliebig fortsetzen.



SONDERFUNKTIONEN VON FLAGS

Einige Flags sind intern für die Ausführung folgender Sonderfunktionen vorprogrammiert.

- Flag 7** **2nd** **Op** 18 weist den Rechner an, Flag 7 zu setzen, wenn keine Fehlerbedingung existiert. Wenn hingegen eine Fehlerbedingung existiert, wird Flag 7 mit der Folge **2nd** **Op** 19 gesetzt. (Siehe spezielle Steueroperationen auf Seite V-27). Nur für den TI-58C gilt, daß die Folge **2nd** **Op** 40 den Rechner anweist, Flag 7 zu setzen, wenn ein PC-100A, B oder C angeschlossen ist.
- Flag 8** Wenn man Flag 8 setzt, unterbricht der Rechner ein Programm, wenn während des Programmlaufs ein Fehler auftritt.
- Flag 9** Wenn Sie den Rechner zusammen mit dem Drucker verwenden, können Sie mit Flag 9 den Protokollmodus (Parallelbetrieb) des Druckers steuern. Ist Flag 9 gesetzt, schaltet sich der Drucker in den Protokollmodus und nach jeder neuen Funktion oder Operation werden die Rechenergebnisse ausgedruckt. Ist Flag 9 rückgesetzt, erfolgt der Ausdruck der Resultate nur nach einem entsprechenden Druckbefehl. Ohne Drucker können Sie Flag 9 normal anwenden.

METRISCHES UMRECHNUNGSPROGRAMM

Entwickeln Sie ein Programm, das Meter in Fuß und Kilometer in Meilen umrechnet. Für diese Aufgabe gibt es natürlich nur wenige Lösungswege. Nach der folgenden Methode werden die Eingabedaten zuerst in Fuß umgerechnet und dann wird getestet, ob die Maßeinheit dieser Eingangsdaten Kilometer oder Meter war. Wenn der Test ergibt, daß der Eingabewert in Kilometern ausgedrückt war, rechnen Sie in Meilen um; wenn nicht, soll das Ergebnis in Fuß angezeigt werden. In R₁ wird das Zwischenergebnis während der Durchführung des Tests gespeichert. Umrechnungsfaktoren:

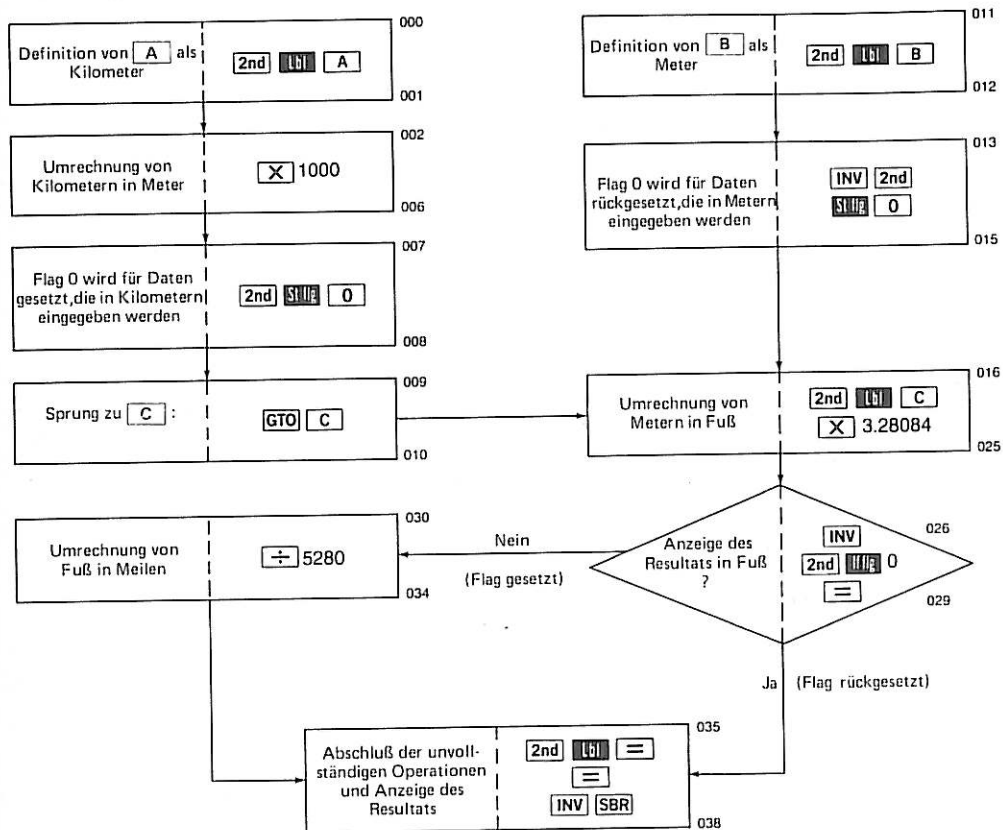
$$1 \text{ km} = 1000 \text{ Meter}$$

$$1 \text{ Meter} = 3.28084 \text{ ft.}$$

$$1 \text{ Meile} = 5280 \text{ ft.}$$



IV



Metrisches Umrechnungsprogramm

IV



PROGRAMM-INSTRUKTIONEN				
Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Löschen des Programmspeichers und Rückstellen des Programmzeigers		2nd CP	
2	Eingabe des Learn-Modus		LRN	000 00
3	Eingabe des metrischen Umrechnungsprogramms			
4	Eingabe von Kilometern ODER	Kilometer	A	Meilen
	Eingabe von Metern und Berechnung des Resultats	Meter	B	Fuß

Speicherplatz und Tastenkod

Tastenfolge

000 76	2nd LbI
001 11	A
002 65	X
003 01	1
004 00	0
005 00	0
006 00	0
007 86	2nd SI/IF
008 00	0
009 61	GTO
010 13	C
011 76	2nd LbI
012 12	B
013 22	INV
014 86	2nd SI/IF
015 00	0
016 76	2nd -LbI
017 13	C
018 65	X
019 03	3

Speicherplatz und Tastenkod

Tastenfolge

020 93	.
021 02	2
022 08	8
023 00	0
024 08	8
025 04	4
026 22	INV
027 87	2nd SI/IF
028 00	0
029 95	=
030 55	÷
031 05	5
032 02	2
033 08	8
034 00	0
035 76	2nd LbI
036 95	=
037 95	=
038 92	INV SBR



Beispiel: Geben Sie das obige Programm ein und rechnen Sie 50 Meter in Fuß und 90 Kilometer in Meilen um.

Eingabe	Taste	Anzeige	Bemerkungen
50	B	164.042	Meter → Fuß
90	A	55.92340909	Kilometer → Meilen

DATENREGISTERVERZWEIGUNGEN — **057**

Dieser leistungsfähige Befehl verwendet die Inhalte der Datenregister 0 bis 9 als Basis für die Entscheidung, ob eine Verzweigung stattfindet oder nicht. Der **057**-Befehl findet primär Anwendung bei der bedingten Schleifenbildung. Genaue Erläuterungen werden deshalb auf dieses Kapitel verschoben (IV-71).

Programmierung von Schleifen

Für die Lösung von Aufgaben benötigt man oft eine Befehlsfolge, die mehrmals in ununterbrochener Reihenfolge wiederholt werden muß, um zum gewünschten Ergebnis zu kommen. Für solche Situationen können Sie eine Schleife programmieren. Schleifenbildung ist eine Programmiertechnik, bei der der Rechner angewiesen wird, eine Befehlsfolge kontinuierlich zu wiederholen, bis die geforderte Aufgabe erfüllt ist. Um eine Schleife zu programmieren, bauen Sie einfach einen Befehl in das Programm ein, der den Programmzeiger immer wieder auf einen früheren Speicherplatz einstellt.

UNBEDINGTE SCHLEIFENBILDUNG

Es gibt zwei Methoden der unbedingten Schleifenbildung:

Mit **RST** verzweigt das Programm immer wieder auf Speicherplatz 000.

Mit **GTO** verzweigt das Programm immer wieder zur angegebenen Adresse.

Erstellen Sie ein Programm, das in Viererabständen zählen kann. Die einfache Folge

+ 4 = 2nd Pause RST

müßte diesen Zweck erfüllen, wenn man sie an den Anfang des Programmspeichers platziert. Nach Eingabe dieser Folge in den Programmspeicher stellen Sie den Learn-Modus ab, gehen auf Speicherplatz 000 zurück, geben eine Anfangszahl ein und drücken **R/S** und beobachten die Zählung. Wenn Sie die Folge ab Speicherplatz 020 in den Programmspeicher einbringen wollen, können Sie **RST** durch **GTO 020** ersetzen und dieselbe Wirkung erzielen. Bedenken Sie nur, daß Sie zu Beginn den Ablauf bei Speicherplatz 020 starten müssen.

Vorsicht mit **RST**, weil dieser Befehl auch alle Flags rücksetzt und das Unterprogrammrücksprungs-Register löscht.

Um die Schleife zu verlassen, programmiert man einen Sprungbefehl in die unbedingte Schleife, der unter bestimmten, von Ihnen angegebenen Bedingungen auf einen Speicherplatz außerhalb der Schleife verzweigt. Führen Sie noch einmal das Zählbeispiel oben aus, beginnen Sie dabei mit 0 und unterbrechen Sie bei 20.



Speicherplatz und Tastenkod	Tastensequenz	Bemerkungen
000 02	2	
001 00	0	
002 32	x=t	20 wird im T-Register gespeichert
003 25	CLR	Löschen der Anzeige
004 76	2nd LbI	
005 85	+	Kennzeichnung dieses Segments mit Label +
006 85	+	
007 04	4	
008 95	=	
009 66	2nd Pause	Anzeige jeder Zählung
010 67	2nd x=t	Vergleich des berechneten Wertes mit dem Wert im T-Register
011 00	1	
012 15	5	Sprung zum Speicherplatz 15, wenn x = 20
013 61	GTO	Andernfalls zurück zu Label +
014 85	+	
015 91	R/S	Stop, wenn x = t

Sobald das Programm in den Programmspeicher eingebracht ist, drücken Sie nur **RST** **R/S**, um es ablaufen zu lassen. Beachten Sie, daß mit dem bedingten Test im Speicherplatz 010 jede durchgehende Zahl geprüft und die Schleife so lange durchlaufen wird, bis die Zählung 20 erreicht. Dann verzweigt das Programm zu 015 und hält. Die Schleife wird mit dem **GTO** – Befehl gebildet.



BEDINGTE SCHLEIFENBILDUNG

Das Zählbeispiel kann auch völlig durch einen bedingten Verzweigungsbefehl gesteuert werden. Wieder soll die Zählung von 0 bis 20 gehen.

Speicherplatz und Tastenkod	Tastenfolge	Bemerkungen
000 02	2	
001 00	0	
002 32	\neq	
003 25	CLR	
004 76	2nd Lbl	Kennzeichnung des Programnteils mit Label A
005 11	A	
006 85	+	
007 04	4	
008 95	=	
009 66	2nd Pause	
010 22	INV	
011 77	2nd \neq	Umkehrung des Tests und Verzweigung zu A, wenn der zuletzt berechnete Wert kleiner als 20 ist
012 11	A	
013 91	R/S	

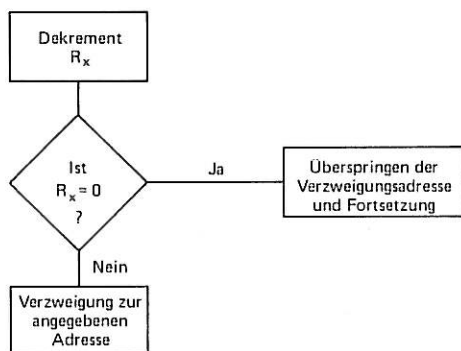
Hier wird die Schleife durch INV 2nd \neq gesteuert.



SCHLEIFENBILDUNG MIT DEM BEDINGTEN VERZWEIGUNGSBEFEHL DSZ

Wenn Sie wissen, wie oft sich eine Folge wiederholen muß, können Sie den DSZ-Befehl (Dekrement und Überspringen bei Null) zur Schleifenbildung anwenden. Die Tastenfolge hierzu ist **2nd** **DSZ** **X**, gefolgt von einer Verzweigungsadresse. X ist die Nummer eines der 10 Datenregister von 0 bis 9, die in Verbindung mit diesem Befehl belegt werden können.

Diese vielseitige Verzweigung vermindert die Größe des Inhalts von Datenregistern X um 1 (bei einem Datenregister kleiner als 1 wird auf 0 erhöht), und testet dann den Inhalt von Register X. (Für die folgende Erklärung gilt die Bezeichnung: R_x = Inhalt von Datenregister X.) Ist $R_x = 0$, wird die Verzweigungsadresse übersprungen. Andernfalls verzweigt der Verarbeitungsfluß zur angegebenen Adresse. DSZ vermindert Register X und überspringt die Verzweigung bei Null. Nachstehend die Funktion dieser Befehlsfolge in graphischer Darstellung.



Wie die anderen Verzweigungsbefehle kann man DSZ manuell über die Tastatur ebenso wie als Bestandteil eines Programms anwenden. Tasten Sie folgendes ein und beobachten Sie die Wirkung;

Taste	Anzeige	Bemerkungen
2nd CP	0.	Löschen des Programmspeichers
2 STO 6	2.	2 wird im Datenregister 06 gespeichert
2nd DSZ 6 136		Verminderung von R_6 um 1, dann die Frage: „Ist $R_6 = 0$?“ Wenn nein, Verzweigung
LRN	136 00	Die Verzweigung zu 136 fand statt
LRN RCL 6	1.	R_6 war 2 und ist jetzt wegen des DSZ-Befehls 1.
2nd DSZ 6 111	1.	Dekrement und erneuter Test
LRN	136 00	Keine Verzweigung, weil R_6 jetzt gleich Null ist
LRN RCL 06	0.	R_6 ist tatsächlich Null

DSZ ist nichts anderes als ein wirksames Zählsystem, das beim Zurückzählen bis Null Schleifen bildet, und dann zu einem anderen Befehl übergeht.

Um die Vorteile im Programm zu erkennen, sei noch einmal auf das Zählbeispiel verwiesen. Sie können sehen, daß man für die Zählung in Viererabständen bis 20 die (+4)-schleife fünfmal durchlaufen muß.



Speicherplatz und Tastenkod	Tastenfolge	Bemerkungen
000 47	2nd CMs	Löschen aller Datenspeicher
001 05	5	
002 48	2nd ErC	5 wird im Register 00 gespeichert und die Anzeige gelöscht
003 00	0	
004 76	2nd Lbl	Kennzeichnung dieses Teils mit Label A
005 11	A	
006 85	+	
007 04	4	
008 95	=	
009 66	2nd Pause	Anzeige jeder Zählung
010 97	2nd DSZ	Verminderung von 00 und Test, ob R_0 gleich Null
011 00	0	
012 11	A	Ist R_0 nicht 0, Verzweigung zu A
013 91	R/S	Stop, wenn $R_0 = 0$

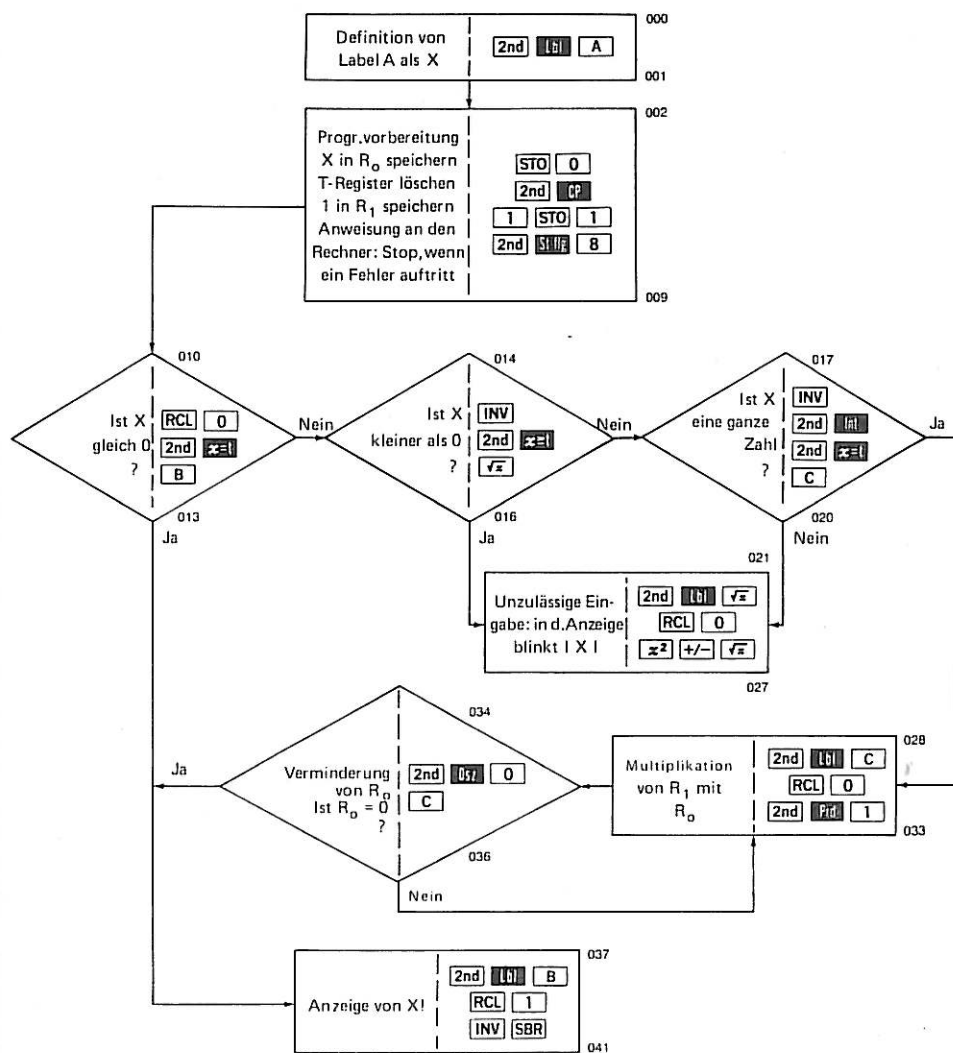
DSZ kann ebenso von der negativen Seite der Zahlengeraden her R_x erhöhen (jeweils 1 zu R_x addieren). Im obigen Beispiel hätte man also auch -5 verwenden können. Auch **INV** **2nd** **DSZ** erhöht und vermindert auf gleiche Weise, die Verzweigungsadresse wird aber jetzt bei Nicht-Null anstatt bei Null übersprungen.

Mehr Einzelheiten siehe „Dekrement und Überspringen bei Null“ auf Seite V-63.

Dieser Befehl ist auch von Vorteil, wenn man mit einer Zahlenreihe von 1 bis N rechnen muß. Man kann mit DSZ diese Zahlenreihe durch Schleifenbildung verarbeiten. Dabei wird der Ausdruck für die verschiedenen Variablenwerte berechnet und der Rechner angewiesen, den Inhalt des Datenregisters aufzurufen, das für jede Anwendung der Variablen um 1 vermindert wird. (Beachten Sie, daß die Reihe in Wirklichkeit von N bis 1 berechnet wird, weil DSZ vermindert.)

XI-PROGRAMM

Als Übung zum Prinzip der DSZ-Schleifen soll ein Programm zur Berechnung der Fakultäten $X!$ geschrieben werden, $X! = x \cdot (x-1) \cdot (x-2) \cdot \dots \cdot 2 \cdot 1$. Entsprechend der Definition dieser Funktion muß X eine positive ganze Zahl sein und $0! = 1$.



X!-Programm



IV

Speicherplatz und Tastenkod

000 76	2nd [Lb]
001 11	A
002 42	STO
003 00	0
004 29	2nd [CP]
005 01	1
006 42	STO
007 01	1
008 86	2nd [STO]
009 08	8
010 43	RCL
011 00	0
012 67	2nd [x=]
013 12	B
014 22	INV
015 77	2nd [x=]
016 34	[√]
017 22	INV
018 59	2nd [int]
019 67	2nd [x=]
020 13	C

Speicherplatz und Tastenkod

021 76	2nd [Lb]
022 34	[√]
023 43	RCL
024 00	0
025 33	[x²]
026 94	[+/-]
027 34	[√]
028 76	2nd [Lb]
029 13	C
030 43	RCL
031 00	0
032 49	2nd [Prd]
033 01	1
034 97	2nd [0sz]
035 00	0
036 13	C
037 76	2nd [Lb]
038 12	B
039 43	RCL
040 01	1
041 92	INV [SBR]

XI-Programm

Im Musterprogramm wird 1 in R_1 gespeichert, um Multiplikation durch Speicherarithmetik zu ermöglichen. Um die Programmierung vollständig zu machen, sind die ersten drei bedingten Verzweigungen eingeschlossen, um unzulässige Eingaben abzufangen. Beachten Sie: Im Falle einer unzulässigen Eingabe unterbricht die im Speicherplatz 027 programmierte Fehlerbedingung das Programm, weil an früherer Stelle Flag 8 gesetzt wurde. Die tatsächliche Schleife befindet sich zwischen den Speicherplätzen 028 bis 036.



Die Anwendung dieses Programms ist sehr einfach. Sie geben einen x-Wert kleiner als 70 ein und drücken **A**. (70! überschreitet die Rechenkapazität Ihres Geräts.)

Beispiel: Berechnen Sie: 6!; -2!; 0!; 7.3!; 39!

Eingabe	Taste	Anzeige	Bemerkungen
6	A	720	6!
2	+/- A	"2"	Unzulässige Eingabe
	CLR	0	Löschen der Fehlerbedingung
0	A	1	0!
7.3	A	"7.3"	Unzulässige Eingabe
	CLR	0	Löschen der Fehlerbedingung
39	A	2.0397882 46	39!

ANMERKUNG: Die Anführungszeichen in der Spalte Anzeige geben an, wenn der Wert in der Anzeige blinkt.

Mehr über die Anwendungsgebiete

KURSWERT VON FESTVERZINSLICHEN WERTPAPIEREN – PROGRAMM

Viele Anleger halten festverzinsliche Wertpapiere für eine sichere und gewinnträchtige Kapitalanlage. Andere würden sich für den Kauf von Zinspapieren entscheiden, wenn sie die tatsächlichen Gewinne aus den Kapitalanlagen analysieren könnten. Entwickeln Sie ein Programm zur Berechnung des Kurswertes (Kaufpreis) eines Zinspapiers mit periodischen Kupons. Hierzu verwendet man eine Formel, nach der sich der Kaufpreis des Zinspapiers aus der Summe der Diskontwerte der Kupons und des Fälligkeitswertes ergibt.

$$PV = I \sum_{j=1}^N (1 + YLD)^{-j} + MV (1 + YLD)^{-N}$$

wobei:

MV = Fälligkeitswert

N = Anzahl der Perioden bis zur Fälligkeit (j = 1, 2, ... N)

I = Kuponwert (Zins pro Periode)

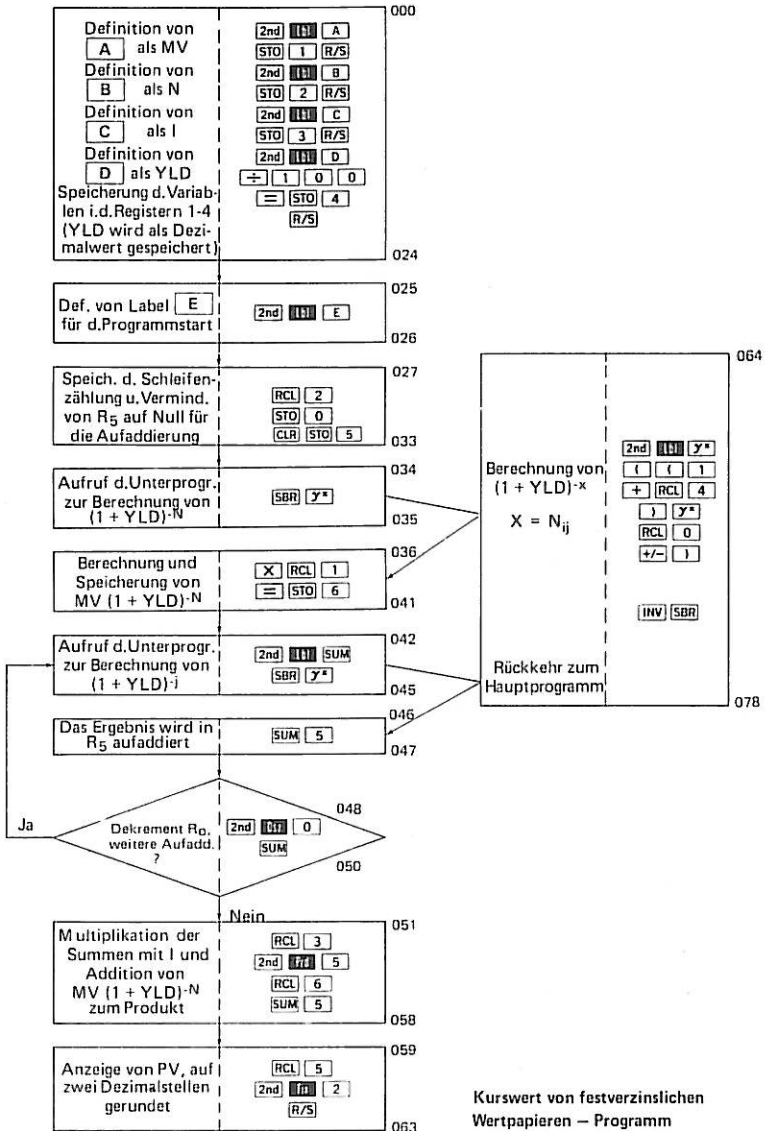
YLD = Zinsertrag bei Fälligkeit

PV = Kaufpreis des Wertpapiers oder Kurswert

Man kann das Programm so schreiben, daß zum Abschluß der Addition eine Schleife verwendet wird. Da Sie bereits im voraus wissen, wie oft die Schleife zu durchlaufen ist, bietet sich der DSZ-Befehl als das zweckmäßigste Mittel der Schleifenprogrammierung an. Insbesondere auch deshalb, weil das Datenregister, dessen Inhalt vermindert werden muß, auch den Wert für j liefert. Schließlich kann man mit einem Unterprogramm zur Berechnung von $(1 + YLD)^x$, $x = j, N$ Programmschritte sparen.



IV





PROGRAMM-INSTRUKTIONEN				
Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Löschen des Programmspeichers und Rückstellen des Programmspeichers		2nd CP	
2	Eingabe des Learn-Modus		LRN	000 00
3	Eingabe des Wertpapierprogramms			
4	Aufheben des Learn-Modus		LRN	0
5	Eingabe des Fälligkeitswertes	MV	A	MV
6	Eingabe der Anzahl der Perioden	N	B	N
7	Eingabe des Kuponwerts	I	C	I
8	Eingabe des Zinsertrags bei Fälligkeit	YLD	D	YLD/100
9	Berechnung des Kurswertes		E	PV
	Variable, die sich in einer neuen Aufgabe nicht ändern, müssen nicht erneut eingegeben werden			



IV

Speicherplatz und Tastenkod

000 76	2nd Lbl
001 11	A
002 42	STO
003 01	1
004 91	R/S
005 76	2nd Lbl
006 12	B
007 42	STO
008 02	2
009 91	R/S
010 76	2nd Lbl
011 13	C
012 42	STO
013 03	3
014 91	R/S
015 76	2nd Lbl
016 14	D
017 55	÷
018 01	1
019 00	0
020 00	0
021 95	=
022 42	STO
023 04	4
024 91	R/S
025 76	2nd Lbl
026 15	E

Speicherplatz und Tastenkod

027 43	RCL
028 02	2
029 42	STO
030 00	0
031 25	CLR
032 42	STO
033 05	5
034 71	SBR
035 45	y^x
036 65	X
037 43	RCL
038 01	1
039 95	=
040 42	STO
041 06	6
042 76	2nd Lbl
043 44	SUM
044 71	SBR
045 45	y^x
046 44	SUM
047 05	5
048 97	2nd DSZ
049 00	0
050 44	SUM
051 43	RCL
052 03	3
053 49	2nd Prd

Speicherplatz und Tastenkod

054 05	5
055 43	RCL
056 06	6
057 44	SUM
058 05	5
059 43	RCL
060 05	5
061 58	2nd fir
062 02	2
063 91	R/S
064 76	2nd Lbl
065 45	y^x
066 53	(
067 53	(
068 01	1
069 85	+
070 43	RCL
071 04	4
072 54)
073 45	y^x
074 43	RCL
075 00	0
076 94	+/-
077 54)
078 92	INV SBR

Kurswert von festverzinslichen Wertpapieren — Programm



Beispiel: Berechnen Sie den Kaufpreis eines Wertpapiers, das in 12 Jahren mit 20.000 DM fällig wird und dessen jährlicher Kuponwert 1.400 DM beträgt, wenn man einen gewünschten Zinsertrag von 8% ansetzt.

Eingabe	Taste	Anzeige	Bemerkungen
20000	A	20000	MV
12	B	12	N
1400	C	1400	I
8	D	.08	YLD
	E	18492.78	→ PV

Ein Kaufpreis von 18.492.78 bringt unter diesen Bedingungen einen jährlichen Zinsertrag von 8%. Der Gesamtgewinn einer solchen Anlage ist $12 \times 1.400.00 + (20.000.00 - 18.492.78) = 18.307.22$

PROGRAMM ZUR LÖSUNG VON QUADRATISCHEN GLEICHUNGEN

Ein besonders anschauliches Beispiel für einige der Programmiertechniken ist das Programm zur Lösung quadratischer Gleichungen. Und wenn Sie selbst in Ihren Problemlösungssituationen mit Gleichungen zweiten Grades befaßt sind, paßt es gut.

Schreiben Sie ein Programm zur Berechnung der reellen und komplexen Wurzeln der Gleichung

$$ax^2 + bx + c = 0 \quad (a \neq 0)$$

Die Wurzeln x_1 und x_2 werden wie folgt ermittelt:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Ist der Wert von $b^2 - 4ac$ positiv oder gleich Null, sind die Wurzeln reell und werden nach den obigen Gleichungen berechnet. Ist jedoch $b^2 - 4ac$ negativ, sind die Wurzeln x_1 und x_2 komplex und müssen, wie unten dargestellt, in ihre Real- und Imaginärteile zerlegt werden.

$$x_1 = R + (i \cdot I)$$

$$x_2 = R - (i \cdot I)$$

wobei:

$$R = -b/2a$$

$$i = \sqrt{-1}$$

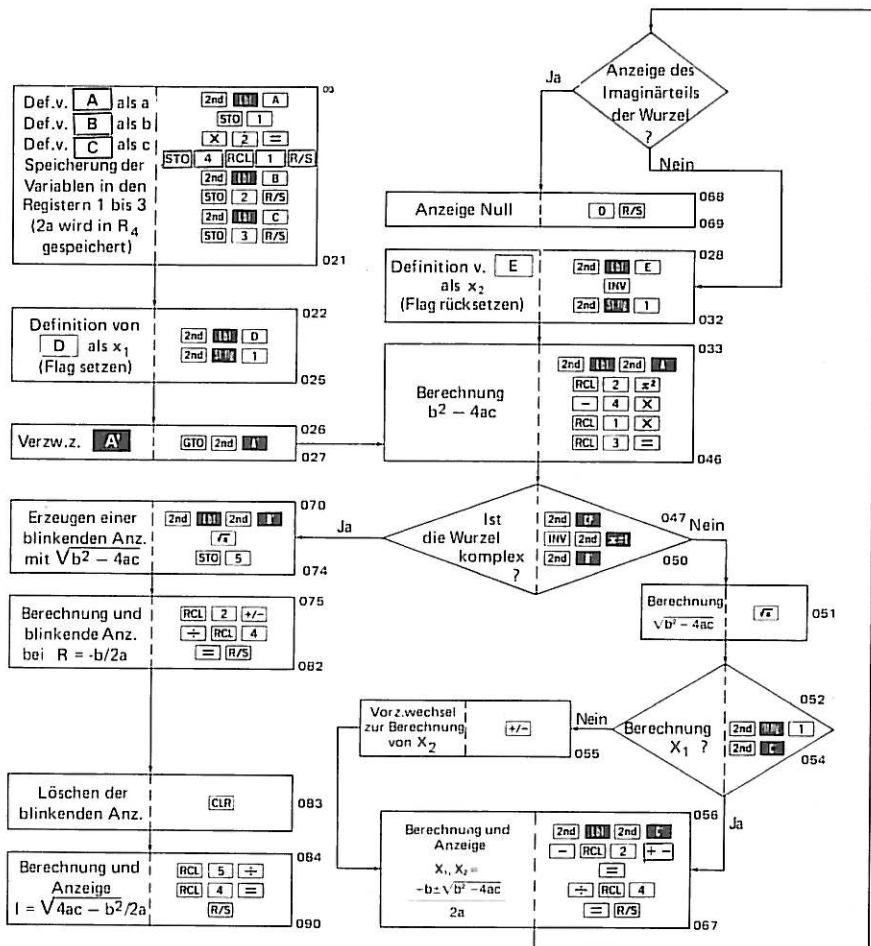
$$I = \sqrt{4ac - b^2}/2a$$

Da x_1 und x_2 mit denselben Grundgleichungen berechnet werden, kann man auf folgende Weise Programmschritte einsparen: Man kombiniert diesen Programmteil und setzt ein Flag, um aufzuzeigen, welche Wurzel berechnet werden muß. Ein separater Programmteil ist für die Zerlegung der komplexen Wurzeln in ihre Real- und Imaginärteile erforderlich. Ob die Wurzeln komplex oder reell sind, kann durch einen Test festgestellt werden, bei dem geprüft wird, ob $b^2 - 4ac$ negativ ist. Beachten Sie, wenn die Wurzeln komplex sind, müssen Sie x_2 nicht berechnen, weil die Werte von R und I für beide Wurzeln gleich sind.



IV

Sie sollten ebenfalls eine Anzeigemöglichkeit finden, ob die Wurzeln reell oder komplex sind. Da bei einer komplexen Wurzel $b^2 - 4ac$ negativ ist, kann man eine blinkende Anzeige erzeugen, wenn man vor Berechnung des Realteils der Wurzel die Quadratwurzel für diesen Wert ermittelt. Beachten Sie, daß tatsächlich der Ausdruck $4ac - b^2$ berechnet wird, und nicht $b^2 - 4ac$, wenn $b^2 - 4ac$ negativ ist. Sie können dieses Ergebnis speichern und später bei der Bestimmung des Imaginärteils der Wurzeln wieder verwenden. Im Musterprogramm unten wird der Imaginärteil der Wurzel über die Taste $\boxed{R/S}$ nach Berechnung des Realteils bestimmt. Als Sicherheitsmaßnahme wird eine Null angezeigt, wenn die Wurzel keinen Imaginärteil besitzt. Dieses Programm eignet sich nicht als Unterprogramm, weil es auch $\boxed{=}$ - und $\boxed{R/S}$ -Befehle enthält.



Programm zur Lösung von quadratischen Gleichungen



PROGRAMM-INSTRUKTIONEN

Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Löschen des Programmspeichers und Rückstellung des Programmzeigers		2nd CP	
2	Eingabe des Learn-Modus		LRN	000 00
3	Eingabe des Programms: Quadratische Gleichung			
4	Aufheben des Learn-Modus		LRN	0
5	Eingabe von a ($a \neq 0$)	a	A	a
6	Eingabe von b	b	B	b
7	Eingabe von c	c	C	c
8	Berechnung von x_1 Blinkt der Realteil in der Anzeige – ist die Wurzel komplex – – Berechnung des Imaginärteils		D R/S	x_1 (Realteil) x_1 (Imaginärteil)
9	Berechnung von x_2 Blinkt der Realteil in der Anzeige – ist die Wurzel komplex – Berechnung des Imaginärteils		E R/S	x_2 (Realteil) x_2 (Imaginärteil)



IV

Speicherplatz und Tastenkod

000 76	2nd Lbl
001 11	A
002 42	STO
003 01	1
004 65	X
005 02	2
006 95	=
007 42	STO
008 04	4
009 43	RCL
010 01	1
011 91	R/S
012 76	2nd Lbl
013 12	B
014 42	STO
015 02	2
016 91	R/S
017 76	2nd Lbl
018 13	C
019 42	STO
020 03	3
021 91	R/S
022 76	2nd Lbl
023 14	D
024 86	2nd St I/p
025 01	1
026 61	GTO
027 16	2nd A'
028 76	2nd Lbl
029 15	E
030 22	INV
031 86	2nd St I/p

Speicherplatz und Tastenkod

032 01	1
033 76	2nd Lbl
034 16	2nd A'
035 43	RCL
036 02	2
037 33	x²
038 75	-
039 04	4
040 65	X
041 43	RCL
042 01	1
043 65	X
044 43	RCL
045 03	3
046 95	=
047 29	2nd CP
048 22	INV
049 77	2nd x² I
050 17	2nd B'
051 34	√x
052 87	2nd II I/g
053 01	1
054 18	2nd C'
055 94	+/-
056 76	2nd Lbl
057 18	2nd C'
058 85	-
059 43	RCL
060 02	2
061 94	+/-

Speicherplatz und Tastenkod

062 95	=
063 55	÷
064 43	RCL
065 04	4
066 95	=
067 91	R/S
068 00	0
069 91	R/S
070 76	2nd Lbl
071 17	2nd B'
072 34	√x
073 42	STO
074 05	5
075 43	RCL
076 02	2
077 94	+/-
078 55	÷
079 43	RCL
080 04	4
081 95	=
082 91	R/S
083 25	CLR
084 43	RCL
085 05	5
086 55	÷
087 43	RCL
088 04	4
089 95	=
090 91	R/S

Programm zur Lösung von quadratischen Gleichungen



Beispiel: Berechnen Sie die Wurzeln der Gleichung:

$$1.5x^2 + 3.7x + 2.25 = 0$$

Eingabe	Taste	Anzeige	Bemerkungen
1.5	<input type="button" value="A"/>	1.5	a
3.7	<input type="button" value="B"/>	3.7	b
2.25	<input type="button" value="C"/>	2.25	c
	<input type="button" value="D"/>	- 1.088036702	Berechnung von x_1 (die unbewegliche Anzeige weist auf eine reelle Wurzel hin)
	<input type="button" value="E"/>	- 1.378629965	Berechnung von x_2

Berechnen Sie die Wurzel der Gleichung:

$$x^2 + 2x + 17 = 0.$$

Eingabe	Press	Anzeige	Bemerkungen
1	<input type="button" value="A"/>	1.	a
2	<input type="button" value="B"/>	2.	b
17	<input type="button" value="C"/>	17.	c
	<input type="button" value="D"/>	"-1."	Berechnung der Wurzel (die blinkende Anzeige bedeutet daß die Wurzeln komplex sind - R wird angezeigt)
	<input type="button" value="R/S"/>	4.	Berechnung des Imaginärteils



WEITERE PROGRAMMIERTECHNIKEN

Programmierung von indirekten Befehlen

Mit dem indirekten Befehl **2nd** **Ind** kann man Datenspeicheroperationen, Verzweigungsfolgen, spezielle Steueroperationen und die Softwareprogramm-Adressierung um eine ganze Reihe vorteilhafter Möglichkeiten erweitern. Das Prinzip ist einfach: Sie gehen zu einem Datenregister, nicht, um dort die benötigte Information aufzufinden, sondern um zu erfahren, wo Sie diese Information erhalten können. Es ist, als ob Sie jemanden beauftragen: „Geh’ und frage Sam, wo Fred ist“, anstatt zu sagen: „Geh’ und finde Fred“. Sie sehen, wenn Sam weiß, wo Fred ist, erfährt der Fragesteller sofort Freds derzeitigen Aufenthaltsort. Dagegen kann es Stunden dauern, wenn sich jemand auf den Weg macht, um Fred zu finden. Übertragen auf die Programmierung, ist es manchmal wesentlich einfacher, sich auf indirektem Weg Informationen zu verschaffen. Tatsächlich können Situationen eintreten, wo Fred nicht direkt gefunden werden kann, so daß die indirekte Methode die einzige Möglichkeit ist. Den Befehlen in der indirekten Form stellt man **2nd** **Ind** voran und läßt eine Datenregisternummer folgen. Dieses Datenregister enthält die erforderliche Information, um den Befehl abzuschließen.

INDIREKTER ZUGRIFF AUF DATENREGISTER

Alle Datenregisterbefehle (Speichern, Aufruf, Austausch, Speicheraddition, Speichermultiplikation) können die indirekte Adressierung verwenden. Betrachten Sie die Folge





IV

Schreiben Sie nun ein Programmsegment zur Löschung einer Reihe von Datenregistern. Zur Vereinfachung löschen Sie das Register 1 durch X, wobei X variieren kann.

Speicherplatz und Tastenkod	Tastenfolge	Bemerkungen
000 76	2nd Lbl	Eingabe von X und Taste A drücken
001 11	A	
002 42	STO	X wird im Datenregister 00 gespeichert
003 00	0	
004 76	2nd Lbl	
005 12	B	
006 25	CLR	
007 72	STO 2nd Ind	Null muß in dem Register gespeichert werden, das in 00 angegeben ist
008 00	0	
009 97	2nd DSZ	DSZ-Schleife für Register 00
010 00	0	
011 12	B	Sprung zu B, wenn $R_{00} \neq 0$
012 92	INV SBR	Programmstop, wenn Register 00 Null erreicht hat

Beim ersten Durchlaufen der Schleife ist X im Register 00, so daß die Folge **CLR** **STO** **2nd** **Ind** 00 in Register X eine Null speichert. DSZ vermindert dann Register 00 auf $(X - 1)$. Mit der indirekten Speicherfolge wird jetzt 0 im Register $(X - 1)$ gespeichert, etc. Die Register wurden in umgekehrter Reihenfolge auf Null gestellt, was aber nicht stören dürfte. Können Sie ein Programm schreiben, wo sie in ihrer numerischen Reihenfolge gelöscht werden?

Beachten Sie hier den speziellen Tastenkod 72 für die Folge **STO** **2nd** **Ind**. Mehrere der indirekten Befehle sind wie hier kombiniert und sparen Programmspeicherplätze ein. Eine vollständige Liste siehe „Befehlskodes (Tastenkodes)“ auf Seite V-48.



INDIREKTE VERZWEIGUNGSBEHELFE

Die Zweckmäßigkeit der indirekten Adressierung läßt sich auf die Programmverzweigungen ausdehnen. Sie erinnern sich, daß es zwei Möglichkeiten gibt, eine Verzweigungsadresse anzugeben: die absolute Adresse oder das Label im Programmspeicher. Die indirekte Adressierung ermöglicht eine andere, flexiblere Methode. Sie geben die Datenregisteradresse an, in der die gewünschte absolute Adresse aufgefunden wird. Eine Labeladresse kann nicht im Datenregister gespeichert werden.

Bei indirekten Verzweigungsfolgen stellt man dem unbedingten Verzweigungsbefehl (**GTO**, **SBR**) oder dem bedingten Verzweigungsbefehl (**2nd** **cc=1**, **2nd** **DSZ**, etc.) die Anweisung **2nd** **Ind** nach. Die Folge muß dann mit der Datenregisteradresse abgeschlossen werden, die die absolute Adresse des Programmspeicherplatzes enthält, zu dem Sie verzweigen wollen. Versuchen Sie diese Folge über die Tastatur.

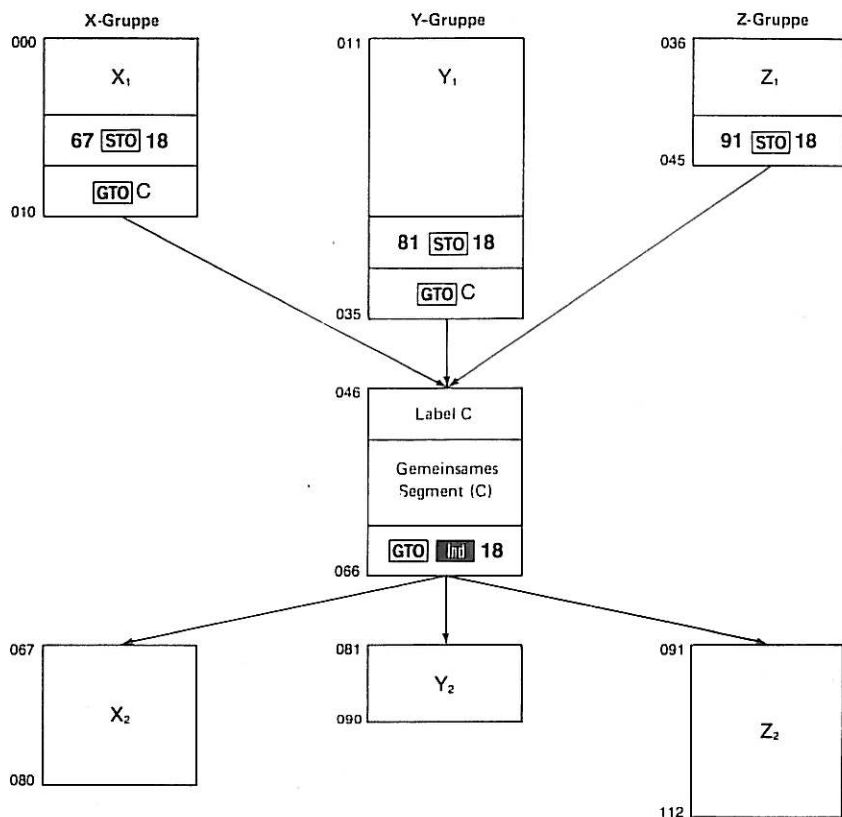
Tastenfolge	Anzeige	Bemerkungen
35 STO 18	35.	35 wird im Datenregister 18 gespeichert
GTO 2nd Ind 18 LRN 035 00		Der Programmzeiger wurde auf Speicherplatz 035 eingestellt

Auch der DSZ-Befehl kann indirekt das Register angeben, dessen Inhalt vermindert werden muß. Das heißt, bei Anwendung dieses Befehls sind Folgen wie **2nd** **DSZ** **2nd** **Ind** **12** **2nd** **Ind** **14** möglich. Sowohl das beim DSZ-Befehl benutzte Datenregister als auch die Verzweigungsadresse kann man indirekt erhalten.

Die folgende graphische Darstellung soll diese Verzweigungsmethode und ihre Anwendung verdeutlichen. Nehmen Sie an, daß drei separate Befehlsgruppen, wie unten angegeben, in ein Programm eingefügt werden müssen.

Gruppe X	Gruppe Y	Gruppe Z
X ₁	Y ₁	Z ₁
C	C	C
X ₂	Y ₂	Z ₂

Der zentrale Teil (C) ist in jeder Befehlsgruppe gleich, es liegt also nahe, diesen gemeinsamen Teil nur einmal zu schreiben. Es ist eine einfache Angelegenheit, am Ende der Segmente X₁ und Y₁ **GTO**-Befehle zu verwenden, um zu C zu gelangen (Z₁ geht direkt auf C), aber wie verzweigt das Programm korrekt zu X₂, Y₂ und Z₂? Das Problem kann mit indirekter Adressierung gelöst werden. Speichern Sie einfach die Adresse des dritten Abschnitts vor der Verzweigung zu C und schließen Sie C mit einem **GTO** **2nd** **Ind** -Befehl ab. Im Diagramm sind zur Erklärung die Speicherplätze am Anfang und am Ende jedes Segments willkürlich gewählt.



ANDERE EIGENSCHAFTEN

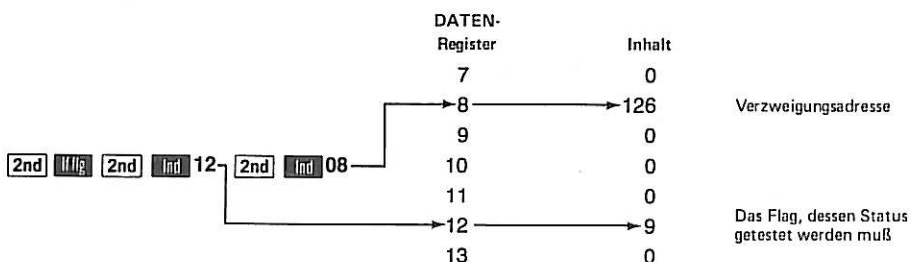
Mit **2nd** **Ind** können Programmflags indirekt gesetzt und rückgesetzt werden. Auch eine Kontrolle der Festkomma-Einstellung des Rechners ist mit diesem Befehl möglich.

Wiederum erfolgt die indirekte Flagsteuerung durch Speichern der Flagnummer in einem Datenregister. Speichert man zum Beispiel 6 in R_{12} und schließt die Folge mit **2nd** **St/Flg** **2nd** **Ind** 12 ab, wird tatsächlich Flag 6 gesetzt, während **2nd** **St/Flg** **2nd** **Ind** 12 **A** abhängig vom Status von Flag 6 zu Label **A** verzweigt.



IV

Da **2nd** **iflg** ein Verzweigungsbefehl ist, kann auch die Verzweigung indirekt ausgeführt werden. Folgen wie **2nd** **iflg** **2nd** **ind** 12 **2nd** **ind** 08 sind deshalb möglich.



Ein äquivalenter Befehl ist **2nd** **iflg** 9 126. Geben Sie dieses Beispiel in den Rechner ein und beobachten Sie die Wirkung.

Taste	Anzeige	Bemerkungen
126 STO 08	126.	126 wird in Register 08 gespeichert
9 STO 12	9.	9 wird in Register 12 gespeichert
2nd iflg 9	9.	Flag 9 wird gesetzt
2nd iflg 2nd ind 12		
2nd ind 8 LRN	126 00	Verzweigung zum Speicherplatz 126

Auf gleiche Weise kann man die Festkomma-Einstellung indirekt steuern, wie das folgende Beispiel zeigt.

Taste	Anzeige	Bemerkungen
2 STO 12	2.	2 wird in Register 12 gespeichert
2nd fix 2nd ind 12	2.00	Festkomma-Einstellung des Rechners auf 2 Dezimalstellen

Mehr über indirekte Adressierung siehe „INDIREKTE ADRESSIERUNG“ auf Seite V-68.



Programmoptimierung

Von den vielen Gründen, ein Programm zu optimieren, haben zwei besondere Bedeutung. Erstens kann man die Anwendung des Programms vereinfachen und zweitens das Programm so straffen, daß es für die gewählte Speicherbereichsverteilung geeignet ist.

PROGRAMMIERTECHNIKEN FÜR VEREINFACHTE ANWENDUNG

Ob man ein Programm einfach und problemlos anwenden kann, ist eine Frage des persönlichen Bedarfs und der eigenen Prioritäten. Generell gilt, daß ein gut geschriebenes Programm mit nur wenigen Tastenbetätigungen ohne Schwierigkeiten durchgeführt werden kann (und nicht nur vom Programmierer).

Bei vielen Programmen muß im Falle einer falschen Eingabe oder eines falschen Tastenbefehls das gesamte Problem neu gestartet werden, eine vielleicht lästige und zeitraubende Arbeit, besonders bei langen und komplizierten Programmen. Die Vereinfachung der Fehlersuche und Korrektur ist eine Möglichkeit, die Anwendung eines Programms zu erleichtern. Im allgemeinen erfolgen diese Verfahren durch Speichern und Bewahren der ursprünglichen Daten. Eine empfehlenswerte Gewohnheit ist außerdem, Programme, die Speicherarithmetik ausführen, mit einem **STO**-Befehl zu beginnen, da die Routine ohne Löschen von Datenregistern wiederholt werden kann.

PROGRAMMIERTECHNIKEN ZUR REDUZIERUNG VON SCHRITTEN

Die Straffung eines Programms auf eine geringere Anzahl von Programmschritten ist eine zeitraubende Übung. Sofern ein Programm mit der Speicherbereichsverteilung vereinbar ist und richtig funktioniert, ist jede weitere Straffung unnötige Zeitverschwendung, wenn man von der persönlichen Genugtuung absieht.

Beim Versuch, die Anzahl der Programmschritte zu reduzieren, sollte man die Aufmerksamkeit zunächst auf die Folgen richten, die mehr als einmal erscheinen. Wenn diese Folgen dann lang genug sind und oft genug benötigt werden, kann man sie durch Unterprogramme ersetzen und auf diese Weise Programmspeicherraum freimachen.

Programme, die zahlreiche Unterprogramme erfordern, können immer noch die Programmspeicherkapazität überschreiten. Damit kommt der Optimierung von Unterprogrammen wesentliche Bedeutung zu.

Es gibt viele Methoden, separate Programmteile zu kombinieren, um Speicherplatz einzusparen. Wenn zum Beispiel ein Unterprogrammaufruf die letzte Operation einer anderen Routine ist, kann man das Unterprogramm in eine Linie mit dem ersten bringen.



IV

Ein Programm
wie das folgende

...

2nd Lbl

2nd t

...

Dieser Teil
wird aus dem
Programm
genommen

SBR

STO

R/S

...

2nd Lbl

STO

...

INV SBR

kann auch in dieser
Formaufgebaut sein

...

2nd Lbl

2nd t

...

2nd Lbl

STO

...

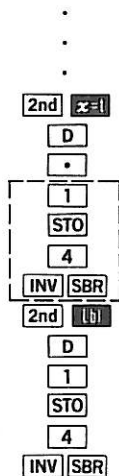
INV SBR

Dabei wurde nicht nur die Einsparung mehrerer Programmschritte realisiert, auch eine Ebene des Unterprogrammrücksprung-Registers wurde frei. **INV SBR** wirkt jetzt wie ein **R/S**-Befehl, weil das Unterprogrammrücksprung-Register nicht belegt ist.

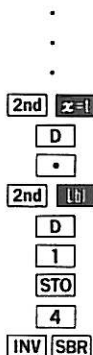


Betrachten Sie als weiteres Beispiel die beiden nächsten Folgen:

Funktionsfähiges Segment



Rationelles Segment



Hier soll . 1 oder 1 abhängig von den Testergebnissen gespeichert werden. Beide Routinen führen die gleiche Funktion aus; die zweite ist jedoch um 4 Programmschritte kürzer als die erste, weil die nicht zugehörigen Befehle (im Bild deutlich eingezeichnet) aus dem Programm eliminiert wurden.



IV

Zu den verschiedenen Techniken, separate Programmteile zu kombinieren, kommen noch zahlreiche Kniffe in der Programmierung, die für Sie von Vorteil sein können. Im nächsten Beispiel will der Programmierer nur den auf zwei Stellen gerundeten Wert der Zahl in der Anzeige für seine Berechnungen weiterverwenden. Den Rechner einfach auf zwei Festkommastellen zu schalten, ist in diesem Fall keine Lösung, weil in den meisten Berechnungen der volle ungerundete Wert verwendet werden.

Funktionsfähiges Segment

.
.
.
X
1
0
0
=
2nd fnt
÷
1
0
0
=
.
.
.

Rationelles Segment

.
.
.
2nd fnt
2
EE
INV
EE
2nd fnt
9
.
.
.

Zweck und Methode der linken Routine sind klar und direkt. Die Folgerungen, auf denen die zweite Folge basiert, sind zwar rationeller, aber nicht so leicht verständlich. Da der [EE]-Befehl nur mit den angezeigten Stellen arbeitet, löscht dieser Befehl die ungewollten Stellen nach der Festkomma-Einstellung der Anzeige. Die Routine normalisiert dann die Anzeige und setzt die Verarbeitung mit dem gerundeten Wert fort.

Die folgenden Programmteile zeigen drei Methoden für die Durchführung ein- und derselben Operation: 10 000 wird zum Anzeigeregister addiert.



.	.	.
.	.	.
.	.	.
+	+	+
1	1	4
0	EE	INV
0	4	2nd log
0	=	=
0	.	.
=	.	.
.	.	.
.	.	.

Das zweite und das dritte Programmsegment belegen die gleiche Anzahl von Programmspeicherplätzen. Die zweite Methode bietet jedoch nur dann Vorteile, wenn man die Exponentialform der Anzeige beibehalten will.

Mit zunehmendem Wissen über die Leistungsfähigkeit Ihres Rechners werden Sie selbst abgekürzte Verfahren finden, die Ihrem Bedarf entsprechen. Merken Sie sich diese Folgen auch für künftige Anwendungen, weil sie Ihre Programmieraufgaben reduzieren. Bis dahin können Sie für die Programm-Optimierung die vielen, bereits eingebauten Eigenschaften nutzen, mit denen Programmschritte eingespart werden. Zu derartigen Eigenschaften gehören Funktionen wie die Speicheroperationen **SUM** und **2nd Prd**, die indirekten Befehle und viele spezielle Steueroperationen.

Wenn Sie dann immer noch Schwierigkeiten haben, das Programm im zugewiesenen Speicherraum unterzubringen, kann es nötig werden, das Programm in Segmente aufzuteilen und Zwischenergebnisse zu berechnen, ehe Sie die Berechnung des Endresultats programmieren. Manchmal jedoch, wenn Sie versuchen, in äußerst direkter und geradliniger Form zu programmieren, gibt es eine weitere Alternative, wie im nächsten Beispiel gezeigt wird.

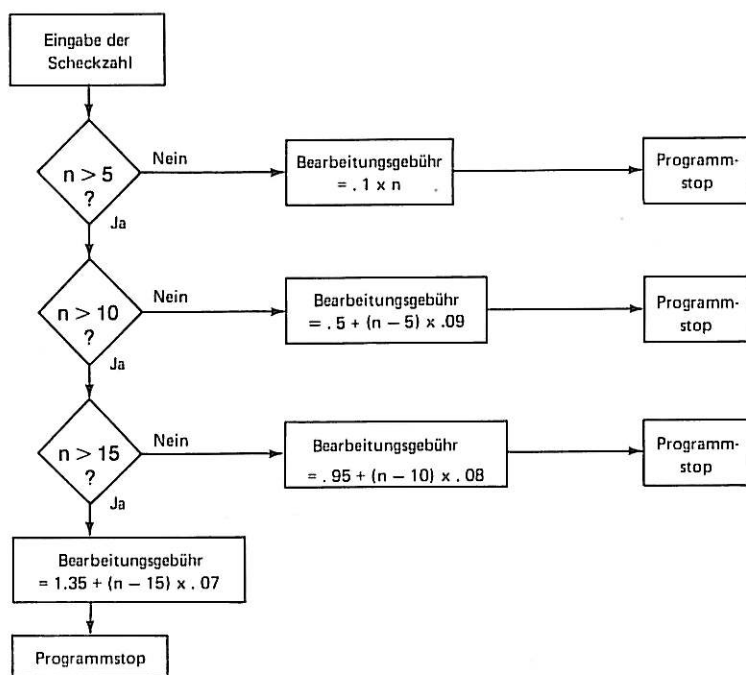
PROGRAMM ZUR BERECHNUNG VON BEARBEITUNGSGEBÜHREN

Als Leiter einer angesehenen ortsansässigen Bank brauchen Sie eine schnelle und einfache Methode, um die monatlichen Bearbeitungsgebühren für die vielen Kunden zu berechnen, die bei Ihrer Bank ein Konto unterhalten. Die Bearbeitungsgebühr für jedes Konto errechnet sich wie folgt:

- DM 0.10 pro Scheck für die ersten 5 Schecks (1 bis 5)
- DM 0.09 pro Scheck für die nächsten 5 Schecks (6 bis 10)
- DM 0.08 pro Scheck für weitere 5 (11 bis 15)
- DM 0.07 pro Scheck für jeden weiteren Scheck über 15.



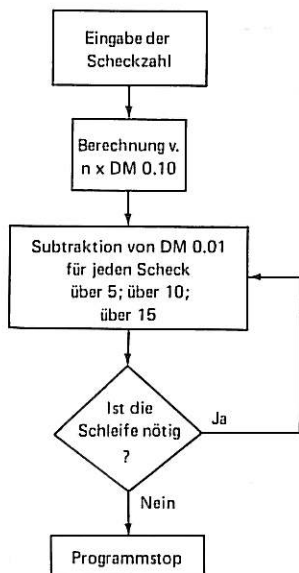
Eine sehr direkte Lösungsmethode wird im folgenden Ablaufplan aufgezeigt.



Programm zur Berechnung der Bearbeitungsgebühren (Grund-Lösung)



Wenn man ein Programm nach dieser Lösungsmethode schreiben würde, benötigt man mindestens 80 oder 90 Programmspeicherplätze. Obwohl eine solche Routine ohne Schwierigkeiten in den Programmspeicher paßt, kann der Fall eintreten, daß man Raum für andere Programnteile schaffen und dieses Programm erheblich rationalisieren muß. Vielleicht sind mit einem anderen Verfahren weniger Programmschritte erforderlich. Betrachten Sie die folgende Lösung:

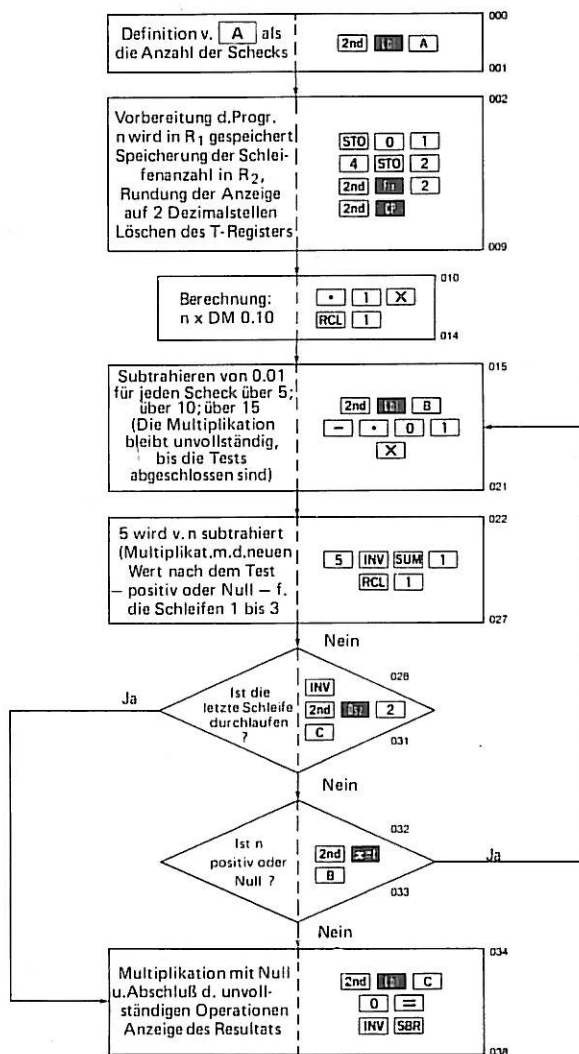


Programm zur Berechnung der Bearbeitungsgebühren
(verbesserte Lösung)

Auf den ersten Blick scheint es, daß ein Programm, das diesen Gedankengang realisiert, leicht in den Programmspeicher Ihres Rechners eingebracht werden kann. Die Argumentation, die hinter diesen Folgen steht, ist jedoch nicht sofort einleuchtend. Untersuchen Sie kurz diese Logik.



IV



Programm zur Berechnung der Bearbeitungsgebühren



Das Programm verläuft geradlinig bis zum Speicherplatz 022, wo die Multiplikation in Schritt 021 unvollständig bleibt, während für n eine Richtigstellung erfolgt und die Tests abgeschlossen werden. Die Schleife wird verwendet, um die Bearbeitungsgebühr für jeden Scheck über 5 auf DM 0.09, über 10 auf DM 0.08 und über 15 auf DM 0.07 zu reduzieren. Der `2nd` `057`-Befehl fragt, welche Schleife gerade durchlaufen wird. Für die Schleifen 1 bis 3 wird der Wert n getestet; ist er negativ, wird eine Null in die Anzeige eingebracht und die unvollständige Multiplikation abgeschlossen. Das Programm wird dann nach Berechnung der gesamten Bearbeitungsgebühren beendet.

Wenn die vierte Schleife erreicht ist, wird die unvollständige Multiplikation immer mit Null abgeschlossen, weil sich sonst die Gebühr für jeden Scheck über 20 auf 0.06 reduzieren würde. Das Programm berechnet dann die gesamten Gebühren und hält. Diese letzte Schleife ist für die Berechnung nicht notwendig. Würde man sie aber weglassen, wären weitere Programmbefehle erforderlich, und das Ziel war, die Größe des Programms auf ein Minimum zurückzuführen.

Für die Aufgabe, die Bearbeitungsgebühren zu berechnen, wurden nur zwei Lösungen angegeben. Berücksichtigt man, daß man die Lösung eines Problems auf viele verschiedene Arten programmieren kann, so zeigen diese beiden Extreme die großen Unterschiede in der Programmiertechnik. Natürlich muß man Kompromisse schließen. In diesem Beispiel braucht man für die zweite Methode weniger als die Hälfte an Programmspeicherraum, bei der ersten Lösung läuft das Programm jedoch schneller ab. Unabhängig von dem Lösungsweg, den Sie programmieren, bedenken Sie immer, daß die richtige Methode die beste für Sie ist.



Speicherplatz und Tastenkode	Tastenfolge	Speicherplatz und Tastenkode	Tastenfolge
000 76	2nd LbI	020 01	1
001 11	A	021 65	X
002 42	STO	022 05	5
003 01	0 1	023 22	INV
004 04	4	024 44	SUM
005 42	STO	025 01	1
006 02	2	026 43	RCL
007 58	2nd fix	027 01	1
008 02	2	028 22	INV
009 29	2nd CP	029 97	2nd DSZ
010 93	.	030 02	2
011 01	1	031 13	C
012 65	X	032 77	2nd $\neq 1$
013 43	RCL	033 12	B
014 01	1	034 76	2nd LbI
015 76	2nd LbI	035 13	C
016 12	B	036 00	0
017 75	-	037 95	=
018 93	.	038 92	INV SBR
019 00	0		

Programm zur Berechnung der Bearbeitungsgebühren

Um das Programm durchzuführen, geben Sie einfach die Anzahl der Schecks ein und drücken **A**. Ein Scheck kostet zum Beispiel DM 0,10, 6 Schecks kosten DM 0,59 und für 63 Schecks betragen die Gebühren DM 4,71.

Programmiertechniken für kürzere Verarbeitungszeiten

In bestimmten Fällen, wenn man Programme mit langer Ablaufzeit sehr oft durchführen muß, kann man den Verarbeitungszeitraum reduzieren. Unter diesen Voraussetzungen können verschiedene Tastenfolgen zu einer schnelleren und rationelleren Programmoperation führen.

Beim Programmlauf sind die Verzweigungen die Operationen, die die meiste Verarbeitungszeit benötigen. Verringert man also die Anzahl der Verzweigungsbefehle auf ein Minimum, läuft das Programm schneller ab. Aus diesem Grund kann man, soweit die Speicherkapazität es erlaubt, Unterprogramme durch fortlaufende Befehle ersetzen und so eine erhebliche Steigerung der Verarbeitungsgeschwindigkeit erreichen, obwohl die Anwendung von Unterprogrammen in früheren Kapiteln nachdrücklich betont wurde.



Sie erinnern sich, daß ein Programmschritt absolut mit einer 3-stelligen Adresse oder mit einem Programm-Label adressiert werden kann. Bei absoluter Adressierung stellt sich der Programmzeiger sofort auf den neuen Speicherplatz ein. Verwendet man jedoch ein Label, muß der Rechner diesen Speicherplatz erst suchen. Diese Labelsuche beginnt immer mit Speicherplatz 000 und wird durch den Programmspeicher fortgesetzt, bis der gewünschte Speicherplatz aufgefunden ist. Von dieser Stelle an wird dann der Ablauf wieder aufgenommen.

Wenn das Programm zuerst in den Rechner eingegeben wird, ist es natürlich schwierig, von vornherein die absoluten Adressen zu kennen. Auch beim Redigieren eines Programms ändern sich diese Adressen häufig, und die Arbeit wird dadurch erheblich erschwert. Das beste Verfahren ist demnach, wenn man das Originalprogramm zunächst mit Labels schreibt, und erst, nachdem es vollständig korrigiert ist, auf absolute Adressierung umstellt. Wiederum ändern sich die Adressen, wenn man die Labels entfernt und die absoluten Adressen einfügt. Dieses Problem läßt sich mit dem **2nd Nop**-Befehl jedoch ohne Schwierigkeiten lösen.

2nd Nop führt einfach keine Operation aus, wenn der Befehl in einem Programm vorkommt. Da dieser Befehl nicht auf den Ablauf wirkt (es sei denn, man verwendet ihn als Label), kann man ihn als Leerbefehl verwenden, um Zwischenräume freizulassen, wenn dies im Rahmen der verfügbaren Speicherkapazität möglich ist. Diese Technik ist unten dargestellt.

	.		.
	.		.
	.		.
027	SBR	027	SBR
028	Inx	028	0
029	2nd Nop	029	7 5
	.		.
	.		.
	.		.
073	2nd Lbl	073	2nd Nop
074	Inx	074	2nd Nop
	.		.
	.		.
	.		.
	.		.
099	GTO	099	GTO
100	Inx	100	0
101	2nd Nop	101	7 5

Umstellung von Label-Adressierung auf absolute Adressierung



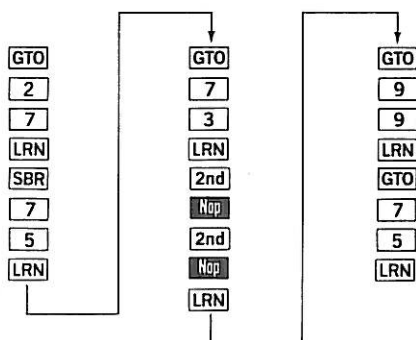
IV

Beachten Sie, daß als absolute Adresse der Speicherplatz 075 verwendet wird, da die Verzweigung zu einer Label-Adresse den Programmzeiger auf den ersten Speicherplatz nach dem Label einstellt.

Bedenken Sie außerdem, daß die absolute Adresse 075 zwei Programmspeicherplätze und nicht drei belegt. Die erste Stelle, 0, wird im Standardtastencode im ersten Platz nach dem Verzweigungsbefehl gespeichert. Die Stellen zwei und drei werden zu dem zweistelligen Kode 75 zusammengefaßt und belegen den nächsten Programmspeicherplatz.

Der Verzweigungsbefehl selbst muß ebenfalls neu eingegeben werden, um den Rechner anzuweisen, die Adresse automatisch zu kombinieren.

Verwenden Sie diese Tastenfolge, um das vorige Beispiel auf absolute Adressierung umzustellen.





Kodebrecher (Zahlenratespiel-Programm)

Dieses letzte Programmierbeispiel ist nicht für den Anfänger gedacht. Die hier verwendeten Techniken ziehen den größten Nutzen aus dem breiten Angebot der verfügbaren Programmiermöglichkeiten. Dieses Spiel macht Spaß und ist überdies sehr lehrreich unter dem Aspekt der Programmstruktur.

„Kodebrecher“ ist ein Rechnerspiel, bei dem der Rechner eine vierstellige geheime Zahl erzeugt, die Sie erraten sollen. Nullen sind nicht zulässig und keine Stelle darf mit einer anderen identisch sein. Selbst mit diesen Einschränkungen gibt es noch 3024 mögliche Kodezahlen, die Ihre Chancen, die Zahl auf Anhieb zu erraten, schwinden lassen. Ihr Versuch wird automatisch vom Rechner ausgewertet. Die Trefferzahl wird in der Form „N“ . „R“ angezeigt. N ist die Anzahl der Ziffern in Ihrem Versuch, die auch in der geheimen Zahl enthalten und in der richtigen Position sind. R ist die Anzahl der Ziffern, die zwar richtig geraten, aber in der falschen Position sind. Ist zum Beispiel die vom Rechner erzeugte Zahl 8261 und Sie raten 6285, erhalten Sie als Spielergebnis 1.2. Das bedeutet, daß eine Ziffer an der richtigen Stelle ist (die 2), und daß zwei Ihrer anderen Ziffern (8 und 6) in der Geheimzahl zwar enthalten, aber nicht in der richtigen Position sind. Ein Spielergebnis von 4.0 zeigt an, daß Sie die Zahl richtig geraten haben.

Testen Sie Ihre Geschicklichkeit und entwickeln Sie ein eigenes Programm für dieses Spiel. Beschäftigen Sie sich dann eingehend mit dem unten angegebenen Beispiel. Optimieren Sie Ihr Programm mit dem Ziel, eine minimale Anzahl von Befehlen zu verwenden.

Der Verarbeitungsfluß in diesem Beispiel ist leicht zu verfolgen; die Komplexität des Kodeprogramms erfordert jedoch eine genaue Gegenüberstellung von Programmablaufplan und Erläuterungen.

Damit das Programm die geheime Zahl herleiten kann, muß es einen Startpunkt haben. Über wird eine Zahl vorgegeben, mit der der Rechner arbeiten kann.



Programmablaufplan	Bemerkungen	Speicherplatz u. Tastenkode	Tastenfolge
Definition von A als Vorgabe	Diese Routine verwendet man zur Bestimmung der Kodezahl.	000 76 001 11 002 22 003 58	2nd 1b1 A INV 2nd 111
Löschen des Speichers u. Einleitung von Programm 15	Mit Programm 15 der Standard-Softwareprogramme werden Zufallszahlen erzeugt.	004 47 005 36 006 15 007 15	2nd 1C15 2nd 171m 1 5 E
Erhöhen d. Registers für die indirekte Adressierung und Schleifenzähler	Die erzeugten Ziffern werden indirekt entsprechend dem Inhalt von R5 gespeichert. Die spezielle Steueroperation 25 verwendet man, um den Inhalt zu erhöhen.	008 76 009 44 010 69 011 25	2nd 1b1 SUM 2nd 1Op 2 5
Wieder abspeichern des Schleifenzählers	Mit einer Schleife testet man neue Ziffern auf Null sowie früher eingegebene Ziffern. Um die Ablaufzeit zu verkürzen, verwendet man eine minimale Anzahl von Schleifen.	012 76 013 42 014 43 015 05 016 42 017 06	2nd 1b1 STO RCL 5 STO 6
Erzeugen einer Zufallsziffer	Das Unterprogramm DMS von Programm 15 verwendet man, um eine Zufallszahl x zu erzeugen, wobei $0 \leq x < 1$. Um die Zahl in den geeigneten Bereich zu bringen, wird sie mit 10 multipliziert, und ihr ganzzahliger Wert wird in das T-Register eingebracht. Würde man die Zufallszahl mit Routine C von Programm 15 erzeugen, wäre zusätzlicher Programmspeicher erforderlich, um den Bereich der Ausgangsinformation festzulegen. Routine C verwendet auch die Datenregister R1 bis R6 , während Unterprogramm DMS ohne sie operiert. Beachten Sie, daß beim direkten Lösungsweg, der Multiplikation mit 10, ein weiterer Programmschritt notwendig ist, weil anstelle von INV EE der CLR -Befehl die Exponentialform der Anzeige aufhebt.	018 36 019 15 020 71 021 88 022 52 023 01 024 59 025 32 026 25	2nd 171m 1 5 SBR 2nd DMS EE 1 2nd 1b1 x<1 CLR

a

b

c

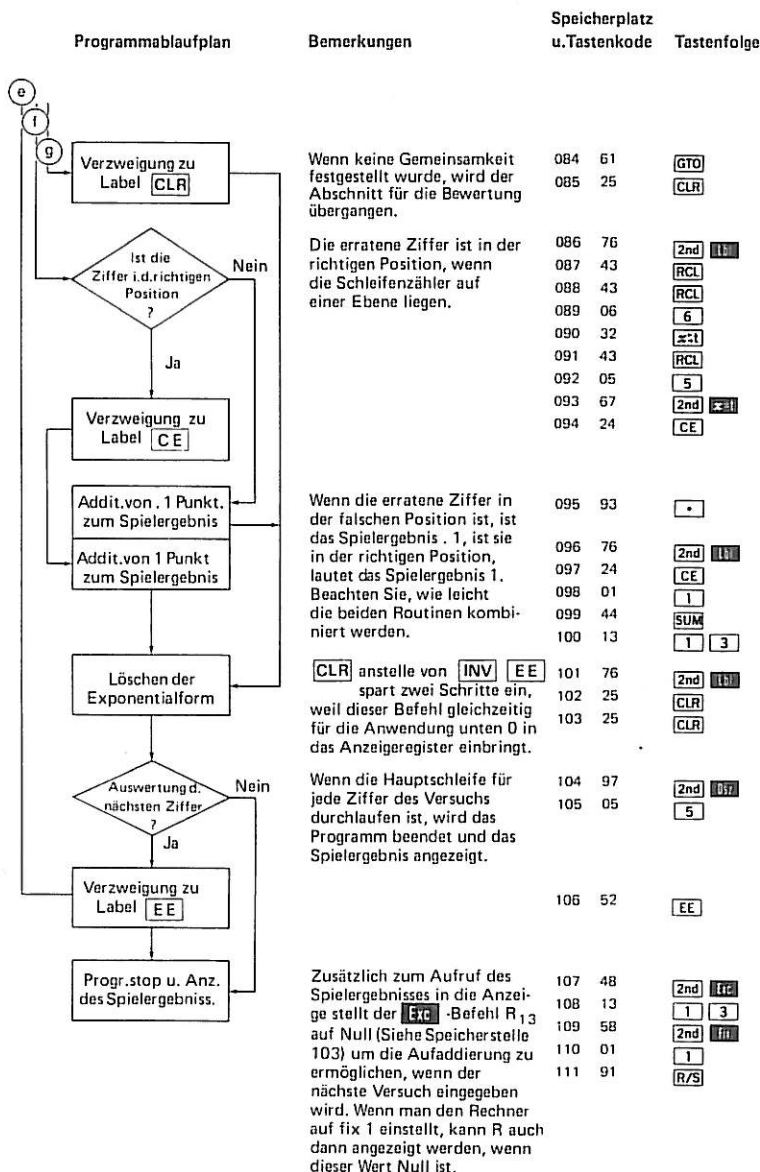


Programmlaufplan	Bemerkungen	Speicherplatz u. Tastenkode	Tastenfolge
<pre> graph TD a((a)) --> b((b)) b --> d1{Wird die Ziffer zurückgewiesen?} d1 -- Ja --> s1[Verzweigung zu Label STO] s1 --> b d1 -- Nein --> c((c)) c --> d2{Test der nächsten Ziffer?} d2 -- Ja --> s2[Verzweigung zu Label 007] s2 --> d2 d2 -- Nein --> s3[Speicherung der Zufallsziffer] s3 --> d3{Wird eine andere Ziffer erzeugt?} d3 -- Ja --> s4[Verzweigung zu Label SUM] s4 --> a d3 -- Nein --> s5[Programmstop und Anzeige 0] s5 --> s6[Definition v. als Versuch] s6 --> s7[Speicherung d. Versuchs u. Erstellen d. Hauptschleifenzählers] </pre>	Da zu Beginn die Datenregister gelöscht werden, testet die erste Schleife die erzeugte Ziffer auf Null. Die restlichen Schleifen testen gegen die zuvor erzeugten Ziffern. Wird die Ziffer zurückgewiesen, wird eine neue Ziffer ohne Erhöhung von R ₅ erzeugt. R ₆ dient zur indirekten Adressierung und als Schleifenzähler. Der 007-Befehl hat ebenfalls eine doppelte Funktion: er steuert den Schleifenprozeß und die indirekte Adressierung.	027 76 028 67 029 73 030 06 031 67 032 42	2nd 1b 2nd =1 RCL 2nd Ind 6 2nd =1 STO
	Wenn die Ziffer akzeptiert ist, wird sie indirekt gespeichert, wobei R ₅ als Hinweisregister für die indirekte Adresse verwendet wird.	036 32 037 72 038 05	=1 STO 2nd Ind 5
	Sobald vier Ziffern akzeptiert sind, ist die Kodezahl komplett. Der Test wird gegen 3 und nicht gegen 4 durchgeführt, um einen Programmschritt einzusparen. Die Alternative wäre 4 INV 2nd =1	039 43 040 05 041 32 042 03 043 77 044 44	RCL 5 =1 3 2nd =1 SUM
		045 00 046 91	0 R/S
	Diese Routine wird zur Bewertung der Rateversuche des Spielers benutzt.	047 76 048 12	2nd 1b B
	Zwei Schleifen werden in dieser Routine verwendet. Die Hauptschleife läuft für jede Ziffer im Versuch einmal durch.	049 42 050 12 051 04 052 42 053 05	STO 1 2 4 STO 5



IV

Programmablaufplan	Bemerkungen	Speicherplatz u. Tastenkode	Tastenfolge
<pre> graph TD Start(()) --> A[Herausnahme der Ziffer für den Test] A --> B[Erstellen des Nebenschleifenzählers] B --> C{Stimmen Ziffern überein?} C -- Ja --> D[Verzweigung zu Label RCL] C -- Nein --> E{Test gegen die nächste Ziffer d. Kodezahl?} D --> E E -- Ja --> F[Verzweigung zu Label 07h] E -- Nein --> G((0)) F --> H((1)) H --> I((9)) </pre>	<p>Dieser Befehl nimmt die letzte Ziffer des Versuchs auf und speichert sie im T-Register, sodaß sie mit der Kodezahl verglichen werden kann. Diese Ziffer wird auch aus dem Versuch entfernt. Die dritte Ziffer des Versuchs wird also dann die letzte Ziffer für die zweite Schleife usw. Die verwendete Gleichung lautet:</p> $\text{Ziffer} = 10 \times \text{INV Int} \frac{\text{Versuch}}{10}$ <p>Wieder wird der EE Trick für die Division und für die Multiplikation mit 10 genutzt.</p> <p>Die Nebenschleife testet die obige Ziffer gegen jede Ziffer in der Kodezahl.</p> <p>Hier wird die Ziffer separat vom Versuch des Spielers mit den Ziffern verglichen, die die Kodezahl aufweist. Wird eine Übereinstimmung festgestellt, verzweigt die Verarbeitung zu dem Abschnitt, die die Versuche auswertet.</p> <p>Dieser Befehl stellt sicher, daß jede geratene Ziffer mit jeder Ziffer der geheimen Zahl verglichen wird (wenn nicht eine Gemeinsamkeit in einem früheren Vergleich gefunden wurde). Wieder erfüllen R_6 und DSZ zwei Funktionen.</p>	<p>054 76 055 52 056 43 057 12 058 52 059 01 060 94 061 42 062 12 063 22 064 59 065 22 066 44 067 12 068 52 069 00 070 00 071 32 072 04 073 42 074 06 075 76 076 97 077 73 078 06 079 67 080 43 081 97 082 06 083 97</p>	<p>2nd [(1)] EE RCL 1 2 EE 1 +/- STO 1 2 INV 2nd [(1)] INV SUM 1 2 EE 0 0 ±=1 4 STO 6 2nd [(1)] 2nd [(07)] RCL 2nd [(1)] 6 2nd ±=1 RCL 2nd [(07)] 6 2nd [(07)]</p>





IV

Das hier entwickelte Programm erfordert die Eingabe einer Dezimalzahl (zwischen 0 und 1) über die Taste **A** als Vorgabe. Sobald die geheime Zahl hergeleitet ist, erscheint eine Null in der Anzeige. Beginnen Sie das Ratespiel mit der Eingabe Ihres Versuchs und drücken Sie **B**. Die Auswertung Ihres Versuchs wird in der früher erklärten Form angezeigt. Und nun das Beispiel:

Eingabe	Taste	Anzeige	Bemerkungen
.258	A	0.	Eintasten der Vorgabe, und Abwarten, bis die geheime Zahl bestimmt ist.
1234	B	0.1	Erster Versuch
5678	B	2.1	Zweiter Versuch
9238	B	1.0	Dritter Versuch
5694	B	1.0	Vierter Versuch
5198	B	2.1	Fünfter Versuch
5718	B	4.0	Der sechste Versuch ist richtig

Ein erfahrener Spieler braucht selten mehr als 6 Versuche.



EINE EINGEHENDE ANALYSE DER EIGENSCHAFTEN UND FUNKTIONEN

Im folgenden erhalten Sie eine ausführliche Analyse über alle Aspekte Ihres Rechners. In diesem Abschnitt sollen Sie alle Einzelheiten nachschlagen können, sobald Sie sich die Grundkenntnisse über die Funktionen des Rechners angeeignet haben. Wenn Sie den Rechner noch nicht gut kennen, empfiehlt es sich, die ersten Abschnitte dieses Buchs noch einmal durchzuarbeiten.

Alle Erklärungen über Operationen und Funktionen gelten sowohl für manuelle Berechnungen über die Tastatur als auch für die Anwendung dieser Operationen in Programmen.

GRUNDOPERATIONEN

Standardanzeige

Zusätzlich zu dem Hinweis, ob der Rechner eingeschaltet ist, liefert die Anzeige vollständige numerische Informationen, mit negativem Vorzeichen, und Dezimalkomma, und blinkt bei Kapazitätsüberlauf und -unterlauf sowie bei einer Fehlerbedingung. Eine Eingabe kann maximal 10-stellig sein. Alle Zifferneingaben nach der zehnten Stelle werden ignoriert.



Bei Anzeige einer negativen Zahl steht das Minuszeichen unmittelbar links vor dieser Zahl.



Informationen über die Genauigkeit aller angezeigten Ergebnisse – siehe Anhang C



Dateneingabe – Tasten

Diese Tasten sind von den übrigen abgesetzt, um die Arbeit mit dem Rechner noch effektiver zu machen. Viele der Operationen sind ganz klar erkennbar, es gibt aber einige Ausnahmen. Die folgenden Anleitungen und Beispiele helfen Ihnen, im Umgang mit Ihrem Rechner Sicherheit und Gewandtheit zu entwickeln.

ZIFFERTASTEN **0** bis **9** – Eingabe der Ziffern von 0 bis 9

. — DEZIMALKOMMA – Eingabe des Dezimalkommata. Es kann immer bei Bedarf eingegeben werden. Gibt man das Komma nicht ein, wird es automatisch rechts von der Zahl angenommen und erscheint in der Anzeige, sobald eine Operations- oder Funktionstaste gedrückt wurde. Bei Zahlen kleiner als 1 ist dem Komma eine Null vorangestellt, wenn nicht alle zehn verfügbaren Anzeigestellen genutzt wurden. Nachnullen beim Dezimalbruchteil einer Zahl werden im allgemeinen nicht ausgewiesen. Der Rechner akzeptiert nur die erste Komma-Eingabe, alle weiteren werden ignoriert. Drückt man das Komma unmittelbar nach der Eingabe eines Exponenten, kann die Mantisse wieder geändert werden, wie etwa ihr Vorzeichen.

2nd **π** — DIE ZAHL π – Eingabe des Wertes von π mit 13-stelliger Genauigkeit (3.141592653590) für Berechnungen; in der Anzeige erscheint der auf 10 Stellen gerundete Wert. Mit der Taste **CE** wird der π -Wert nicht gelöscht, er kann jedoch durch eine andere Zahl überschrieben werden.

+/- — VORZEICHENWECHSEL – Anweisung, das Vorzeichen der ausgewiesenen Zahl zu ändern. Drückt man **+/-** nach **EE**, oder nach Eingabe eines Exponenten, ändert sich das Vorzeichen des Exponenten. Die Eingabe einer positiven Zahl erfolgt einfach durch Drücken der entsprechenden Tasten in der Reihenfolge von links nach rechts, genau wie die Zahl geschrieben wird. Mit jeder Zifferneingabe werden die bereits angezeigten Ziffern um je eine Stelle nach links verschoben. Bei Eingabe einer einzelnen Zahl wird nur das erste Komma, das Sie eintasten, akzeptiert.

Beispiel: $7.892 - \pi + (-2) = 2.750407346$

Taste	Anzeige
7.892 -	7.892
2nd π	3.141592654
+	4.750407346
2 +/-	-2
=	2.750407346



Löschoperationen

[CE] (clear entry) – LÖSCHEN DER EINGABE – löscht die Eingaben, die mit den Zifferntasten, der Kommataste und der Vorzeichenwechsellaste erfolgten, jedoch nur, wenn zuvor keine Funktionstaste gedrückt wurde. Rechenergebnisse, Zahlen, die aus dem Speicher aufgerufen wurden, oder π werden mit der Taste **[CE]** nicht gelöscht. Wenn nötig, wird mit **[CE]** das Blinken der Anzeige abgestellt. Die **[CE]**-Anweisung beeinflusst keine unvollständigen Operationen.

[CLR] (general clear) – LÖSCHEN – löscht laufende Berechnungen und die Anzeige. Mit der Anweisung wird die Exponentialform aufgehoben und die Anzeige wieder auf das Standardformat eingestellt. Auch das Blinken der Anzeige wird gestoppt. Die Löschstaste hat keine Wirkung auf den Inhalt des Daten- und Programmspeichers, ebensowenig auf das T-Register, den Winkelmodus, auf technische Notation oder Festkomma-Anzeige oder auf die Speicherbereichsverteilung.

Tatsächlich löscht sich nach den meisten Berechnungen der Rechner automatisch. Wenn zum Abschluß einer Berechnung die Taste **[=]** gedrückt wird, erscheint das Ergebnis in der Anzeige und der Rechner ist damit für den Beginn einer neuen Aufgabe auch ohne Anwendung einer der Löschstasten vorbereitet. Die Inhalte der Datenspeicher werden nicht automatisch gelöscht.

[2nd] [CP] (clear program) – PROGRAMMLÖSCHUNG – löscht alle Programmspeicherplätze (einschl. Datenschutzes) sowie das Unterprogrammrücksperrungs-Register und das T-Register, alle Flags werden rückgesetzt, und der Programmzeiger wird auf 000 gestellt, wenn die Anweisung manuell über die Tastatur gegeben wird. Ist sie in ein Programm eingebaut, wird nur das T-Register gelöscht.

[2nd] [CM] (clear data memory) – DATENSPEICHER-LÖSCHUNG – Anweisung, alle Datenspeicher-Register entsprechend der im Augenblick festgelegten Speicherbereichsverteilung zu löschen.

Doppelfunktionstasten (**[2nd]** und **[INV]**) Die meisten Tasten Ihres Rechners sind doppelt belegt. Die Erstfunktion ist auf der Taste selbst aufgedruckt, und die Zweitfunktion ist direkt über der Taste angegeben. Um eine Funktion auszuführen, drücken Sie einfach die gewünschte Taste. Um die Zweitfunktion anzuwenden, drückt man die Taste **[2nd]**, und dann die Taste direkt unter der gewünschten Zweitfunktion. Zum Beispiel wird der natürliche Logarithmus einer Zahl berechnet, wenn man die Taste **[lnx]** drückt. Der Zehnerlogarithmus einer Zahl wird dagegen mit den Tasten **[2nd] [lnx]** ermittelt. Um die Zweitfunktion kenntlich zu machen, wurde in diesem Handbuch die Darstellung **[2nd] [Symbol]** gewählt. Erstfunktionen werden also durch das schwarze Symbol auf weißem Grund gekennzeichnet (**[Symbol]**), Zweitfunktionen durch **[2nd]** und das weiße Symbol auf schwarzem Grund (**[2nd] [Symbol]**). Wenn man **[2nd]** zweimal nacheinander drückt, oder wenn eine nach **[2nd]** betätigte Taste keine Zweitfunktion hat, operiert der Rechner automatisch mit der Erstfunktion.

Wenn die blinkende Anzeige durch ein unmittelbares Nacheinanderdrücken zweier Operationstasten erfolgt (Fehlerbedingung 5, Seite B-1), wird durch die Taste **[CE]** die zuletzt eingegebene Operationsanweisung gelöscht.



Wie mit der **[2nd]** – Taste wird auch mit der Umkehrfunktions-Taste **[INV]** die Rechenkapazität erweitert, ohne daß die Anzahl der Tasten erhöht werden muß. Stellt man **[INV]** einer anderen Taste voran, wird der Sinn dieser Taste umgekehrt. Die Umkehrfunktion kann zusammen mit folgenden Tasten angewandt werden, um die angegebene Funktion zu erhalten.

Funktion

Funktion	Umkehrfunktion
EE	EE wird aufgehoben
ENG	ENG und EE aufgehoben
Fix	FIX wird aufgehoben
log	10^x
ln x	e^x
y^x	$\sqrt[x]{y}$
Int	Bruchteil (Fraction, Vorkommateil wird abgeschnitten)
sin	arc sin
cos	arc cos
tan	arc tan
Prod	dividieren
SUM	subtrahieren
D.MS	Dezimalgrad
$P \rightarrow R$	$R \rightarrow P$
$\Sigma +$	$\Sigma -$
x	Standardabweichung
list	Datenregister-Auflistung
SBR	return (Rücksprung)
$x = t$	$x \neq t$
$x \geq t$	$x < t$
if flg	if no flag
st flg	reset flag
Dsz	Überspringen bei nicht-Null
Write	Einlesen (read)

Die Anwendung der Taste **[INV]** in Programmen ist in den entsprechenden Programmierabschnitten erläutert. Eine Umkehranweisung kann durch zweimaliges Drücken der **[INV]** – Taste aufgehoben werden, wenn dazwischen keine andere Taste betätigt wurde. Bei Tasten, die keine Umkehrfunktion haben, z. B. **[SC]**, **[LRN]**, etc., wird eine vorangestellte **[INV]** – Anweisung ignoriert. In Verbindung mit der **[2nd]** – Taste kann die Umkehrfunktionstaste vor oder nach **[2nd]** gedrückt werden, d. h., **[INV] [2nd] [log]** ist gleichbedeutend mit **[2nd] [INV] [log]**. Jedoch nicht im LRN-Modus. Im LRN-Modus muß die Folge **[INV] [2nd]** unbedingt eingehalten werden. Beispiele für die Anwendung von **[INV]** zusammen mit einer anderen bestimmten Taste finden Sie jeweils in dem Abschnitt, in dem diese Taste behandelt wird.



Anzeigeformen

Zusätzlich zu der vielseitigen 10-stelligen Standardform der Anzeige gibt es noch einige andere Anzeigemöglichkeiten, die den Operationsbereich und die Flexibilität des Rechners erweitern. Obwohl nur maximal 10 Stellen eingegeben oder ausgewiesen werden können, hält das interne Anzeigeregister die Ergebnisse immer mit 13 Stellen. Die Ergebnisse werden dann nur für die Anzeige gerundet. Diese zusätzlichen Stellen sollen die Toleranz des angezeigten Wertes sichern, sie sind aber nicht als Erweiterung des Genauigkeitsgrades gedacht. Vorsicht bei der Anwendung dieser Schutzstellen! Ausführliche Informationen über Genauigkeit siehe Anhang C.

EXPONENTIALFORM

EE (enter exponent) – EXPONENTEN-EINGABE – Anweisung, daß die folgende Zahleneingabe ein Exponent von 10 ist. Nach Drücken der Taste **EE** werden alle weiteren Resultate in der Exponentialform ausgewiesen, bis man **CLR** drückt oder den Rechner ausschaltet. Auch mit den Tastenfolgen **INV EE** oder **INV 2nd Eng** kann dieses Anzeigeformat aufgehoben werden, aber nur, wenn die ausgewiesene Zahl im Bereich $\pm 5 \times 10^{-11}$ bis $\pm 1 \times 10^{10}$ liegt. Wird **EE** nach einem Ergebnis (Zwischen- oder Endergebnis) gedrückt, werden die Stellen 11, 12 und 13 gestrichen und nur der Wert in der Anzeige wird bei weiteren Berechnungen zugrunde gelegt.

Jede Zahl kann als das Produkt aus einem Wert (der Mantisse) und einer Zehnerpotenz (dem Exponenten), eingegeben werden. Sie geben die Mantisse ein (bis zu 8 Stellen), drücken **EE**, und geben dann den Exponenten ein (beliebige 2 Stellen).



Diese Möglichkeit erlaubt die Arbeit mit Zahlen im Bereich $\pm 1 \times 10^{-99}$ bis $\pm 9.9999999 \times 10^{99}$. Zahlen, die kleiner sind als .000000001 oder größer als 9999999999 müssen in der Exponentialform eingegeben werden. Wenn Rechenergebnisse diese Grenzen überschreiten, schaltet sich der Rechner automatisch auf Exponentialform. Beim Eingabevorgang tastet man zunächst die maximal 8-stellige Mantisse (einschließlich Vorzeichen) ein, drückt **EE** und gibt dann den Exponenten von 10 und dessen Vorzeichen ein.

Beispiel: Die Zahl 320 000 000 000 kann als 3.2×10^{11} geschrieben und wie folgt in den Rechner eingegeben werden:

Taste	Anzeige
CLR	0
3.2	3.2
EE	3.2 00
11	3.2 11



Nachdem **[EE]** gedrückt wurde, können zwar mehr als zwei Ziffern eingegeben werden, aber nur die beiden zuletzt eingetasteten werden als Exponent interpretiert. Damit erhält man die Möglichkeit, eine falsche Exponenteneingabe zu korrigieren, ohne die gesamte Eingabe mit **[CE]** löschen zu müssen.

Ein positiver Exponent zeigt an, um wieviel Stellen das Komma der Mantisse nach rechts verschoben werden müßte. Ist der Exponent negativ, müßte das Komma entsprechend nach links gerückt werden.

Unabhängig davon, wie eine Mantisse für die Exponentialform eingegeben wird, normiert der Rechner die Zahl in der Form, daß links vom Dezimalkomma eine einzelne Stelle hervorgehoben wird, wenn man eine Funktions- oder Operationstaste drückt.

Beispiel: Eingabe von 6025×10^{20}

Taste	Anzeige
[CLR]	0
6025	6025
[EE]	6025 00
20	6025 20
[+]	6.025 23

In der Exponentialform ist die Mantisse auf maximal 8 Stellen limitiert, um Raum für die Anzeige des Exponenten zu erhalten. Eine Mantisse, die auf eine Berechnung zurückzuführen ist, wird ebenfalls mit bis zu 8 Stellen ausgewiesen, wird aber intern mit bis zu 13 Stellen geführt. Der 13-stellige Wert wird auch für alle nachfolgenden Berechnungen verwendet. Mehr über diese Stellen erfahren Sie im Anhang C.

Anmerkung: Wenn mehr als 8 Ziffern für die Mantisse eingegeben wurden, ist der Übergang auf die Exponentialform nicht sofort möglich, auch wenn **[EE]** gedrückt wird. Drückt man **[EE]** mit mehr als 8 Stellen in der Anzeige, erfolgt die Umstellung auf Exponentialform erst, wenn eine Operations- oder Funktionstaste gedrückt wird.

Die Vorzeichenwechsellaste kann dazu benutzt werden, der Mantisse und dem Exponenten der Zehnerpotenz ein negatives Vorzeichen zuzuweisen.

Drücken Sie einfach **[+/-]** nach Eingabe der Mantisse oder des Exponenten, um das betreffende Vorzeichen zu ändern. Wenn Sie das Vorzeichen der Mantisse ändern oder in deren Dezimalbruchteil noch Ziffern eingeben wollen, nachdem die Taste **[EE]** gedrückt wurde, korrigiert man mit der Taste **[.]**, und gibt dann das Vorzeichen der Mantisse oder die zusätzlichen Ziffern zum Dezimalbruchteil ein.

Beispiel: Geben Sie -4.962×10^{-12} ein, korrigieren Sie den Exponenten und vervollständigen Sie dann den Dezimalbruchteil der Mantisse auf -4.96236×10^{12} .

Taste	Anzeige	Bemerkungen
[CLR]	0	
4.962 [+/-]	-4.962	Eingabe der Mantisse und des Vorzeichens
[EE]	-4.962 00	
12 [+/-]	-4.962 -12	Eingabe des Exponenten und d. Vorzeichens
[+/-]	-4.962 12	Vorzeichenänderung für den Exponenten
[.] [+/-]	4.962 12	Vorzeichenänderung für die Mantisse
36 [+/-]	-4.96236 12	Ergänzung der Mantisse



Daten in Exponentialform können mit Daten in der Standardform gemischt werden. Der Rechner wandelt die eingegebenen Daten für die korrekte Berechnung um. Wenn **EE** gedrückt wurde, weist der Rechner alle Ergebnisse in der Exponentialform aus, bis man **CLR**, die Folgen **INV** **EE** oder **INV** **2nd** **Eng** drückt, oder bis der Rechner ausgeschaltet wird. **CE** löscht eine Eingabe in Exponentialform, aber die Anzeigeform selbst wird dadurch nicht aufgehoben.

Beispiel: $1.816 \times 10^3 - 581.432191 = 1.2345678 \times 10^3 = 1234.567809$

Taste	Anzeige
CLR	0
1.816 EE	1.816 00
3 -	1.816 03
581.432191 =	1.2345678 03
INV EE	1234.567809

Drückt man **INV** **EE**, um die Exponentialform aufzuheben, und ist die Zahl nicht im Bereich $\pm 1 \times 10^{10}$ bis $\pm 5 \times 10^{-11}$, schaltet sich der Rechner nur dann wieder auf das Standardformat, wenn ein Rechenergebnis noch im anzeigbaren Bereich liegt.

Beispiel: $(7 \times 10^{11} + 5 \times 10^{10}) \div 25 = 1200000000$

Taste	Anzeige
7 EE	7 00
11 +	7. 11
5 EE	5 00
10 = INV EE	7.5 11
÷	7.5 11
25 = ÷	3. 10
25 =	1200000000.

Wenn der berechnete Wert 9999999999 übersteigt oder unter .0000000001 liegt, schaltet sich die Anzeige automatisch in Exponentialform. Wenn die Exponentialform eingeführt wird, ohne daß **EE** im Verlauf dieser Rechenfolge gedrückt wurde, erfolgt die Umstellung auf die Standardanzeige automatisch, sobald dieses numerisch möglich ist.

Für die Umwandlung von Rechenergebnissen auf Exponentialform gibt es zwei Möglichkeiten.

1) Man drückt **X** **1** **EE** **=**, eine Folge, mit der die Zahl im Anzeigeregister mit $1 \times 10^0 = 1$ multipliziert und die Anzeige auf Exponentialform umgeschaltet wird. Die volle 13-stellige Zahl bleibt erhalten.

2) Man drückt **EE** **=**. Vorsicht bei dieser zweiten Methode. Mit ihr wird der Rechner zugleich angewiesen, den GERUNDETEN Wert in der Anzeige für nachfolgende Berechnungen zu verwenden.

Vermeiden Sie Anzeigebefehle mit einem **=** mitten in einer Berechnung, weil mit der **=**-Anweisung alle unvollständigen Berechnungen abgeschlossen werden. Um dies zu vermeiden, wenden Sie die genannten Umformungsmethoden nur nach Abschluß Ihrer Berechnungen an, oder aber Sie multiplizieren mit **X** **1** **EE**, gefolgt von einer anderen Operation.



TECHNISCHE NOTATION

Mit den Tasten **2nd** **Eng** erhält man eine modifizierte Art der Exponentialform. Der ausgewiesene Wert in dieser Anzeigeform besteht aus einer Mantisse und einem Exponenten, die so ausgerichtet sind, daß der Exponent ein Vielfaches von drei ist (10^{12} , 10^{-6} , etc.), und daß die Mantisse 1,2 oder 3 Stellen links vom Komma aufweist. Damit kann der Rechner die Resultate sofort in brauchbaren Maßeinheiten anzeigen, wie 10^{-12} für Picofarad, 10^{-3} für Millimeter, 10^6 für Megohm oder 10^{-9} für Nanosekunden.

Beispiel: Berechnen Sie den Durchmesser eines Fadens in Mikrometer (1 Mikrometer = 10^{-6} Meter), dessen Umfang 3×10^{-3} Meter beträgt.

$$U = \pi d \quad d = U/\pi$$

Taste	Anzeige
CLR 2nd Eng	0. 00
3 EE	3 00
3 +/- ÷	3. -03
2nd π =	954.92966-06

Mit den Tasten **INV** **2nd** **Eng** wird dieses Anzeigeformat wieder aufgehoben, Löschooperationen oder **INV** **EE** haben hierauf jedoch keinen Einfluß.

FESTKOMMA-KONTROLLE

Im Standardformat der Anzeige, in der Exponentialform oder in der technischen Notation kann die Anzahl der ausgewiesenen Stellen nach dem Komma gewählt werden. Drückt man **2nd** **Fix** und dann die gewünschte Anzahl von Dezimalstellen (0 bis 8), wird der Rechner angewiesen, alle angezeigten Ergebnisse auf die vorgewählte Dezimalstellenzahl zu runden. Diese Rundung wirkt nur auf die Anzeige, nicht auf das Anzeigeregister. Für alle weiteren Berechnungen wird also der volle ungerundete Wert verwendet.

Mit den Folgen **2nd** **Fix** 9 oder **INV** **2nd** **Fix** wird die Festkommawahl aufgehoben. Dateneingaben können nach wie vor 10-stellig (8-stellig in der Exponentialform) sein, und alle folgenden Berechnungen basieren auf den 13-stelligen ungerundeten Ergebnissen; davon ausgenommen sind die Grad-Umrechnungen, wo nur der angezeigte Wert benutzt wird. Nur die Anzeige wird für die geforderte Dezimalstellenzahl geändert, es sei denn, Sie drücken **EE** **INV** **EE**, um den nicht ausgewiesenen Teil zu löschen.

Die Constant Memory-Eigenschaft des TI-58C hält die Festkomma-Einstellung gespeichert. Die Festkomma-Wahl bleibt daher bis zu einer Änderung oder bei kurzfristigem Ausfall der Batterieversorgung wirksam.



Achten Sie darauf, daß Sie die technische Notation vom vorigen Beispiel aufgehoben haben.

Beispiel: $2/3 = .666666667$

Taste	Anzeige
2 \div	2.
3 $=$.666666667
2nd fix 5	0.66667
2nd fix 2	0.67
2nd fix 0	1.
INV 2nd fix	.666666667

Bedenken Sie, daß der Wert im Anzeigeregister auf das gewünschte Format gerundet wird.

Beispiel: $1 \times 10^{-3} : 2 = .0005$

Taste	Anzeige
1 EE	1 00
3 $+/-$ \div	1.-03
2 $=$	5.-04
2nd fix 2	5.00-04
INV EE	0.00 *
2nd fix 3	0.001
2nd fix 4	0.0005
2nd fix 5	0.00050

* Beachten Sie, daß die Null etwa in der Mitte des Beispiels nicht wirklich eine Null im Anzeigeregister ist. Der Wert rundet sich nur nicht auf das Anzeigeformat mit 2 Dezimalstellen. Bedenken Sie immer, daß das Anzeigeregister durch die Wahl der Dezimalstellen nicht beeinflußt wird.

BLINKENDE ANZEIGE

Die Anzeige blinkt immer dann, wenn die Kapazitätsgrenzen des Rechners überschritten wurden oder wenn eine unzulässige mathematische Operation gefordert wird. Drücken Sie CE , um das Blinken ohne Einwirkung auf laufende Berechnungen abzustellen. Von dieser Stelle an können die Berechnungen fortgesetzt werden, wenn die Zahl in der Anzeige noch brauchbar ist. Auch mit CLR wird das Blinken abgestellt, hier werden jedoch auch laufende Berechnungen und der Anzeigewert gelöscht. Eine komplette Auflistung der Fehler-, sowie der Überlauf-/Unterlauf-Bedingungen und deren Folgen finden Sie im Anhang C.



ARITHMETISCHE BERECHNUNGEN

Die Eingabemethode für Zahlen und Operationen entspricht in den meisten Fällen der mathematischen Problemstellung. Der Rechner „erinnert“ sich an jede Operation, und speichert sie, wenn nötig, bis sie entsprechend den Standardregeln der Algebra durchgeführt werden können.

Die Grundfunktionen – $+$ $-$ \times \div $=$

Bei der Lösung einfacher Additions-, Subtraktions-, Multiplikations- oder Divisionsaufgaben kann bei diesem Rechner mit seinem algebraischen Operationssystem das Problem so, wie es geschrieben wird, auch eingetastet werden.

Beispiel: $1.6 \times 10^{-19} \times 6.025 \times 10^{23} = 9.64 \times 10^4$

Taste	Anzeige
CLR	0
1.6 EE	1.6 00
19 +/- X	1.6 -19
6.025 EE	6.025 00
23 =	9.64 04

Beachten Sie, daß mit der Taste **=** die arithmetische(n) Operation(en) abgeschlossen werden und das Endergebnis angezeigt wird.

Mit der **CLR** -Anweisung zu Beginn einer neuen Folge werden alle laufenden Berechnungen gelöscht, und es ist immer sichergestellt, daß keine unvollständigen Operationen aus früheren Berechnungen zurückbleiben.

Der Löschbefehl erübrigt sich, wenn die vorhergehende Aufgabe mit **=** abgeschlossen wurde. Ein nachfolgendes **=** mit einer Zahleneingabe hat dieselbe Wirkung wie **CLR** mit dem Unterschied, daß **=** weder die Exponentialform aufhebt noch das Blinken der Anzeige abstellt.

Drückt man zwei der Operationstasten $+$ $-$ \times \div **y^x** – hintereinander, blinkt die Anzeige, ebenso, wenn einer dieser Tasten ein **=** oder eine rechte Klammer **)** folgt, oder wenn eine linke Klammer **(** vorangestellt ist. Mehr über Fehlerbedingungen siehe Anhang B.

Das Ergebnis aus einer Berechnung kann direkt als erste Zahl in einer zweiten Rechnung verwendet werden. Es ist unnötig, die Zahl erneut einzutasten.

Beispiel: $1.84 + 0.39 = 2.23$ und dann $(1.84 + 0.39)/365 = .006109589$

Taste	Anzeige	Bemerkungen
1.84 +	1.84	
.39 =	2.23	$1.84 + 0.39$
\div	2.23	
365 =	0.006109589	$2.23 \div 365$



Algebraisches Operationssystem

Die algebraische Hierarchie ist ein wesentliches Merkmal des algebraischen Operationssystems. Um Operationen wirksam kombinieren zu können, wurden die Standardregeln der algebraischen Hierarchie in den Rechner einprogrammiert. Diese algebraischen Regeln ordnen den verschiedenen mathematischen Operationen Prioritäten zu. Ohne eine feste Rangliste oder Prioritäten wären Ausdrücke wie $5 \times 4 + 3 \times 2$ mehrdeutig.

oder	$5 \times (4 + 3) \times 2 = 70$	
oder	$(5 \times 4) + (3 \times 2) = 26$	*Die Klammern können entfallen
oder	$((5 \times 4) + 3) \times 2 = 46$	
oder	$5 \times (4 + (3 \times 2)) = 50$	

Die Regeln besagen, daß die Multiplikation stets vor der Addition auszuführen ist. Die algebraisch korrekte Antwort ist also $(5 \times 4) + (3 \times 2) = 26$. Nachstehend die vollständige Liste der Prioritäten für die Interpretation von mathematischen Ausdrücken:

1. Mathematische Funktionen
2. Potenzen (y^x) und Wurzeln ($\sqrt[x]{y}$)
3. Multiplikation, Division
4. Addition, Subtraktion
5. Gleichheitsanweisung

1. Bei mathematischen Funktionen wird der angezeigte Wert sofort durch den Wert der Funktion ersetzt; (trigonometrische und logarithmische Funktionen, Quadrate, Quadratwurzeln, e^x , 10^x Ganzzahl- und Nachkommafunktion, Absolutwert, Reziprokwert und Umrechnungen)
2. Potenzen (y^x) und Wurzeln ($\sqrt[x]{y}$) werden als nächste berechnet.
3. Nach den mathematischen Funktionen, den Potenzen und Wurzeln und anderen Multiplikationen und Divisionen folgen Multiplikation und Division.
4. Addition und Subtraktion werden erst nach Abschluß aller Operationen einschließlich Multiplikation und Division und anderer Additionen und Subtraktionen durchgeführt.
5. Die Gleichheitsanweisung schließt alle unvollständigen Operationen in der angegebenen Reihenfolge ab.



Eine Operation schließt eine andere Operation mit gleicher (oder niedrigerer) Prioritätsstufe ab. Der Rechner kennt diese Regeln und wendet sie auf jedes eingetastete Problem an. Einige Operationen werden sofort ausgeführt, andere werden unvollständig belassen, bis sie den Regeln entsprechend abgeschlossen werden können. Beachten Sie zur Veranschaulichung die Zuordnung im folgenden Beispiel:

Beispiel: $4 : 5^2 \times 7 + 3 \times 0,5 \cos 60^\circ = 3.241320344$

Taste	Anzeige	Bemerkungen
4 \div	4.	(4 :) wird gespeichert
5 \wedge^2	25.	(5 ²) wird als Sonderfunktion
$\square \wedge^2$		$\square \wedge^2$ sofort errechnet
\times	0.16	(4 : 5 ²) x wird berechnet, weil : gleiche
7 $+$	1.12	Priorität hat wie x
3 \times	3.	x ist in der Priorität höher eingestuft als +, also
.5 \times	0.5	wird (4 : 5 ² x 7) berechnet und 1,12 + gespeichert
60 2^{nd} \cos	0.5	(3x) wird gespeichert
\square	3.241320344	.5 \times wird gespeichert
		$\cos 60^\circ$ wird sofort berechnet
		alle Operationen werden abgeschlossen:
		.5 $\cos 60^\circ$ wird zuerst berechnet, dann folgt
		3 x .5 $\cos 60^\circ$, und schließlich wird dieses Er-
		gebnis zu 1.12 addiert;

Bei Eingabe des Ausdrucks, wie er geschrieben wird, kann der Rechner ihn richtig als

$$[(4 : 5^2) \times 7] + (3 \times 0.5 (\cos 60^\circ))$$

interpretieren.

Der wesentliche Aspekt hierbei ist, daß die Operationen streng entsprechend ihrer relativen Priorität stattfinden, so wie die Regeln es erfordern. Der Rechner behält alle gespeicherten Operationen, und ruft für die Ausführung jede Operation und ihre zugeordnete Zahl auf, zur rechten Zeit und an der richtigen Stelle. Ist man einmal vertraut mit der Reihenfolge dieser Operationen, sind die meisten Aufgaben ohne Schwierigkeiten zu lösen, nicht zuletzt wegen der direkten Eingabemethode. Durch Anwendung von Klammern erhalten Sie eine weitere Möglichkeit, die Verarbeitungsfolge zu kontrollieren.

Klammern

Es gibt Operationsfolgen, bei denen es erforderlich wird, den Rechner genau anzuweisen, wie ein Problem zu berechnen ist, um das richtige Ergebnis zu erhalten. Mit Klammern können Sie Zahlen und Operationen gruppieren. Wenn Sie eine Reihe von Zahlen und Operationen einklammern, weisen Sie damit den Rechner an, diesen Ausdruck zuerst zu bearbeiten — auf eine einzelne Zahl zu reduzieren — und dann fortzufahren.



Der Vorteil von Klammern wird bei folgendem Versuch deutlich: Drücken Sie $\{ 5 \times 7 \}$, und der Wert 35 erscheint in der Anzeige. Der Rechner ermittelte 5×7 und ersetzte diesen Ausdruck durch 35, obwohl die Taste $=$ nicht gedrückt wurde. Wegen dieser Funktion der Klammern gelten die algebraischen Regeln mit ihrer Operationshierarchie jetzt für jedes Klammernpaar. Mit Klammern wird sichergestellt, daß Sie Ihr Problem so eintasten können, wie Sie einen Ansatz formuliert hätten. Der Rechner behält jede Operation und berechnet jeden Teilausdruck, sobald alle notwendigen Informationen verfügbar sind. Wenn eine rechte Klammer gesetzt wird, werden alle Operationen bis zurück zur entsprechenden offenen Klammer abgeschlossen. Klammern sind auch dann empfehlenswert, wenn Sie Zweifel haben, wie der Rechner einen Ausdruck verarbeitet.

Obwohl Ausdrücke manchmal in der Form $(3 + 2) (4 + 5)$ geschrieben sind, wobei man zwischen den Klammernpaaren eine Multiplikation voraussetzt, müssen Sie manuell eine Multiplikationsanweisung eintasten, damit diese Operation durchgeführt wird. Der Rechner führt keine Operationen durch, die auf diese Weise angezeigt sind.

Beispiel: $4 \times (5 + 9) : (7 - 4)^{(2 + 3)} = .2304526749$

Geben Sie diesen Ausdruck ein und verfolgen Sie den Rechengang.

Taste	Anzeige	Bemerkungen
4 \times (4.	(4x) wird gespeichert; die Berechnung des Klammerausdrucks bleibt unvollständig
5 +	5.	(5+) wird gespeichert
9)	14.	(5 + 9) wird berechnet
\div	56.	nach der Hierarchie wird (4×14) berechnet;
(56.	(56:) wird gespeichert; die Berechnung des Klammerausdrucks bleibt unvollständig
7 -	7.	(7-) wird gespeichert
4)	3.	(7 - 4) wird berechnet
y^x (3.	Vorbereitung des Exponenten
2 +	2.	
3)	5.	(2 + 3) wird berechnet
=	.2304526749	(7 - 4) ^(2 + 3) wird berechnet und dann wird $4 \times (5 + 9)$ durch dieses Ergebnis dividiert

Die Anzahl der Operationen und zugeordneten Zahlen, die gespeichert werden können, ist begrenzt. Tatsächlich können maximal 9 Klammern gleichzeitig geöffnet und maximal 8 Operationen gleichzeitig unvollständig sein, aber nur in extrem komplexen Aufgaben nähert man sich dieser Limitierung. Beim Versuch, mehr als 9 Klammern zu öffnen, oder wenn der Rechner mehr als 8 Operationen speichern soll, blinkt die Anzeige.



Beispiel: $5 + \left\{ 8 / [9 - (2/3)] \right\} = 5.96$

Taste	Anzeige	Bemerkungen
5 + (5.	
8 ÷ (8.	
9 - (9.	
2 ÷ 3)	.666666667	(2/3) wird berechnet
)	8.333333333	[9 - (2/3)] wird berechnet
)	0.96	8/[9 - (2/3)]
=	5.96	5 + {8/[9 - (2/3)]}

Da die **=**-Taste die Eigenschaft hat, alle unvollständigen Operationen abzuschließen, könnte sie auch hier anstelle der drei **)**-Tasten verwendet werden. Rechnen Sie diese Aufgabe noch einmal, drücken Sie aber anstelle der ersten rechten Klammer **)** die Taste **=**.

Beispiel: $3 \times \left\{ 4 \left[2^{-\left(\sqrt[3]{7}\right)} \right] \right\} = 4.700043401$

Taste	Anzeige	Bemerkungen
CLR (0.	
3 × (3.	
4 y^x (4.	
2 y^x (2.	
7 INV y^x	7.	
4)	1.626576562	$\sqrt[4]{7}$
+/-	-1.626576562	$-(\sqrt[4]{7})$
)	.3238557891	$2^{-(\sqrt[4]{7})}$
)	1.566681134	$4^{.323...}$
)	4.700043401	$3 \times 4^{.323...}$

Bei jeder geschlossenen Klammer wird deren Inhalt bis zurück zur nächsten offenen Klammer berechnet und durch einen einzigen Wert ersetzt. Mit dieser Kenntnis können Sie die Verarbeitungsfolge für jeden beliebigen Zweck bestimmen. Insbesondere können auch Zwischenergebnisse überprüft werden.



BLINDOPERATION MIT KLAMMERN

Eine weitere Technik im Zusammenhang mit Klammern ist die **[CE]** – Blindoperation. Mit **[CE]** kann der Anzeigewert ein zweites Mal eingegeben werden, ohne daß man den Wert selbst wieder eintasten muß. Insbesondere kann damit ein Wert in ein Klammerpaar eingebracht werden, wenn er in einem Ausdruck zweimal nacheinander benötigt wird. Verfolgen Sie den Rechenweg im nachstehenden Beispiel.

Beispiel: $3.296214 + (3.296214 \times 6) = 23.073498$

Taste	Anzeige	Bemerkungen
[CLR] 3.296214 [+]	3.296214	
[(] [CE] [X]	3.296214	[CE] gibt den Wert 3.296214 erneut ein
6)	19.777284	
=	23.073498	

Der vielziffrige Wert 3.296214 mußte nur einmal eingegeben werden.

ALGEBRAISCHE FUNKTIONEN

Die Funktionen mit einer Variablen sind am einfachsten zu beschreiben und zu verstehen. Diese Funktionen operieren mit dem Wert im Anzeigeregister und ersetzen diesen Wert sofort durch den entsprechenden Funktionswert. Diese Funktionen beeinflussen keine laufenden Berechnungen und können deshalb jederzeit innerhalb einer Rechnung verwendet werden. Die Genauigkeit dieser Funktionen wird im Anhang C und im Text dort, wo es erforderlich ist, behandelt.

Reziprokwert

[1/x] – REZIPROKWERT – berechnet den Reziprokwert des Wertes x im Anzeigeregister durch Division von 1 durch x. Wenn x = 0 ist, blinkt die Anzeige.

Taste	Anzeige
3.2 [1/x]	0.3125

Beachten Sie, daß der Anzeigewert unmittelbar durch den entsprechenden Funktionswert ersetzt wird, wenn man eine der mathematischen Funktionstasten drückt.



Logarithmen

[lnx] – NATÜRLICHER LOGARITHMUS – Berechnung des natürlichen Logarithmus (zur Basis e) des Wertes x im Anzeigeregister. Die Anzeige blinkt bei $x \leq 0$.

[2nd] [log] – ZEHNERLOGARITHMUS – Berechnung des Zehnerlogarithmus (zur Basis 10) des Wertes x im Anzeigeregister. Die Anzeige blinkt bei $x \leq 0$.

Beispiel: $\log(1 + \ln 1.7) = .1848697249$

Taste	Anzeige
[CLR]	0
([1] +	1.
1.7 [lnx])	1.530628251
[2nd] [log]	.1848697249

Potenzen von 10 und e

[INV] [lnx] – NATÜRLICHER ANTILOGARITHMUS (e^x) – Berechnung des natürlichen Antilogarithmus e^x für den Wert x im Anzeigeregister. $-227.9559242 \leq x \leq 230.2585092$; bei Werten außerhalb dieses Bereichs blinkt die Anzeige.

[INV] [2nd] [log] – DEKADISCHER ANTILOGARITHMUS (10^x) – Berechnung des dekadischen Antilogarithmus 10^x für den Wert x im Anzeigeregister. $-99 \leq x \leq 99.999999998$; bei Werten außerhalb dieses Bereichs blinkt die Anzeige.

Beispiel: $e^{(3 + 10^{0.3})} = 147.7116873$

Taste	Anzeige
[CLR] ([3] +	3.
.3 [INV] [2nd] [log])	4.995262315
[INV] [lnx]	147.7116873

Winkelberechnungen

Ihr Rechner gestattet Ihnen ein Maximum an Flexibilität bei Berechnungen mit Winkeln.

WINKELMODI

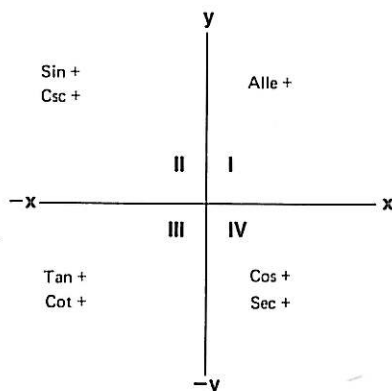
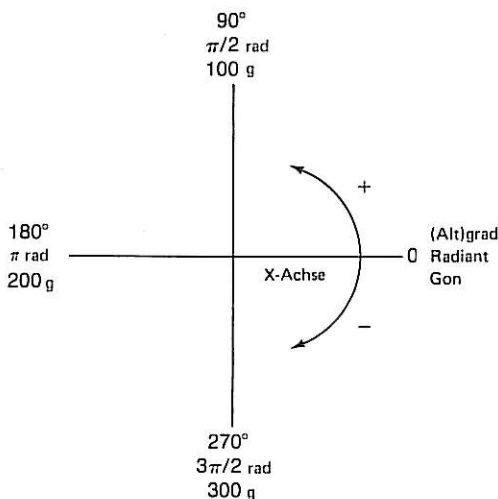
Winkel können in Dezimalgrad, Radiant oder Neugrad (Gon) gemessen werden (rechter Winkel = $90^\circ = \pi/2$ Radiant = 100 Gon). Hierzu drücken Sie entweder **[2nd] [Deg]** – für Altgrad, **[2nd] [Rad]** – für Radiant oder **[2nd] [Grad]** – für Neugrad. Beim Einschalten ist der Rechner auf den Winkelmodus Altgrad eingestellt. Dieser Modus wird beibehalten, bis er durch eine der anderen Alternativen geändert wird. Ist der Rechner einmal auf einen bestimmten Modus geschaltet, werden alle eingegebenen und berechneten Winkel in dieser Einheit gemessen, bis man eine andere Maßeinheit wählt oder den Rechner ausschaltet. **[CE]** und **[CLR]** wirken nicht auf den Winkelmodus.

Der Winkelmodus hat keinerlei Einfluß auf Berechnungen, außer wenn die trigonometrischen Funktionen oder die Polar-/rechtwinkligen Umrechnungen durchgeführt werden. Die Wahl der richtigen Winkleinheit ist sehr einfach – UND LEICHT ZU VERGESSEN. Das Unterlassen dieses Schritts ist die Ursache für viele Fehler, wenn mit Rechnern gearbeitet wird, bei denen die Winkleinheit gewählt werden kann.



TRIGONOMETRISCHE FUNKTIONEN

[2nd] [sin] · [2nd] [cos] · [2nd] [tan] – Sinus, Kosinus, Tangens – Berechnung von Sinus, Kosinus oder Tangens des Wertes im Anzeigeregister. Alle Winkel werden von der X-Achse aus gemessen, gegen den Uhrzeigersinn für positive Winkel und im Uhrzeigersinn für negative Winkel, wie unten dargestellt.



Das Diagramm rechts zeigt, in welchem Quadranten, I – IV, die angegebenen trigonometrischen Funktionen positiv sind. Diejenigen Funktionen, die in einem bestimmten Quadranten nicht aufgelistet sind, haben negative Werte.

Bedenken Sie beim Messen von Winkeln, daß jeder Winkel eine Entsprechung mit entgegengesetztem Vorzeichen hat. Zum Beispiel $-45^\circ = 315^\circ$.

Wenn Ihr Winkel in Grad – Minuten – Sekunden ausgedrückt ist, kann er mit der Taste **[D.MS]** in seine Dezimalform umgerechnet werden. Siehe das Kapitel über Umrechnungen im Abschnitt V, Seite 31. Stellen Sie sicher, daß der Rechner auf den Winkelmodus Altgrad geschaltet ist. Im Zweifelsfall drücken Sie **[2nd] [Deg]**.

Beispiel: $\sin 30^\circ 13' 48'' + \tan 315^\circ = -0.5$

Taste	Anzeige
30.1348 [2nd] [D.MS]	30.23
[2nd] [sin] [+]	.5034724109
315 [2nd] [tan]	-1.
[=]	-.4965275891



Trigonometrische Werte können auch für Winkel größer als ein Einheitskreis berechnet werden. Solange die trigonometrische Funktion im Standardformat und nicht in der Exponentialform angezeigt wird, sind für den Bereich von $\pm 36\,000$ Grad, $\pm 200\pi$ Radiant oder $\pm 40\,000$ Gon alle ausgewiesenen Stellen genau. Außerhalb dieses Bereichs nimmt die Genauigkeit um eine Stelle je Dekade ab. Ist das Argument x größer als $\pm 3.6 \times 10^{14}$ Grad (4.0×10^{14} Gon) oder größer als $\pm 6.2799993 \times 10^{12}$ Radiant, wird die Periodizität der Funktionen nicht mehr erkannt.

Die anderen trigonometrischen Funktionen können genauso problemlos berechnet werden.

$$\begin{aligned}\cot &= \boxed{2nd} \boxed{\tan} \boxed{1/x} \\ \sec &= \boxed{2nd} \boxed{\cos} \boxed{1/x} \\ \csc &= \boxed{2nd} \boxed{\sin} \boxed{1/x}\end{aligned}$$

ARKUSFUNKTIONEN

INV – UMKEHRFUNKTION – Stellt man **INV** einer anderen Taste voran, wird die Funktion dieser Taste umgekehrt. In Verbindung mit den trigonometrischen Funktionen erhält man die Arkusfunktionen, zum Beispiel $\arcsin x$ ($\sin^{-1} x$) über die Tastenfolge **INV** **2nd** **sin**.

Bei den Arkusfunktionen wird der Winkel berechnet, dessen Funktionswert angezeigt ist. Der größte Winkel, der aus einer Arkusfunktion resultiert, ist 180 Grad (π Radiant oder 200 Gon). Da diese Funktionen viele Winkeläquivalente haben, d. h., $\arcsin .5$ für 30° , 150° , 390° etc., wird der Winkel aus jeder Funktion mit folgenden Einschränkungen wiedergegeben:

ARKUSFUNKTION

$\arcsin x$
 $\arcsin (-x)$
 $\arccos x$
 $\arccos (-x)$
 $\arctan x$
 $\arctan (-x)$

BEREICH DES RESULTIERENDEN WINKELS

0 bis 90° , $\pi/2$ Radiant oder 100 g
0 bis -90° , $\pi/2$ Radiant oder -100 g
0 bis 90° , $\pi/2$ Radiant oder 100 g
 90° bis 180° , $\pi/2$ bis π Radiant oder 100 bis 200 g
0 bis 90° , $\pi/2$ Radiant oder 100 g
0 bis -90° , $-\pi/2$ Radiant oder -100 g

Für $\arcsin x$ und $\arccos x$ gilt: $-1 \leq x \leq 1$; außerhalb dieses Definitionsbereiches blinkt die Anzeige.

Beispiel: $\pi/4 + \tan^{-1}(.2\pi) = 1.34638028$

Taste	Anzeige
2nd Rad	0
2nd π \div	3.141592654
4 +	.785391634
(.2 X 2nd π)	.6283185307
INV 2nd \tan	.5609821161
=	1.34638028

Die Wahl der Winkleinheit Radiant kann an beliebiger Stelle vor der Folge **INV** **2nd** **\tan** erfolgen. Dennoch empfiehlt es sich, die Winkleinheit grundsätzlich zu Beginn einer Aufgabe zu wählen, um sicher zu sein, der Winkelmodus ist korrekt eingestellt, ehe man sich auf die Eingabe des Problems konzentriert. Gleich, wann der Winkelmodus gewählt wird, er wirkt immer nur auf Winkelberechnungen.



Die Arkusfunktionen der anderen trigonometrischen Funktionen können wie folgt berechnet werden:

$$\begin{aligned}\operatorname{arccot} &= \boxed{1/x} \boxed{\text{INV}} \boxed{2\text{nd}} \boxed{\tan} \\ \operatorname{arcsec} &= \boxed{1/x} \boxed{\text{INV}} \boxed{2\text{nd}} \boxed{\cos} \\ \operatorname{arccsc} &= \boxed{1/x} \boxed{\text{INV}} \boxed{2\text{nd}} \boxed{\sin}\end{aligned}$$

ALTGRAD–RADIANT–NEUGRAD–(GON) UMRECHNUNGEN

Häufig müssen die Werte für Winkel von einer Maßeinheit auf eine andere umgerechnet werden. Verwenden Sie die Umrechnungsfaktoren in der folgenden Tabelle:

VON \ IN	GRAD	RADIANT	GON
GRAD		$\times \frac{\pi}{180}$	$\div 0.9$
RADIANT	$\times \frac{180}{\pi}$		$\times \frac{200}{\pi}$
GON	$\times 0.9$	$\times \frac{\pi}{200}$	

Diese Operationen können bei jeder Winkelmodus-Einstellung des Rechners durchgeführt werden.

Beispiel: Rechnen Sie 120 Grad in Radiant und Gon um.

Taste	Anzeige	Bemerkungen
120 $\boxed{\times}$ $\boxed{2\text{nd}}$ $\boxed{\pi}$ $\boxed{\div}$	376.9911184	
180 $\boxed{=}$ $\boxed{\times}$	2.094395102	Radiant
200 $\boxed{\div}$ $\boxed{2\text{nd}}$ $\boxed{\pi}$ $\boxed{=}$	133.3333333	Gon
$\boxed{\times}$	133.3333333	Gon
.9 $\boxed{=}$	120.	Grad

Da diese Umrechnungen vom Winkelmodus des Rechners unabhängig sind, ist äußerste Vorsicht angebracht, wenn diese Ergebnisse Basis für weitere Berechnungen sind. Die Winkleinheit muß extra gewählt werden, damit sie den Einheiten der Ergebnisse entspricht.



Ganzzahlfunktion und Absolutwert

2nd **Int** — GANZZAHLFUNKTION (integer) – Der Bruchteil der Zahl in der Anzeige wird mit dieser Anweisung gelöscht. **INV** **2nd** **Int** löscht den ganzzahligen Teil einer Zahl im Anzeigeregister.

2nd **|x|** — ABSOLUTWERT – Der Wert im Anzeigeregister wird positiv.

Beispiel: Ermitteln Sie den Absolutwert und den ganzzahligen Teil von $-13/5$.

Taste	Anzeige
13 +/- ÷	-13.
5 =	-2.6
2nd Int	-2.
2nd x 	2.

Besonders in Programmierfolgen erweisen sich diese Funktionen als wertvoll.

Beachten Sie, daß der Ganzzahlfunktion der Wert im Anzeigeregister und nicht der ausgewiesene Wert zugrundeliegt. Wenn man also **2nd** **Int** drückt, und der Inhalt des Anzeigeregisters 4.999999999 beträgt (in der Anzeige auf 5 gerundet), dann bleibt als ganze Zahl 4 in der Anzeige. Um hier den Wert 5 in der Anzeige zu erhalten, drückt man vor **2nd** **Int** zuerst **EE** **INV** **EE**, um die nicht angezeigten Stellen zu eliminieren. In der Exponentialform wird durch die Ganzzahl taste nur der tatsächliche Bruchteil einer Zahl entfernt, nicht der scheinbare Bruchteil. Bei 1.2345×10^3 **2nd** **Int** ergibt sich zum Beispiel 1.234×10^3 . Tatsächlich wird 1234.5 zu 1234.

Quadrat und Quadratwurzel

x² — QUADRAT – Berechnung des Quadrats der Zahl im Anzeigeregister. Ist $|x| \geq 10 \pm^{50}$, blinkt die Anzeige.

√x — QUADRATWURZEL – Berechnung der Quadratwurzel der Zahl im Anzeigeregister. Bei negativem x blinkt die Anzeige.

Beispiel: $[\sqrt{3.1452 - 7 + (3.2)^2}]^{1/2} = 2.239078197$

Taste	Anzeige
CLR (0.
3.1452 √x -	1.773471173
7 +	-5.226528827
3.2 x²	10.24
)	5.013471173
√x	2.239078197



Potenzen und Wurzeln

[y^x] – POTENZEN – Der Wert des Anzeigeregisters, y, wird in die x-te Potenz erhoben. Die Eingabefolge ist y **[y^x]** x, gefolgt von einer Operations- oder der Gleichheitstaste. Bei y < 0 blinkt die Anzeige.

[INV] [y^x] – WURZELN – ($\sqrt[x]{y}$ oder $y^{1/x}$) – Berechnung der x-ten Wurzel des Wertes y im Anzeigeregister. Die Eingabefolge ist y **[INV] [y^x]** x, gefolgt von einer Operations- oder der Gleichheitstaste. Bei y < 0 und x = 0 oder y = 0 und x < 0 blinkt die Anzeige.

Diese mathematischen Funktionen wirken nicht unmittelbar auf das Anzeigeregister. Sie erfordern die Eingabe eines zweiten Wertes, gefolgt von einer Operation, ehe die Funktion ausgeführt werden kann.

Beispiel: $\sqrt[3]{2.36^{23}} = .9362893421$

Taste	Anzeige	Bemerkungen
2.36 [y^x]	2.36	Eingabe von y für y^x
.23 [+/-]	-0.23	Eingabe von x für y^x
[INV] [y^x]	.8207865654	Berechnung von y für $\sqrt[x]{y}$
3 [=]	.9362893421	Eingabe von x für $\sqrt[x]{y}$ und Ergebnis

Die y^x -Funktionen werden auf logarithmischem Weg ermittelt. Nach den mathematischen Definitionen ergeben sich für die verschiedenen x- und y- Kombinationen die folgenden Wirkungen:

Funktionsverhalten			
y	x	y^x	$\sqrt[x]{y}$
0	0	1.	"1."
0	-x	"9.9999999 99"	"9.9999999 99"
0	x	0.	0.
1	0	1.	"1."
y	0	1.	"9.9999999 99"
-1	0	"1."	"1."
-y	0	"1."	"9.9999999 99"
-y	$\pm x$	" y ^{$\pm x$} "	" $\sqrt[x]{ y }$ "



Speichermöglichkeiten

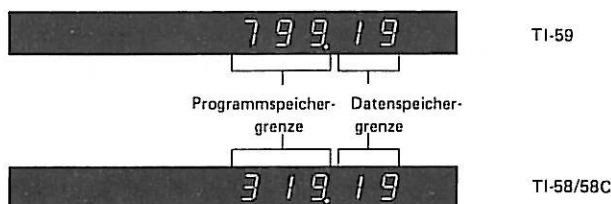
Die adressierbaren Datenspeicher-Register ermöglichen es, Daten für spätere Verwendung zu speichern oder zu akkumulieren. Diese Speicherbereiche werden im allgemeinen als Datenspeicher oder als Datenregister bezeichnet, als Gegensatz zum Programmspeicher, in den die Programme eingebracht sind. Die Speichertasten beeinflussen keine laufenden Berechnungen und können daher an jeder Stelle verwendet werden. Auch der Speicherinhalt des TI-58C geht nicht verloren, wenn das Gerät abgeschaltet wird, oder wenn die Batterieversorgung für kurze Zeit ausfällt.

Normalerweise ist es gleichgültig, welche Werte in welchem Datenregister abgespeichert werden. Nur bei Berechnungen von Mittelwert und Standardabweichungen werden die Register 01 bis 06 intern belegt. Wenn Werte erhalten bleiben sollen und Sie führen diese statistischen Berechnungen durch, müssen Sie andere Speicherregister als 01 bis 06 verwenden. Wenn Sie sehr viele Datenregister für die Speicherung benutzen, werden Sie eine Art Buchführung benötigen, um sich zu merken, welche Werte in welchen Registern gespeichert sind.

Wahl der Speicherkapazität (Speicherbereichsverteilung)

In diesem Kapitel finden Sie nach den Informationen über den TI-59 die Angaben über den TI-58/58C in Klammern und fettgedruckt.

Wenn der Rechner erstmals eingeschaltet wird, sind 60 (30) Register für die Datenspeicherung reserviert. Die Datenregister sind Teil des gesamten Speicherbereichs, wo sowohl Programme als auch Daten gespeichert werden. Mit der Möglichkeit, diesen Speicherbereich in Gruppen von je 10 Registern aufzuteilen, kann das Verhältnis Programmspeicher zu Datenspeicher Ihrem Bedarf entsprechend modifiziert werden. Geben Sie die Anzahl der Zehnergruppen ein, die Sie für die Datenspeicherung benötigen und drücken Sie **2nd** **Op** 17, zum Beispiel für 20 Datenregister 2 **2nd** **Op** 17. Diese Verteilung wird in der Anzeige wie folgt dargestellt:



Damit wird angezeigt, daß für die Datenspeicherung 20 Register, 00 – 19, zur Verfügung stehen, und daß der Programmspeicherbereich 800 (320) Speicherplätze umfaßt. Sonderformen der Anzeige, d. h. Festkomma-Einstellung, Exponentialform und technische Notation müssen vor der Speicherbereichsverteilung aufgehoben werden. Jedes nicht für die Datenspeicherung verwendete Register erhöht den Programmspeicher um 8 Speicherplätze.

Mit der Tastenfolge **2nd** **Op** 16 kann die augenblickliche Verteilung jederzeit geprüft und in der oben dargestellten Form angezeigt werden. Mehr über die Speicherbereichsverteilung siehe „Speicherkapazität und Verteilung“ auf Seite V-42.

Da bis zu 100 (60) Datenregister belegt werden können, muß unmittelbar nach einer speicherbezogenen Anweisung die zweistellige Registeradresse XX folgen. Unterbleibt die Eingabe der Speicheradresse, blinkt der augenblickliche Wert in der Anzeige. Eine Kurzformadressierung mit einer einstelligen Adresse ist jedoch dann möglich, wenn die Adresse kleiner 10 ist, und eine nichtnumerische Eingabe folgt.



Löschen des Datenspeichers

2nd CMs — DATENSPEICHERLÖSCHUNG (clear data memory) — Anweisung, alle Datenspeicherregister in der augenblicklich festgelegten Verteilung zu löschen.
Diese Taste hat keinen Einfluß auf das T-Register, noch auf den Programmspeicher, die Speicherbereichsverteilung, die Anzeige oder auf laufende Berechnungen.

Speicherung und Aufruf von Daten

STO XX — SPEICHERUNG — Der Wert im Anzeigeregister wird im Register XX gespeichert. Ein früher gespeicherter Wert in diesem Register wird dabei gelöscht.

RCL XX — AUFRUF — Aufruf und Anzeige des im Datenregister XX gespeicherten Wertes; dieser Wert bleibt dabei im Register XX erhalten. Eine aufgerufene Zahl kann als Zahleneingabe für einen beliebigen mathematischen Ausdruck verwendet werden.

Beispiel: Speicherung und Aufruf der Zahl 3.012 im Register 22

Taste	Anzeige
3.012 STO 22	3.012
CLR	0
RCL 22	3.012

Die Anwendung der Speicheranweisungen kann Tastenarbeit ersparen, wenn man vielziffrige Zahlen speichert, die mehrere Male verwendet werden.

Beispiel: Berechnen Sie $3x^2 - x - 7.1$ für $x = 2.9467281$

Taste	Anzeige
CLR 3 X	3.
2.9467281 STO 12	2.9467281
x² -	26.04961949
RCL 12	2.9467281
- 7.1 =	16.00289139

Der vielziffrige Wert von x mußte nur einmal eingegeben werden. Speicherung und Aufruf hatten keine Wirkung auf laufende Berechnungen.



Die Datenspeicher sind auch geeignet, Zwischenergebnisse und sich wiederholende Zahlen zu halten.

Beispiel: Berechnen Sie $\frac{\sin(3x/2) - \cos(3x/2)}{x}$ für $x = 20.6821776$ Grad

Taste	Anzeige	Bemerkungen
2nd CHS ((3 X	3.	
20.6821776 STO 14	20.6821776	x wird in Register 14 gespeichert
÷ 2) STO 17	31.0232664	3x/2 wird in Register 17 gespeichert
2nd SIN -	.5153861069	
RCL 17	31.0232664	Aufruf von 3x/2 aus Register 17
2nd COS) ÷	-.3415719789	
RCL 14	20.6821776	Aufruf von x aus Register 14
=	-.0165152812	Ergebnis

Beim Versuch, ein Register zu belegen, das in der augenblicklichen Verteilung nicht bereitgestellt ist, blinkt die Anzeige.

Direkte Registerarithmetik

Eine angezeigte Zahl kann jederzeit während einer Berechnung und ohne jeden Einfluß auf sie gespeichert werden. Darüber hinaus können Sie mit dem Anzeigewert und einem Wert in einem Datenregister Addition, Subtraktion, Multiplikation und Division durchführen. Das Anzeigeregister selbst wird nicht geändert. Eine blinkende Anzeige nach einer dieser Operationen bedeutet, daß Sie die Kapazitätsgrenze des Rechners in diesem Register überschritten haben, (vorausgesetzt, daß nicht eine Registeradresse aufgerufen wurde, die außerhalb des Bereichs der augenblicklichen Verteilung liegt).

SUM XX – SPEICHERADDITION – Der Wert des Anzeigeregisters wird zum Inhalt des Datenregisters XX addiert und das Ergebnis in XX gespeichert.

INV **SUM** XX – SPEICHERSUBTRAKTION – Der Wert des Anzeigeregisters wird vom Inhalt des Datenregisters XX subtrahiert und das Ergebnis in XX gespeichert.

2nd **Prd** XX – SPEICHERMULTIPLIKATION – Der Inhalt des Datenregisters XX wird mit dem Wert im Anzeigeregister multipliziert und das Ergebnis in XX gespeichert.

INV **2nd** **Prd** XX – SPEICHERDIVISION – Der Inhalt des Datenregisters XX wird durch den Wert im Anzeigeregister dividiert und das Ergebnis in XX gespeichert.

Mit dieser Speicherarithmetik erübrigen sich lange Aufruf-Operationen und Tastenfolgen für erneute Speicherung.



Beispiel: Berechnen Sie $x^2 + 9$ für $x = -1, 2$ und 3 und addieren Sie die Ergebnisse.

Taste	Anzeige	Speicher 3
1 \pm/\mp x^2 $+$	1.	0
9 $=$ STO 03	10.	10
2 x^2 $+$	4.	10
9 $=$ SUM 03	13.	23
3 x^2 $+$	9.	23
9 $+$	18.	23
RCL 3	23.	23
$=$	41.	23

Beachten Sie, daß die erste Berechnung im Register 03 mit der Taste **STO** abgespeichert wurde. Dieses Verfahren ist bei der direkten Registerarithmetik empfehlenswert, um sicherzustellen, daß nur die benötigten Werte in dem speziellen Register akkumuliert werden. Die Taste **STO** löscht jeden vorhergehenden Inhalt dieses Registers, ehe der neue Wert gespeichert wird.

Beispiel: Der Prozentsatz von Studenten, die an einem College jedes Jahr ihren Abschluß machen, beträgt 76,8% im ersten Jahr, 81,3% im zweiten, 92,2% im dritten und 95,9% im letzten Jahr. Wie hoch ist der Prozentsatz derer, die graduieren, und welcher Prozentsatz schließt die Jahre 3 und 4 ab?

Taste	Anzeige
.768 X	0.768
.813 X	0.624384
.922 STO 11 X	0.575682048
.959 2nd Prd 11 $=$	0.552079084
RCL 11	0.884198

Etwa 55% der Studenten, die in dieses College eintreten, graduieren. Mehr als 88% derer, die das 3. Jahr und das Studienjahr erreichen, graduieren.



SPEICHER-/ANZEIGE AUSTAUSCH

2nd **fix** XX – SPEICHERAUSTAUSCH – Der Inhalt des Datenregisters XX wird mit dem Anzeigehalt vertauscht. Der Anzeigewert wird gespeichert und der zuvor gespeicherte Wert angezeigt.

Die Austausch Taste hat mehrere Zwecke zusätzlich zur Einsparung von Tastenbetätigungen. Man kann zwei Ergebnisse überprüfen, ohne dabei eines zu verlieren. Auch Zahlen können kurzfristig in XX gespeichert und bei Bedarf weiterverwendet werden.

Beispiel: Berechnen Sie $A^2 + AB + 2B^2$ für $A = .258963$ und $B = 1.255632$

Taste	Anzeige	Bemerkungen
.258963 STO 13	0.258963	A wird in Register 13 gespeichert
x² + 1.255632 X	1.255632	Eingabe von B
2nd fix 13	0.258963	B wird gespeichert, A aufgerufen
+ 2 X	2.	
RCL 13	1.255632	Aufruf von B aus Register 13
x² =	3.545447504	Ergebnis



SPEZIELLE STEUEROOPERATIONEN

Eine Reihe von Operationen, die über die Taste **Op** durchgeführt werden, können die Leistung des Rechners noch beträchtlich erweitern. Einige dieser Operationen sind in jeder Betriebsart möglich, während andere nur in einem speziellen Modus oder in Verbindung mit dem Drucker PC-100A, PC-100B oder PC-100C wirksam werden.

Jede der Steuerooperationen wird mit der Tastenfolge **2nd Op** nn eingeleitet, wobei nn der entsprechende, zugeordnete 2-stellige Kode ist (Kurzformadressierung ist möglich). Kurzbeschreibungen der Kodes finden Sie in der folgenden Aufzählung, die vollständigen Informationen erhalten Sie auf den weiteren Seiten. Wenn nn > 39 (40 für den TI-58C) blinkt die Anzeige.

Kode nn	Funktion
00*	Vorbereiten des Druckerregisters
01*	Eingabe von 10 Ziffern in die Anzeige als 5 alphanumerische Kodes für das äußerste linke Viertel der Druckzeile;
02*	Eingabe von 10 Ziffern in die Anzeige als 5 alphanumerische Kodes für das innere linke Viertel der Druckzeile;
03*	Eingabe von 10 Ziffern in die Anzeige als 5 alphanumerische Kodes für das innere rechte Viertel der Druckzeile;
04*	Eingabe von 10 Ziffern in die Anzeige als 5 alphanumerische Kodes für das äußerste rechte Viertel der Druckzeile;
05*	Der Inhalt des Druckerregisters wird ausgedruckt.
06*	Ausdruck der letzten 4 Schriftzeichen von OP 04 mit dem augenblicklichen Anzeigewert;
07*	Plotten (aufzeichnen) in den Spalten 0 bis 19 nach Angaben aus der Anzeige;
08*	Auflistung der Labels, die derzeit im Programmspeicher verwendet werden;
09	Belegen des Programmspeichers mit einem zuvor angegebenen Software-Programm;
10	Anwendung der Signumfunktion auf den Anzeigeregisterwert;
11	Berechnung der Varianz
12	Berechnung der Steigung und des Schnittpunktes
13	Berechnung des Korrelationskoeffizienten
14	Berechnung neuer y-Werte (y') für ein x in der Anzeige
15	Berechnung neuer x-Werte (x') für ein y in der Anzeige
16	Anzeige der augenblicklichen Speicherbereichsverteilung
17	Änderung der Speicherbereichsverteilung
18	Wenn in einem Programm keine Fehlerbedingung existiert, wird Flag 7 gesetzt.
19	Wenn in einem Programm eine Fehlerbedingung existiert, wird Flag 7 gesetzt
20 – 29	Inkrement eines Datenregisters 0 – 9 um 1
30 – 39	Dekrement eines Datenregisters 0 – 9 um 1
40	Flag 7 wird gesetzt, wenn der Drucker angeschlossen ist (nur TI-58C).

* Nur in Verbindung mit dem PC-100A, PC-100B oder PC-100C.



Druckerfunktionen — **Op** 00-08

Diese Steueroperationen wirken nur bei angeschlossenem Drucker PC-100A, PC-100B oder PC-100C. Durch den Ausdruck einer "Hardcopy" der Rechenergebnisse erhöht der Drucker die Flexibilität Ihres Rechners. Die Steueroperationen erweitern zugleich die Druckerfunktionen durch Ausdruck alphanumerischer Informationen, das Plotten von Daten und die Auflistung der im Programm verwendeten Labels mit der entsprechenden Programmspeicheradresse. Die Anwendung dieser speziellen Steueroperationen finden Sie im Abschnitt VI-7, Druckersteuerung, genau erläutert.

Analyse eines Softwareprogramms (Programmübernahme) — **Op** 09

Die Tastenfolge **2nd** **Pgm** mm **2nd** **Op** 09 übernimmt ein Softwareprogramm mm in den Programmspeicher. Damit kann das Programm analysiert und redigiert, im einzelnen genau überprüft und auf den individuellen Bedarf abgestimmt werden. Vorsicht bei den Auswirkungen von Korrekturen auf absolute Adressen und kombinierte Tastenkodes. Wenn ein Softwareprogramm in den Programmspeicher eingebracht wird, ersetzt es ab Speicherplatz 000 alle zuvor gespeicherten Befehle. Die Übertragung eines Programms vom Programmspeicher in den Softwaremodul ist nicht vorgesehen. Ist ein Programm zu umfangreich für den Programmspeicher, sei es wegen der momentanen Verteilung oder wegen des gesamten Speicherbereichs selbst, wird es nicht übernommen und die Anzeige blinkt. Die Anzeige blinkt auch dann, wenn der derzeit eingesetzte Modul das Programm mm nicht enthält.

Signum-Funktion — **Op** 10

Mit **Op** 10 wird die Signum-Funktion auf den augenblicklichen Wert „x“ im Anzeigeregister angewandt, wobei die Anzeige wie folgt reagiert.

Anzeigeregister- wert x	Anzeige- reaktion
$x > 0$	1.
$x = 0$	0.
$x < 0$	- 1.

Statistik — **Op** 11 — 15

Die Steueroperationen 11 — 15 beziehen sich auf statistische Berechnungen, die im folgenden Kapitel, „Umrechnungen u. Statistik“, behandelt werden.



Speicherbereichsverteilung — **Op** 16 — 17

Mit **2nd** **Op** 16 wird die augenblickliche Verteilung zwischen Programmspeicher und Datenspeicher angegeben. Der letzte belegbare Programmspeicherplatz sowie das Datenregister mit der höchsten Nummer werden, durch ein Dezimalkomma getrennt, ausgewiesen. Die ursprüngliche Verteilung beim TI-59 liegt bei 479,59, beim TI-58 bei 239,29. Der TI-58C hält jedoch die gewählte Bereichsverteilung gespeichert, bis sie geändert oder bis das Batteriepaket längere Zeit entfernt wird. Um die Verteilung des Speicherbereichs erneut zu ändern, gibt man die Anzahl der benötigten Zehnergruppen der Datenregister (0 – 10) ein, und drückt **2nd** **Op** 17. Die Verteilung wird dann wie oben angezeigt. Siehe hierzu die Kapitel „Speichermöglichkeiten“ und „Speicherkapazität und Verteilung“, Seite V - 42.

Testoperationen — **Op** 18 — 19

Mit den Operationen 18 und 19 wird der Fehlerstatus eines ablaufenden Programms kontrolliert. Als Bestandteil eines Programms setzt **2nd** **Op** 18 Flag 7, wenn keine Fehlerbedingung vorliegt. **2nd** **Op** 19 setzt Flag 7, wenn eine Fehlerbedingung vorliegt. Flag 7 kann mit dem „if flag“ Testbefehl überprüft werden. Das Testergebnis ist dann die Grundlage für weitere Entscheidungen. Flag 7 wird nicht verändert, wenn die Testbedingung nicht erfüllt ist. Weitere Informationen siehe „Flags“ im Programmierabschnitt, Seite V — 65

Inkrement u. Dekrement von Datenregistern — **Op** 20 — 29/30 — 39

Mit dem **Op**- Befehl kann der Inhalt eines der Datenregister 0 – 9 um 1 erhöht oder vermindert werden.

Mit **2nd** **Op** 2n wird Register n um 1 erhöht, wobei n ein Datenregister zwischen 0 und 9 ist.

Mit **2nd** **Op** 3n wird Register n um 1 vermindert, wobei n ein Datenregister zwischen 0 und 9 ist.

Wenn man eine dieser Tastenfolgen drückt oder wenn sie in einem Programm enthalten sind, wird der Inhalt von Register n unabhängig von dem darin gespeicherten Wert um 1 verändert.

Beispiel: **2nd** **Op** 34 subtrahiert 1 vom Inhalt des Datenregisters 4.

Druckertest — **Op** 40 (nur für den TI-58C).

Wenn der TI-58C an einen PC-100A, PC-100B oder einen PC-100C angeschlossen wird, wird bei der Folge **2nd** **Op** 40 Flag 7 gesetzt.



Umrechnungen und Statistik

Mehrere häufig verwendete mathematische Folgen sind fest in den Rechner einprogrammiert. Diese Operationen dienen der optimalen Effektivität des Rechners, da sich die Tastenarbeit für die Durchführung dieser iterativen Folgen erheblich reduziert.

Umrechnungen

Umrechnungen zwischen den Koordinatensystemen (polar – rechtwinklig) sind mit Ihrem Rechner jederzeit möglich. Ebenso können Winkel, die in Grad – Minuten – Sekunden ausgedrückt sind, in Dezimalgrad umgeformt werden und umgekehrt.

WINKELUMRECHNUNGEN

[2nd] [DMS] – GRAD–MINUTEN–SEKUNDEN IN DEZIMALGRAD – Umrechnung eines in Grad–Minuten–Sekunden angegebenen Winkels in seine Dezimalgradentsprechung. **[INV] [2nd] [DMS]** kehrt diese Umrechnung um. Minuten und Sekunden können jeweils eine beliebige zweistellige Zahl sein. Diese Umrechnung basiert nur auf dem angezeigten Wert. Bei der Eingabe für Grad–Minuten–Sekunden wird zwischen die Gradzahl und die Minuten ein Komma gesetzt, also GG.MMSSss. Gelegentlich werden wegen der Rundung bei der Umrechnung in Grad–Minuten–Sekunden 60 Minuten oder 60 Sekunden angezeigt. Dies bedeutet, daß auf die nächsthöhere Gradzahl (oder bei Sekunden auf die nächste Minute) gerundet werden muß.

Taste	Anzeige	Bemerkungen
[2nd] [DMS]	47.2202	GG.dddd
[INV] [2nd] [DMS]	47.131272	GG.MMSSss

GG steht für Grad und dd für den Dezimalbruchteil eines Grads. Mit MM werden die Minuten und mit SSss die Sekunden und die Bruchteile der Sekunden symbolisiert. Bei den trigonometrischen Funktionen werden nur Dezimalgrad akzeptiert, nicht Minuten und Sekunden. Also muß zuvor für alle die Winkel die D.MS- Umrechnung durchgeführt werden, die in Grad–Minuten–Sekunden ausgedrückt sind. Achten Sie darauf, daß Sie für Minuten und Sekunden jeweils 2 Stellen verwenden.

Beispiel: $5^{\circ} 4' 3''$ muß als 5.0403 eingegeben werden, damit der Wert korrekt interpretiert wird.

Die D.MS- Umrechnung ist auch auf die Umrechnung von Stunden–Minuten–Sekunden in das Dezimaläquivalent anwendbar.

Polar-/rechtwinklige Koordinaten-Umrechnungen

[x↔t] – X-t-AUSTAUSCH – Der Wert des Anzeigeregisters x wird gegen den Wert t im T-Register ausgetauscht. Diese Taste wird für die Eingabe bei Koordinatenumrechnungen, in der Statistik und bei bestimmten Testvorgängen in der Programmierung verwendet.

Dieses T-Register ist unabhängig von allen anderen Speicherbereichen, funktioniert aber im wesentlichen wie ein Datenregister. Es ist nur extern über die Taste **[x↔t]** zugänglich, mit der der Wert im Anzeigeregister in das T-Register eingebracht wird, und umgekehrt der Inhalt des T-Registers in das Anzeigeregister und in die Anzeige.

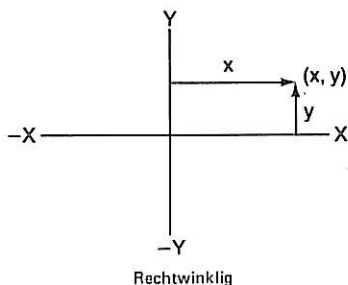
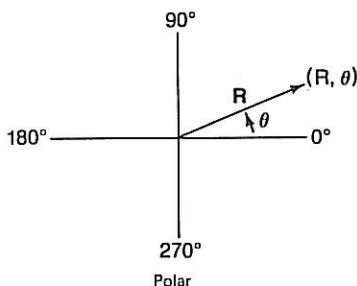
(**[CP]** löscht das T-Register, wenn im Programm vorhanden)

[2nd] [P↔R] – POLAR/RECHTWINKLIG – Umrechnung von Polarkoordinaten in rechtwinklige Koordinaten. Mit **[INV] [2nd] [P↔R]** werden rechtwinklige Koordinaten in Polarkoordinaten umgeformt. Intern benötigt man bei diesen Berechnungen 4 unvollständige Operationen.

Diese Berechnungen benötigen 4 unvollständige Operationen und eine Unterprogrammebene.



Die polar-rechtwinklige Umformung wird verwendet, um Polarkoordinaten, die durch den Abstand r und den Richtungswinkel θ einen beliebigen Punkt beschreiben, in rechtwinklige Koordinaten umzuformen, die einen Punkt durch die Abszisse x und die Ordinate y in rechten Winkeln gegeneinander beschreiben.



Tastenfolgen für die Koordinatenumrechnungen:

Polar – Rechtwinklig

Eingabe von „R“

Taste $\boxed{\times \div}$

Eingabe von „ θ “

Taste $\boxed{2nd} \boxed{P \rightarrow R}$ zur Anzeige von „y“

Taste $\boxed{\times \div}$ zur Anzeige von „x“

Rechtwinklig – Polar

Eingabe von „x“

Taste $\boxed{\times \div}$

Eingabe von „y“

Taste $\boxed{INV} \boxed{2nd} \boxed{P \rightarrow R}$ zur Anzeige von „ θ “

Taste $\boxed{\times \div}$ zur Anzeige von „R“

Überprüfen Sie die richtige Winkleinheit für θ bei Eingabe oder Rechenergebnis.

Der Wert für θ , der sich aus der Umformung von rechtwinkligen Koordinaten in Polarkoordinaten ergibt, liegt im Bereich

$$\left. \begin{array}{l} -90^\circ \\ -\pi/2 \text{ rad} \\ -100 \text{ g} \end{array} \right\} \leq \theta < \left\{ \begin{array}{l} 270^\circ \\ 3\pi/2 \text{ rad} \\ 300 \text{ g} \end{array} \right.$$

Berechnete Winkel, die im vierten Quadranten liegen, werden also als negative Winkel ausgewiesen.

Diese Umrechnung erfordert die Wahl des Winkelmodes, um die gewünschten Winkleinheiten für Dateneingabe und -ausgabe festzulegen.

Beispiel: Formen Sie in rechtwinklige Koordinaten um: $r = 5$, $\theta = 30^\circ$

Winkleinheit: Grad

Taste

5 $\boxed{\times \div}$

30 $\boxed{2nd} \boxed{P \rightarrow R}$

$\boxed{\times \div}$

Anzeige

0.

2.5

4.330127019

Bemerkungen

Eingabe von „r“

Eingabe von „ θ “, Anzeige des Wertes von „y“

Wert von „x“



Beispiel: Rechnen Sie in Polarkoordinaten um (Radiant):

$$x = 3, y = 4$$

Taste	Anzeige	Bemerkungen
CLR 2nd Rad	0	Einstellung des Winkelmodus auf Radiant
3 x₀₁	0.	„x“ wird gespeichert
4 INV 2nd P→R	0.927295218	„y“ wird eingegeben, der Wert von „θ“ wird in Radiant angezeigt
x₀₁	5.	Wert von „R“

P→R und Grad-Umrechnungen verwenden jeweils eine Unterprogrammebene und bis zu 4 unvollständige Operationen.

Polar-rechtwinklige Umrechnungen sind von besonderem Vorteil bei der Vektorrechnung.

Statistik

Oft erscheint es wünschenswert, eine Variable durch eine andere auszudrücken, auch wenn die Variablen nicht als Funktionen voneinander definiert werden können. Diese Variablen können als Datenpunkte graphisch dargestellt werden, wobei im Diagramm die sogenannte unabhängige Variable auf der x-Achse eingetragen wird, und die abhängige Variable auf der y-Achse. Die Gesamtheit der Datenpunkte kann dann durch Berechnung des Mittelwerts, der Standardabweichung und der Varianz jeder Variablen analysiert werden. Den Datenpunkten kann eine Gerade angepaßt werden (lineare Regression), wobei der Anwender den Schnittpunkt und die Steigung dieser Geraden berechnet. Von dieser Ausgangsposition ist es möglich, weitere Datenpunkte zu interpolieren oder zu extrapolieren, und ein Korrelationskoeffizient kann Aufschluß darüber geben, wie gut die Gerade den Datenpunkten angepaßt ist.

Dateneingabe

2nd **Pgm** **1** **SBR** **CLR** — Vorbereitung des Rechners auf statistische Berechnungen. Dabei werden die Datenregister 01 bis 06 sowie das T-Register auf Null gestellt. Einer der Software-Module muß für diese Folge natürlich im Rechner eingesetzt sein. Andernfalls müssen Sie die Datenregister 01 bis 06 und das T-Register manuell oder durch geeignete Programmierung löschen.

x₀₁ **t** — x - t - AUSTAUSCH — Der Wert t des T-Registers wird gegen den Wert x des Anzeigeregisters ausgetauscht. In der Statistik erfolgt über diese Taste die Eingabe der unabhängigen Variablen.

2nd **Σ+** — STATISTISCHE SUMMIERUNG — Jedes Variablenpaar (x_i, y_i) wird in den Datenregistern 1 bis 6 akkumuliert. Mit **INV** **2nd** **Σ+** wird die Eingabe eines ungewollten Datenpunkts (Wertepaar) wieder rückgängig gemacht. Um eine Anordnung von Daten mit einer Variablen einzugeben, tastet man jeden Wert ein und drückt dann **2nd** **Σ+**. Eine falsche Eingabe wird dadurch gelöscht, daß man den ungewollten Wert (Paar) erneut eingibt, und dann **INV** **2nd** **Σ+** drückt. Nach jeder Eingabe (oder Löschung) wird die Anzahl der eingegebenen Werte angezeigt.



Zweidimensionale statistische Daten werden für jeden Datenpunkt (x_i, y_i) mit nachstehender Tastenfolge eingegeben.
 $i = 1, 2, 3, \dots, N$

x_i **[x=t]** y_i **[2nd]** **[Σ+]**

Die Nummer des Datenpunktes wird nach jeder Eingabe ausgewiesen. Um einen ungewollten Datenpunkt zu löschen, gibt man die falschen Werte für x und y erneut ein, und drückt dann **[INV]** unmittelbar vor **[2nd]** **[Σ+]**. Die Gesamtanzahl der bisher eingegebenen Datenpunkte, N , wird daraufhin automatisch um 1 vermindert.

Jeder Datenpunkt wird nach der Eingabe in den Datenspeicher-Registern 01 bis 06 wie folgt aufgenommen.

SPEICHERREGISTER	INHALT	
01	Σy	} abhängige Variable
02	Σy^2	
03	N	
04	Σx	} unabhängige Variable
05	Σx^2	
06	Σxy	

Daten, die bereits so gruppiert sind, können direkt in diese Register eingegeben werden und die Analyse kann sofort beginnen.

Der Rechner „SUMMIERT“ die eingehenden Werte in diesen Speicherregistern. Die Inhalte dieser Register müssen vorher gelöscht werden, andernfalls ergeben sich beim Akkumulieren der statistischen Daten Fehler. Stellen Sie zu Beginn der Dateneingabe mit der Tastenfolge **[2nd]** **[PgM]** **1** **[SBR]** **[CLR]** die Register 1 bis 6 und das T-Register auf Null.

Mittelwert, Varianz und Standardabweichung

Nach Eingabe aller Daten (oder bei einem beliebigen Zwischenpunkt nach Eingabe von 2 oder mehr Datenpunkten) können Mittelwert, Standardabweichung und Varianz für jedes Datenfeld errechnet werden.

[2nd] **[Σ]** – Berechnung und Anzeige des Mittelwertes der abhängigen Daten (y-Feld). Drücken Sie dann **[x=t]** zur Anzeige des Mittelwertes der unabhängigen Daten (x-Feld).

[INV] **[2nd]** **[Σ]** – Berechnung und Anzeige der Standardabweichung der abhängigen Daten (y-Feld). Drückt man anschließend **[x=t]**, wird die Standardabweichung der unabhängigen Daten (x-Feld) angezeigt.

[2nd] **[0p]** **11** – Berechnung und Anzeige der Varianz der abhängigen Daten (y-Feld). Drückt man anschließend **[x=t]** wird die Varianz der unabhängigen Daten (x-Feld) ausgewiesen.



Der Rechner verwendet folgende Gleichungen:

$$\text{Mittelwert des x-Feldes} = \bar{x} = \frac{\sum x}{N}$$

Mittelwert des y-Feldes =

$$= \bar{y} = \frac{\sum y}{N}$$

wobei N = Gesamtzahl der eingegebenen Datenpunkte

$$\text{Standardabweichung des x-Feldes} = \sigma_x = \left[\frac{\sum x^2 - \frac{(\sum x)^2}{N}}{N-1} \right]^{1/2}$$

$$\text{Standardabweichung des y-Feldes} = \sigma_y = \left[\frac{\sum y^2 - \frac{(\sum y)^2}{N}}{N-1} \right]^{1/2}$$

$$\text{Varianz des x-Feldes} = \sigma_x^2 = \frac{\sum x^2}{N} - \bar{x}^2$$

$$\text{Varianz des y-Feldes} = \sigma_y^2 = \frac{\sum y^2}{N} - \bar{y}^2$$

Zu Ihrem Vorteil haben Sie bei der Berechnung von Standardabweichung und Varianz die Möglichkeit, zwischen N- oder N-1 Gewichtung zu wählen. Die N-Gewichtung führt zu einer Maximum-Likelihood-Schätzfunktion, die im allgemeinen zur Beschreibung von Grundgesamtheiten verwendet wird. Die N-1 Gewichtung ist eine von systematischen Fehlern freie, unverzerrte Schätzfunktion und wird gewöhnlich für Stichproben benutzt.

Die Varianzfunktion verwendet die N-Gewichtung, die Standardabweichung die N-1 Gewichtung. Die Varianz ist als das Quadrat der Standardabweichung definiert.

Um also die Varianz mit N-1 Gewichtung zu ermitteln, drückt man $\text{INV} \text{ 2nd } \overline{x} \text{ } x^2$ und $\text{x} \div \text{t } x^2$. Die Standardabweichung mit N-Gewichtung wird über die Tasten $\text{INV} \text{ 2nd } \text{Op } 11 \sqrt{x}$ und $\text{x} \div \text{t } \sqrt{x}$ berechnet. Siehe die nachstehende Tabelle mit diesen Tastenfolgen:

Tastenfolgen

Funktion	Gewichtung	y-Feld	x-Feld
Mittelwert		$\text{2nd } \overline{x}$	$\text{x} \div \text{t}$
Standardabweichung	N	$\text{2nd } \text{Op } 11 \sqrt{x}$	$\text{x} \div \text{t } \sqrt{x}$
Varianz	N	$\text{2nd } \text{Op } 11$	$\text{x} \div \text{t}$
Standardabweichung	N-1	$\text{INV } \text{2nd } \overline{x}$	$\text{x} \div \text{t}$
Varianz	N-1	$\text{INV } \text{2nd } \overline{x} \text{ } x^2$	$\text{x} \div \text{t } x^2$



Für Daten mit einer Variablen wird die Taste Σx^2 nicht benötigt. Sie muß nur für Eingabe und Anzeige der Merkmale von unabhängigen Variablen (x-Feld) gedrückt werden. Die Speicherregister 01 bis 06 und das T-Register werden noch benutzt.

Beispiel: Analysieren Sie folgende Testwerte: 96, 81, 87, 70, 93, 77

Taste	Anzeige	Bemerkungen
2^{nd} Pgm 1 SBR CLR	0	Einleitung
96 2^{nd} $\Sigma +$	1.	erste Eingabe
81 2^{nd} $\Sigma +$	2.	zweite Eingabe
97 2^{nd} $\Sigma +$	3.	dritte Eingabe (falsch)
97 INV 2^{nd} $\Sigma +$	2.	Löschen d. 3. Eingabe
87 2^{nd} $\Sigma +$	3.	Korrekte 3. Eing.
70 2^{nd} $\Sigma +$	4.	4. Eingabe
93 2^{nd} $\Sigma +$	5.	5. Eingabe
77 2^{nd} $\Sigma +$	6.	6. Eingabe
INV 2^{nd} \bar{x}	9.879271228	Standardabweichung
2^{nd} \bar{x}	84.	Mittelwert
2^{nd} Op 11	81.33333333	Varianz
RCL 01	504.	gesamte Punktezah

Beachten Sie, daß die Standardabweichung zuerst berechnet werden kann, obwohl für die Bestimmung der Standardabweichung der Mittelwert benötigt wird.

Beispiel: Jemand bestellt eine Menge von Röhren, die auf 100 cm lange Stücke zusammengeschnitten sind. Die Genauigkeit der Länge soll überprüft werden, ebenso die Gleichförmigkeit, die $6.0 \text{ g/cm} \pm 0.01$ betragen soll. Der Test erfordert die gleichzeitige Analyse von 6 Stichproben.

Stichprobe	1	2	3	4	5	6
Länge (cm)	101.3	103.7	98.6	99.9	97.2	100.1
Gewicht (g)	609	626	586	594	579	605

Wie hoch ist das durchschnittliche Gewicht der genommenen Stichproben? Wie genau arbeitet die Schneidemaschine? Wie ist die Gleichförmigkeit der Stichproben?



Taste	Anzeige	Bemerkungen
2nd Pgm 1 SBR CLR	0	Einleitung der Berechnung
101.3 x_Δt	0.	Eingabe x_1
609 2nd Σ+	1.	Eingabe y_1
103.7 x_Δt	102.3	Eingabe x_2
626 2nd Σ+	2.	Eingabe y_2
98.6 x_Δt	104.7	Eingabe x_3
586 2nd Σ+	3.	Eingabe y_3
99.9 x_Δt	99.6	Eingabe x_4
594 2nd Σ+	4.	Eingabe y_4
97.2 x_Δt	100.9	Eingabe x_5
579 2nd Σ+	5.	Eingabe y_5
100.1 x_Δt	98.2	Eingabe x_6
605 2nd Σ+	6.	Eingabe y_6
2nd Σ_x	599.8333333	durchschnittl. y-Feld (Gewicht)
÷ x_Δt	100.1333333	durchschnittl. x-Feld (Länge)
=	5.990346205	durchschnittliche Gleichförmigkeit (g/cm)
INV 2nd Σ_x	17.05774509	Gewichtsabweichung
x_Δt	2.240238083	Längenabweichung

Das durchschnittliche Gewicht der Stichproben beträgt etwa 599.8 Gramm. Die Maschine schneidet die Röhren in der Länge auf 100.1 Zentimeter. Die Gleichförmigkeit ist besser als 5.99 Gramm/Zentimeter, also liegt innerhalb der akzeptablen Toleranz. Schließlich beträgt die Standardabweichung der Gewichte der verschiedenen Stücke etwa 17 Gramm mit einer durchschnittlichen Längenabweichung von etwa 2 1/4 Zentimeter.

Lineare Regression

2nd **Op** **12** – Der Schnittpunkt mit der y-Achse der an die Datenpunkte angepaßten Geraden wird berechnet und angezeigt. Drückt man **x_Δt** nach **2nd** **Op** **12**, wird die Steigung der Anpassungsgeraden ausgewiesen. Datenpunkte, die eine vertikale Gerade beschreiben, haben als Sonderfall keinen Schnittpunkt mit der y-Achse und eine unendliche Steigung. Für den Rechner ist dies eine unzulässige Operation. Durch Berechnung des Korrelationskoeffizienten (**2nd** **Op** **13**) wird eine vertikale oder horizontale Gerade ermittelt und die Anzeige blinkt.

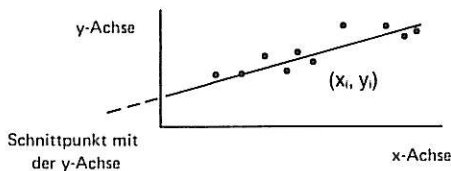
2nd **Op** **13** – Berechnung und Anzeige des Korrelationskoeffizienten der einzelnen Datenpunkte in bezug auf die Gerade, die diesen Punkten angepaßt ist. Der Wert liegt zwischen -1 und 1, mit perfekter Korrelation bei ± 1 . Ist die Steigung 0 oder unendlich, blinkt die Anzeige. Diese Bedingung kann im Programm mit **2nd** **Op** **19** und Programmflags gesteuert werden.

2nd **Op** **14** – Berechnung und Anzeige eines linearen Schätzwertes y' auf der Regressionsgeraden, der einer x-Eingabe über die Tastatur entspricht. Wenn die Eingabedaten zuvor eine vertikale Gerade beschreiben (unendliche Steigung), sollen neue y' -Werte nicht berechnet werden.

2nd **Op** **15** – Berechnung und Anzeige eines linearen Schätzwertes x' auf der Regressionsgeraden, der einer y-Eingabe über die Tastatur entspricht. Ist die Steigung 0, blinkt die Anzeige.



Die Dateneingabe für die lineare Regression ist identisch mit der für die Berechnungen von Mittelwert, Standardabweichung und Varianz, die auch hier durchgeführt werden können. Ist eine Datengruppe einmal eingegeben, können alle statistischen Funktionen zur Analyse herangezogen werden. Mit der linearen Regression kann die Beziehung einer Variablen zu einer anderen untersucht werden. Die lineare Regression wird hier nach der Kleinstquadratmethode durchgeführt, ein Verfahren, bei dem die Summe der Quadrate der Abweichungen der tatsächlichen Datenpunkte von der besten Anpassungsgeraden ein Minimum beträgt. Praktisch werden die Datenpunkte aufgezeichnet, und eine Gerade so eingepaßt, daß sie die Datenpunkte gleichförmig trennt. Die Quadratwurzel der Quadrate ihrer vertikalen Abstände wird minimiert.



Diese Gerade wird durch $y = mx + b$ beschrieben, wobei m die Steigung der Geraden und b ihr Schnittpunkt mit der y -Achse ist.

Da die Daten selten vollkommen linear sind, kann man ermitteln, wie gut die angepaßte Gerade tatsächlich den Daten annähert ist. Dieses Maß ist der Korrelationskoeffizient, der aus den Parametern der Variablen und der linearen Gleichung errechnet werden kann.

Die Steigung und der Schnittpunkt mit der y -Achse der Regressionsgeraden sind wie folgt definiert.

$$\text{Steigung} = m = \frac{\sum xy - \frac{\sum x \sum y}{N}}{\sum x^2 - \frac{(\sum x)^2}{N}}$$

$$\text{Schnittpunkt mit der } y\text{-Achse} = b = \frac{\sum y - m \sum x}{N}$$

$$\text{Der Korrelationskoeffizient} = R = \frac{m \sigma_x}{\sigma_y}$$

Die Vorhersage weiterer Datenpunkte ist möglich. Man wählt einfach einen neuen Wert für x oder y und der Rechner ermittelt dann einen entsprechenden y - oder x -Wert auf der Regressionsgeraden. Grundlage für dieses Verfahren ist die Geradengleichung $y = mx + b$, wobei m (die Steigung) und b (der Schnittpunkt mit der y -Achse) aufgrund der zuvor eingegebenen Daten bestimmt werden.



Beispiel: Eine Versicherungsgesellschaft stellt fest, daß die Menge der Vertragsabschlüsse entsprechend der Anzahl der Mitarbeiter variiert.

Anz. d. Mitarbeiter im Verkauf	7	12	3	5	11	8
Verkaufsziffern in 1000/Monat	99	152	81	98	151	112

Wieviele Mitarbeiter müssen bei dieser Gesellschaft im Verkauf tätig sein, damit der Umsatz auf 200 000 DM pro Monat steigt. Welcher monatliche Umsatz wäre bei 15 Mitarbeitern zu erwarten?

Taste	Anzeige	Bemerkungen
2nd prog 1 SBR CLR	0	Einleiten der Berechnung
7 x=t	0.	Erster x-Wert
99 2nd Σ+	1.	Datenpunkt 1
12 x=t	8.	Zweites x
152 2nd Σ+	2.	Datenpunkt 2
3 x=t	13.	etc.
81 2nd Σ+	3.	
5 x=t	4.	
98 2nd Σ+	4.	
22 x=t	6.	falsche Eingabe
151 2nd Σ+	5.	
22 x=t	23.	} Löschen der falschen Eingabe
151 INV 2nd Σ+	4.	
11 x=t	21.	
151 2nd Σ+	5.	
8 x=t	12.	
112 2nd Σ+	6.	
200 2nd Op 15	17.81578947	benötigtes Verkaufspersonal für 200 000 DM Umsatz
15 2nd Op 14	176.5561798	Umsätze bei 15 Mitarbeitern
2nd Op 12	51.66853933	Schnittpunkt d. Geraden mit der y-Achse
x=t	8.325842697	Steigung der Geraden

Steigung und Schnittpunkt mit der y-Achse wurden so berechnet, daß die Gerade auf Wunsch aufgezeichnet werden kann. Die Steigung entspricht der Zunahme der Verkaufsziffern pro Person. Der Schnittpunkt mit der y-Achse ist der unabhängige Umsatz.

Von besonderer Bedeutung ist noch, daß bei der Anwendung einer der mathematischen Funktionen auf ein Element oder auf beide Elemente des Zufallsvariablenpaares andere Regressionstypen möglich sind.

Beispiel: Wenn man den Logarithmus einer der Variablen ermittelt, ehe sie als Datenpunkt eingegeben wird, erhält man eine semi-logarithmische Kurvenanpassung. Diese Variationen können durch Anwendung der natürlichen Logarithmen, der Exponentialfunktionen, der Potenzen und Wurzeln und des Reziprokwerts erreicht werden.



Zu Beginn Ihrer Datenanalyse müssen Sie den Kurventypus wählen, der für Ihre besondere Aufgabe charakteristisch ist. Man kann auch mehrere Kurventypen versuchen, um zu sehen, welche Kurve dem Bedarf am besten entspricht.

Beispiel: Eine Stadt gibt folgende Bevölkerungsdaten bekannt. Welcher Bevölkerungsstand läßt sich für das Jahr 1980 vorher-sagen, und in welchem Jahr wird die Bevölkerung voraussichtlich auf 50 000 Einwohner angewachsen sein?

Jahr	1930	1940	1950	1960	1970
Bevölkerungsstand	3221	5361	9212	15410	27612

Es ist typisch für Bevölkerungsdaten, daß sie eine Exponentialkurve der Form $y = ae^{bx}$ beschreiben. Wenn man den Logarithmus von beiden Seiten dieser Gleichung berechnet, erhält man $\ln y = \ln a + bx$. Deshalb kann man bei Aufzeichnung von x gegen $\ln y$ (semilogarithmisch) eine Gerade eintragen.

Taste	Anzeige
2nd Pgm 1 SBR CLR	0.
1930 x↵	0.
3221 ln x 2nd Σ+	1.
1940 x↵	1931.
5361 ln x 2nd Σ+	2.
1950 x↵	1941.
9212 ln x 2nd Σ+	3.
1960 x↵	1951.
15410 ln x 2nd Σ+	4.
1970 x↵	1961.
27612 ln x 2nd Σ+	5.
1980 2nd Op 14 INV ln x	46081.80979
50000 ln x 2nd Op 15	1981.524472

1980 wird die Bevölkerung auf etwa 46 080 Einwohner angewachsen sein, und 1981 die 50 000-Einwohner -Marke erreichen.

Trendlinien – Analyse

Bei Daten, die in periodischen Intervallen gesammelt wurden, wie in jährlichen oder täglichen Abständen oder bei Daten pro Ereignis, kann der Rechner den x -Wert pro Datenpunkteingabe automatisch um 1 erhöhen. Zu Beginn ordnet der Rechner den jeweiligen Inhalt des T-Registers als x -Wert für den ersten Datenpunkt zu, und addiert dann 1 für den zweiten, 1 für den dritten etc. Alle Datenpunkte werden über 2nd Σ+ eingegeben. Der Anfangswert für x kann durch Eingabe des ersten x -Wertes auf eine beliebige Zahl eingestellt werden, von der aus der Rechner dann um jeweils 1 erhöht:

x_1 x↵ · y_1 2nd Σ+ · y_2 2nd Σ+ · y_3 2nd Σ+ etc.

Um eine ungewollte Dateneingabe zu löschen, drückt man x↵ – 1 x↵ und gibt dann den unerwünschten y -Wert ein.



Beispiel: Eine Computerfirma verzeichnet folgende Jahresgewinne:

Jahr	1962	1963	1964	1965-1970	1971	1972	1973	1974
Gewinn in Millionen	-2.1	-0.3	0.8	stillgelegt	2.9	2.8	3.6	4.0

Welcher Gewinn ist für 1980 zu erwarten und wann wird voraussichtlich die 10-Millionen-Marke überschritten?

Taste	Anzeige	Bemerkungen
2nd Pgm 1 SBR CLR	0.	Einleiten d. Berechnung
1962 x^{±t}	0.	Einstellung von x
2.1 +/- 2nd Σ+	1.	1962 Verlust
.3 +/- 2nd Σ+	2.	1963 Verlust
.8 2nd Σ+	3.	1964 Gewinn
1971 x^{±t}	1965.	Neueinstellung von x
2.9 2nd Σ+	4.	1971 Gewinn
2.8 2nd Σ+	5.	1972 Gewinn
3.6 2nd Σ+	6.	1973 Gewinn
4 2nd Σ+	7.	1974 Gewinn
1980 2nd Op 14	6.52181966	
10 2nd Op 15	1988.297788	

Für 1980 kann die Firma 6.5 Millionen Gewinn erwarten. Die 10-Millionen-Marke dürfte 1988 erreicht werden.

Beliebige statistische Funktionen können beliebig für die statistischen Berechnungen verwendet werden. Für das letzte Beispiel hätte man ebenso Mittelwert, Standardabweichung, Varianz, Steigung und Schnittpunkt berechnen können.

Statistik in Berechnungen

Statistische Operationen können im Verlauf komplexer Berechnungen durchgeführt werden. Hierbei dürfen bis zu 4 Operationen unvollständig sein und man kann immer noch die statistischen Berechnungen einschleiben. (In der Statistik werden intern bis zu 4 Verarbeitungsebenen benötigt.)

Beispiel: Eine Versicherungsgesellschaft errechnet ihre Betriebskosten vielleicht mit der Formel $3 + 2 \times 1.2^{(4+N)}$, wobei N die Anzahl der Mitarbeiter ist, die für einen Umsatz von 200 000 benötigt werden. Tasten Sie einfach $3 + 2 \times 1.2^{(4+N)}$ ein, und führen Sie dann eine lineare Regression wie in einem der letzten Beispiele durch. Nachdem die Anzahl der Mitarbeiter für den angegebenen Umsatz vorhergesagt wurde, schließen Sie die Berechnung mit **=** ab. Beachten Sie, daß während der statistischen Berechnung vier Operationen unvollständig sind.

Als Teile in einem Programm beanspruchen statistische Berechnungen eine Unterprogrammebene. Weitere Informationen über Unterprogramme folgen.



PROGRAMMIEREN – ALLGEMEIN

Programmierung Ihres Rechners

Bei der Lösung eines Problems manuell über die Tastatur bestimmen Sie die für die Berechnung nötige Folge von Operationen und Funktionen, und tasten dann Ihre Lösung ein. Programmieren bedeutet nicht viel mehr als die Eingabe der Betriebsart Learn (Learn-Modus) und die Anweisung an den Rechner, sich die resultierende Tastenfolge zu merken. Tatsächlich werden die einzelnen Tastenbefehle in den Plätzen des Programmspeichers gespeichert, und jeder Tastendruck wird auf diese Weise ein Programmbefehl. Die Reihe der Tastenbefehle ist jetzt zu einem Programm geworden. Wenn die Befehle des Programms ausgeführt werden, erzeugen sie dasselbe Ergebnis wie bei entsprechender manueller Bedienung des Rechners. Ist das Programm einmal gespeichert, kann es immer wieder mit immer neuen Variablen durchgeführt werden, und dabei erübrigt sich die Eingabe aller Programmbefehle. Damit spart man nicht nur Zeit bei der Eingabe, sondern verringert auch das Risiko für Eingabefehler, die dann später das Lösungsverfahren beeinträchtigen.

Das Programm bleibt im Programmspeicher erhalten, bis es durch ein anderes ersetzt bzw. mit den Tasten **[2nd]** **[CP]** gelöscht, oder bis der Rechner ausgeschaltet wird (nur TI-58/59). In der Zwischenzeit können Sie das Programm so oft wie nötig verwenden. Sie stellen zum Beispiel fest, daß Sie eine Antwort aus einem gespeicherten Programm brauchen, während Sie eine Reihe manueller Operationen durchführen; Sie können das Programm dann einfach aufrufen und ablaufen lassen, und mit den Programmresultaten Ihre manuellen Berechnungen wieder aufnehmen.

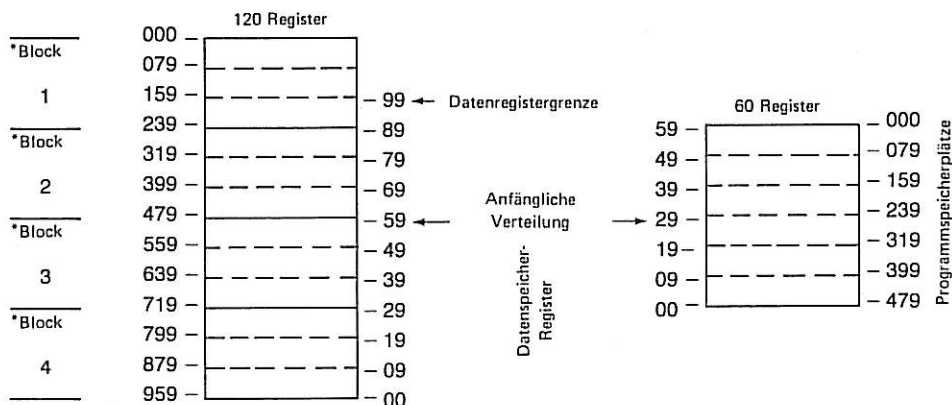
Durch die Constant Memory - Eigenschaft des TI-58C bleibt der Programmspeicher-Inhalt erhalten, auch wenn der Rechner ausgeschaltet wird, oder wenn das Batteriepaket herausgenommen oder kurze Zeit entladen wurde (in jedem Fall muß der Rechner ausgeschaltet sein). Auch der Datenspeicherinhalt, die Speicherbereichsverteilung und die Festkommaseinstellung bleiben durch die Constant Memory - Eigenschaft des TI-58C erhalten. Wenn Sie ein laufendes Programm durch Abschalten des TI-58C unterbrechen, können eine unvollständige Berechnung, der Flag-Status, der Unterprogramm-Status, der t-Register-Inhalt oder der Winkel-Status verloren gehen. Unterbrechen Sie also grundsätzlich ein laufendes Programm, ehe Sie den TI-58C ausschalten.

Das Programmiersystem des Rechners genügt auch höchsten Ansprüchen – und ist doch einfach in der Anwendung. Bei gründlichen Kenntnissen der Rechnerstruktur sind Sie mit einem Problemlösungsgerät ausgestattet, das mit seinen Möglichkeiten dem Minicomputer sehr nahekommt.



Speicherkapazität und Verteilung

Der gesamte Speicherbereich innerhalb des Rechners dient der Daten- und Programmspeicherung.



Programmierbarer Rechner
TI-59

Programmierbarer Rechner
TI-58/TI-58C (Constant Memory)

SPEICHERBEREICH

Für den Rest dieses Abschnitts gilt : Angaben über den TI-58/TI-58C sind in Klammern gesetzt. Sie stehen unmittelbar nach den TI-59 - Informationen.

Im gesamten Speicherbereich stehen 120 (60) Register für die Speicherung zur Verfügung. Anfänglich besteht eine gleichmäßige Verteilung zwischen Programm- und Datenspeicher mit je 60 (30) Registern. In jedem Register des Programmspeichers können 8 Programmbefehle gespeichert werden, also $8 \times 60 = 480$ ($8 \times 30 = 240$) Programmschritte, wenn 60 (30) Register für die Datenspeicherung bleiben.

Dieser Speicherbereich kann in einem anderen Verhältnis in Gruppen von je 10 Registern aufgeteilt werden. Man kann zum Beispiel alle 120 (60) Register für ein Programm mit $120 \times 8 = 960$ ($60 \times 8 = 480$) Programmspeicherplätzen verwenden und auf Datenregister verzichten. Oder man wählt die Verteilung so, daß 100(60) Datenregister verfügbar sind und $20 \times 8 = 160$ (0) Programmspeicherplätze. Weitere Kombinationen sind möglich mit einer Einschränkung: Man kann maximal 100(60) Datenregister verwenden, da jede Speicheroperation eine zweistellige Adresse erfordert. Also können nur die Register 00 bis 99 (00-59) belegt werden.

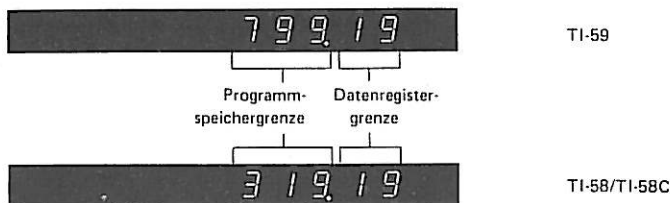
Alle 60 Register des TI-58C sind in die Constant Memory - Eigenschaft einbezogen, und ihr Inhalt geht durch Abschalten des Geräts nicht verloren.

*Block = gemeinsamer Speicherbereich für Daten und Programm



Um den Speicherbereich zu verteilen, geben Sie die Anzahl der benötigten Zehnergruppen der Datenregister ein, 0 bis 10 (0 – 6), und drücken dann die Tasten **[2nd]** **[Op]** 17, für 20 Datenregister also 2 **[2nd]** **[Op]** 17.

Bevor die Speicherbereichsverteilung geändert wird, lösche alle Festkomma und Exponential- (**[Eng]** und **[EE]**) Einstellungen. In der Anzeige wird diese Verteilung wie folgt dargestellt.



Damit wird angezeigt, daß für die Datenspeicherung 20 Register, 00 bis 19, zur Verfügung stehen, und daß der Programmspeicherbereich 800 (320) Speicherplätze von den Adressen 000 bis 799 (000 bis 319) umfaßt.

Mit der Tastenfolge **[2nd]** **[Op]** 16 kann die derzeitige Verteilung jederzeit kontrolliert und in der oben dargestellten Form angezeigt werden.

Grundfunktionen für die Programmsteuerung

Mit Kenntnissen über mehrere wichtige Steuerfunktionen können Sie mit der Programmierung Ihres Rechners beginnen.

[LRN] – LEARN – Wenn diese Taste einmal gedrückt wird, schaltet sich der Rechner in die Betriebsart Learn (Learn-Modus). Dann können Sie mit der Belegung des Programmspeichers beginnen und die Anweisungen später ablaufen lassen. Drückt man **[LRN]** erneut, wird die Steuerung wieder an die Tastatur gegeben und die ursprüngliche Anzeigeform wird wiederhergestellt. Bei Anwendung der Learn-Taste bleibt der Wert in der Anzeige immer erhalten, aber ein eventuelles Blinken der Anzeige wird gestoppt. Der Learn-Modus kann nicht eingegeben werden, wenn im Programmspeicher ein geschütztes Programm ist.

[2nd] **[CP]** (clear program) – PROGRAMMLÖSCHUNG – Wenn diese Anweisung manuell über die Tastatur gegeben wird, löscht sie alle Plätze des Programmspeichers, das Unterprogrammrückprung-Register sowie das T-Register; außerdem werden alle Flags rückgesetzt und der Programmzeiger auf 000 eingestellt. Auch der Datenschutzwert wird aufgehoben (nur TI-59). Als Teil in einem Programm stellt diese Anweisung nur das T-Register auf Null.

[R/S] – RUN/STOP – Umkehrung des derzeitigen Verarbeitungsstatus. Drückt man **[R/S]**, und das Programm läuft, wird es unterbrochen. Wenn das Programm gerade unterbrochen ist, wird die Verarbeitung mit der Anweisung **[R/S]** an der augenblicklichen Position des Programmzeigers wieder aufgenommen. Damit können vor Ende des Ablaufs noch Daten eingegeben oder Zwischenergebnisse geprüft werden.



RST (reset) – RÜCKSTELLEN – Anweisung, den Programmzeiger auf den Speicherplatz 000 des Programmspeichers rückzustellen (dies gilt auch für die Software-Programme). **RST** löscht das Unterprogrammrücksprung-Register und alle Programmflags werden rückgesetzt. Darüberhinaus wird die Anweisung zur Unterbrechung eines Software-Programms verwendet, wenn das Programm sich nicht selbst abstellen kann. In diesem Fall gehen alle Ergebnisse bis auf die im Datenregister gespeicherten verloren.

2nd PAUSE – PAUSE – Während des Programmablaufs bewirkt diese Taste, daß der derzeitige Wert des Anzeigeregisters für kurze Zeit ausgewiesen wird. Pausenbefehle können immer bei Bedarf gegeben werden, auch fortlaufend, damit die Anzeige länger sichtbar bleibt. Bleibt die Taste während der Durchführung eines Programms gedrückt, wird das Ergebnis jedes Programmschritts angezeigt. Beim Ablauf eines Software-Programms oder eines geschützten Programms ist die Taste unwirksam.

Learn-Modus

Wenn eine Rechenfolge einmal festliegt, schaltet man mit den Tasten **2nd CP LRN** in den Learn-Modus und tastet dann die Folge in den Programmspeicher ein. Mit **2nd CP** wird sichergestellt, daß das Programm, beginnend mit Speicherplatz 000, eingegeben wird. Im Learn-Modus hat die Anzeige folgendes Format.



Die Programmspeicherplätze beginnen mit 000 und werden bis zu Ihrer gewählten Verteilung fortlaufend numeriert. Anfänglich ist der Inhalt des Programmspeichers in allen Plätzen Null, und die Nullbelegung bleibt in allen Speicherplätzen erhalten, die nicht mit Absicht geändert werden.

Im allgemeinen enthält jeder Speicherplatz eine Operation, eine Adresse, oder nur eine einzelne Ziffer.

Der Befehlskode (oder Tastenkode) ist eine zweistellige Zahl, die jeder Operation entsprechend ihrer tatsächlichen Lage auf der Tastatur zugeordnet ist. Der Rechner zeigt normalerweise 00 als Tastenkode an, wenn ein Programm eingegeben wird. Grund: Der Rechner geht automatisch zum nächsten verfügbaren Speicherplatz über, sobald ein Befehl vollständig eingegeben ist. Ausführliche Informationen über die Befehlskodes finden Sie in dem Kapitel über Programmredigierung.

Die Funktion des Programmzeigers (Befehlsadressregisters) ist es, eine genaue Kontrolle zu ermöglichen, welche Adresse im Programmspeicher gerade erreicht ist. Im Learn-Modus bewegt sich dieser Zeiger durch den Programmspeicher, wobei immer der nächste freie Speicherplatz ausgewiesen wird.

Nach Eintasten eines Programms in den Programmspeicher drückt man **LRN** erneut, um den Rechner wieder auf Tastatursteuerung zu schalten, dann werden die Variablen eingegeben und der Programmablauf gestartet.



Eingabe Ihres Programms

Folge zum Eintasten Ihres Programms:

1. Sie geben manuell über die Tastatur **RST** oder **2nd CP** ein. In beiden Fällen wird das Befehlsadressregister auf 000, den ersten Speicherplatz des Programmspeichers, eingestellt. Die Folge **2nd CP** löscht auch den Programmspeicher.
2. Mit der Taste **LRN** schalten Sie in den Learn-Modus. Die charakteristische 5-stellige Anzeigeform kennzeichnet diesen Modus.
3. Das Programm wird vollständig in Einzelschritten einschließlich aller erforderlichen **2nd** - und **INV** -Präfixe eingetastet. Die Anzeige weist immer den nächsten Speicherplatz aus, der für einen Befehl frei ist, und nicht den eben belegten.
4. Achten Sie darauf, daß das Programm die Programmspeicherkapazität nicht überschreitet. Wenn der letzte Speicherplatz belegt ist, schaltet sich der Rechner automatisch wieder in den Rechenmodus, wo das charakteristische Anzeigeformat natürlich fehlt.
5. Drücken Sie die Taste **LRN** erneut, um den Learn-Modus wieder aufzuheben.
6. Lassen Sie eine Testaufgabe mit bekannten Ergebnissen ablaufen und überprüfen Sie damit das Programm auf seine Richtigkeit. Wenn nötig, redigieren Sie das Programm.

Folgendes Beispiel soll die obigen Punkte veranschaulichen. Beispiel: Erstellen Sie ein Programm, um das Volumen eines geraden Kreiszylinders mit dem Radius r und der Höhe h zu berechnen.

Lösungsgleichung: $V = \pi r^2 h$

Gewünschte Programmoperation: Eingabe von r
 Programmstart
 Halt für die Eingabe von h
 Berechnung des Volumens, Halt und Anzeige des Resultats



Tastenfolge	Anzeige	Bemerkungen
2nd CP	0	Einstellung des Programmzeigers auf Speicherplatz 000 und Löschen des Programmspeichers
LRN	000 00	Der Rechner wird in den Learn-Modus geschaltet und Beginn der Programmeingabe
π^2	001 00	r^2 (r wird vor Ausführung des Programms eingegeben)
X	002 00	
2nd π	003 00	Nur 1 Speicherplatz wird belegt
X	004 00	πr^2 im Anzeigeregister
R/S	005 00	Halt für die Eingabe von h
=	006 00	Berechnung des Resultats
R/S	007 00	Programmstop, V wird angezeigt
RST	008 00	Rückkehr zu dem Befehl in 000
LRN	0	Übergang auf Tastatursteuerung

Die letzten beiden eingegebenen Tastenbefehle haben besondere Funktionen. Die Taste **R/S** unterbricht den Ablauf und zeigt das Endergebnis an. Mit **RST** erfolgt automatisch die Rückkehr zum Speicherplatz 000, wenn r für eine neue Variablengruppe eingegeben und **R/S** gedrückt wird.

Die Taste **R/S** kehrt den augenblicklichen Verarbeitungsstatus des Programms um. Wenn man diese Taste drückt oder wenn die Anweisung als Programmbefehl auftritt, während ein Programm gerade abläuft, wird die Verarbeitung unterbrochen und sofort auf Tastatursteuerung geschaltet. Gibt man dann die Anweisung **R/S** manuell, beginnt der Programmablauf erneut, und die Berechnungen werden von der Stelle an wieder aufgenommen, wo die Verarbeitung unterbrochen wurde. Der Programmzeiger bleibt dort, wo er zum Zeitpunkt der Programmunterbrechung war. Die Verarbeitung kann also ohne negativen Einfluß auf die Berechnungen wieder anlaufen, vorausgesetzt, Sie nehmen keine Änderungen vor.

Ablauf Ihres Programms

Wenn ein Programm abläuft, werden die Befehle, beginnend mit der augenblicklichen Adresse des Programmzeigers, der Reihe nach durchgeführt. (Vorteilhafte Ausnahmen werden später noch erläutert.) Um diese Verarbeitung einzuleiten, drücken Sie einfach die Taste **R/S** (run/stop). Die Position des Programmzeigers deckt sich immer mit dem Verarbeitungsstand des Programms. Wenn der Rechner den Ablauf über die Programmgrenze hinaus fortsetzen will, wird die Verarbeitung unterbrochen und die Anzeige blinkt. Aus diesem Grund sollten Programme grundsätzlich mit **R/S** oder mit **INV** **SBR** oder mit einem Sprungbefehl, Operationen, die später ausführlich behandelt werden, enden.

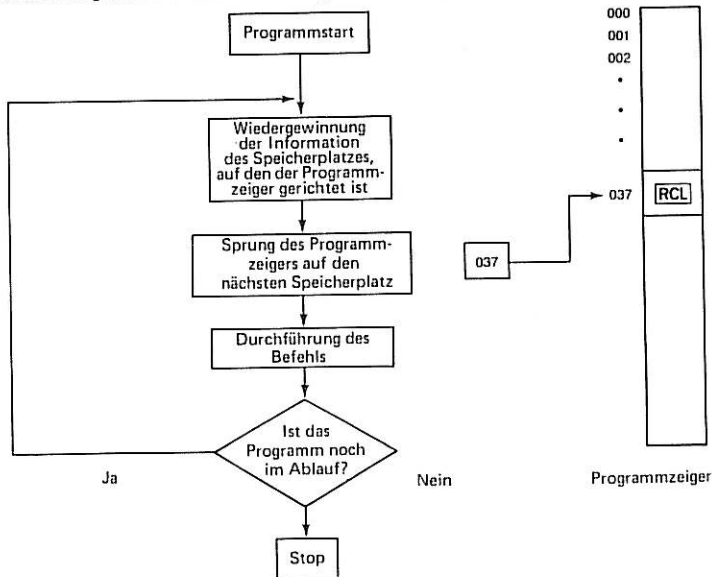
Mit dem Volumenproblem im Programmspeicher soll nun das Programm ablaufen.
Berechnen Sie das Volumen für $r = 3$ und $h = 9$



Taste	Anzeige	Bemerkungen
RST	0	Einstellung des Programmzeigers auf 000
3	3	Eingabe von r
R/S	(dunkel)	Beginn des Programmablaufs
	28.27433388	π^2 – der Wert im Anzeigeregister, als die R/S – Anweisung gegeben und das Programm damit unterbrochen wurde
9	9	Eingabe von h
R/S	(dunkel)	Wiederaufnahme der Verarbeitung
	254.4690049	Programmstop, V wird angezeigt

Beachten Sie, daß die Anzeige dunkel bleibt (mit Ausnahme eines schwach leuchtenden „c“ im äußersten linken Teil der Anzeige), während das Programm abläuft. Der Zeitraum, in dem die Anzeige dunkel ist, kann stark variieren, je nachdem, welches Programm gerade abläuft, und an welche Stromquelle das Gerät angeschlossen ist.

Läuft eine Befehlsfolge ab, wird der Verarbeitungsablauf ständig durch den Programmzeiger kontrolliert, der sich auf jeden Befehl richtet, wenn er ausgeführt werden muß. Siehe folgende Abbildung:



Da noch weitere Programmiermöglichkeiten behandelt werden, wird auch die Rolle des Programmanzeigers noch erweitert.



Arbeiten mit Programmen

[SST] (single-step) — EINZELSCHRITT — Der Programmzeiger wird um einen Schritt weiterbewegt. Im Learn-Modus bewirkt diese Taste, daß der nächste Speicherplatz angezeigt wird. Drückt man **[SST]** manuell über die Tastatur, wird das Programm in Einzelschritten durchgeführt, und jeder Schritt angezeigt.

[BST] (back-step) — EINZELSCHRITT ZURÜCK — Im Learn-Modus bewirkt diese Taste, daß sich der Programmzeiger um einen Schritt zurückbewegt, und daß der dort gespeicherte Befehlscode angezeigt wird.

Diese Tasten sind unwirksam, wenn Software-Programme ablaufen.

Zum Eintasten eines Programms oder zur Kontrolle, wenn es bereits in den Programmspeicher eingebracht ist, gibt es zwei Befehle, die das schrittweise Vorgehen innerhalb eines Programms von einer Anweisung zur anderen ermöglichen. Drückt man die Einzelschritt-Taste **[SST]** im Learn-Modus, rückt der Programmzeiger um eins weiter, und die Nummer des nächsten Speicherplatzes sowie der Befehlscode werden ohne Einfluß auf die gespeicherten Befehle ausgewiesen. Wiederholte **[SST]** — Eingaben ermöglichen also, den Programmspeicher in Einzelschritten zu durchlaufen, und die Codes für die hier gespeicherten Befehle zu überprüfen.

Der Einzelschritt-Befehl kann manuell über die Tastatur und im Learn-Modus eingegeben werden. Bei Tastatursteuerung bewirkt **[SST]** die tatsächliche Durchführung eines gespeicherten Programms in Einzelschritten. Jedesmal, wenn **[SST]** gedrückt wird, wird der Befehl in der derzeitigen Position des Programmzeigers ausgeführt, und der Programmzeiger rückt weiter wie beim Programmablauf. In der Anzeige wird der aus jenem Befehl errechnete Wert ausgewiesen. Manchmal sind mehrere Einzelschrittbefehle erforderlich, ehe sich eine Reaktion zeigt, da die laufende Operation oft aus mehreren Schritten besteht. Bei der Folge + RCL 09 wären zum Beispiel drei Einzelschrittbefehle nötig, ehe der Inhalt des Speicherregisters 09 aufgerufen wird.

Wird ein Programm in Einzelschritten vor oder zurück im Learn-Modus durchlaufen, erscheint der Tastenkode jedes Speicherplatzes, der die verschiedenen Befehle angibt.

Befehlskodes (Tastenkodes)

Im Learn-Modus werden die derzeitige Position des Programmzeigers sowie der Befehl des betreffenden Speicherplatzes angezeigt. Der Befehl wird durch einen zweistelligen Kode dargestellt, der sich direkt aus der Platzierung der Taste auf der Tastatur herleitet. Die erste Stelle gibt eine der 9 Zeilen an (1 bis 9 von oben nach unten), in der sich die Taste befindet. Aus der zweiten Stelle entnimmt man die Spalte, in der die Taste eingeordnet ist (von links nach rechts mit 1 bis 5 durchnummeriert). Die Taste **[STO]** befindet sich zum Beispiel in Reihe 4, Spalte 2; der Befehlskode ist also 42. Bei Zweitfunktionen wird zu einer bestimmten Zeile-Spalte-Adresse 5 addiert, d. h., **[2nd]** **[sin]** ist Kode 38, weil die Funktion über **[x²]**, Kode 33, angegeben ist. Für die äußerste rechte Tastenspalte wird bei Zweitfunktionen 5 addiert, aber ändern Sie keinesfalls die Nummer der Reihe, d. h., **[2nd]** **[E]** ist Kode 15 + 5 = 10, nicht 20. Den Zifferntasten 0 bis 9 sind die Kodes 00 bis 09 zugeordnet.



Tastatur und Tastenkodes

Taste	Tasten- kode	Taste	Tasten- kode	Taste	Tasten- kode	Taste	Tasten- kode
A 16		B 17		C 18		D 19	
A 11		B 12		C 13		D 14	
		INV 27		log 28		CP 29	
2nd Kombi-Befehl		INV 22		lnx 23		CE 24	
Pgm 36*		P→R 37		sin 38		cos 39	
LRN ohne Kode		x↔t 32		x² 33		√x 34	
Ins ohne Kode		CMs 47		etc 48*		Prd 49*	
SST ohne Kode		STO 42*		RCL 43*		SUM 44*	
Del ohne Kode		Eng 57		fix 58*		Int 59	
BST ohne Kode		EE 52		(53) 54	
Pause 66		x↔t 67*		Mem 68		Op 69*	
GTO 61*		7 07		8 08		9 09	
Lbl 76*		x↔t 77*		Σ+ 78		Σ 79	
SBR 71*		4 04		5 05		6 06	
St Hg 86*		ll Hg 87*		D/MS 88		π 89	
RST 81		1 01		2 02		3 03	
Write 96		Dsz 97*		Adv 98		Prt 99	
R/S 91		0 00		• 93		+/- 94	
						Ind 40 oder Kombi-Befehl	
						yx 45	
						 x 50	
						÷ 55	
						Deg 60	
						X 65	
						Rad 70	
						- 75	
						Grad 80	
						+ 85	
						List 90	
						= 95	

*Befehle, die weitere Befehle oder Adressen erfordern, um vollständig zu sein.

Anmerkung: Der Befehl **Ind** ist manchmal numerisch mit dem Kode der Taste kombiniert, mit der zusammen er verwendet wird.

Eine Klarsichtfolie als Auflage für den Rechner mit dem Tastenkodewird mitgeliefert. Wenn sie über die Tasten gelegt wird, zeigt sie den Kode, der jeder Taste zugeordnet ist.



Tastenkodes in ihrer numerischen Reihenfolge

Tasten- kode	Taste	Tasten- kode	Taste	Tasten- kode	Taste
00	0	39	2nd COS	72	STO 2nd Ind
09	9	40	2nd Ind	73	RCL 2nd Ind
10	2nd E	42	STO	74	SUM 2nd Ind
11	A	43	RCL	75	-
12	B	44	SUM	76	2nd (b)
13	C	45	y ^x	77	2nd x [≠]
14	D	47	2nd CMs	78	2nd Σ+
15	E	48	2nd Exc	79	2nd x̄
16	2nd A'	49	2nd Prd	80	2nd Grad
17	2nd B'	50	2nd 1/x	81	RST
18	2nd C'	52	EE	83	GTO 2nd Ind
19	2nd D'	53	(84	2nd Op 2nd Ind
20	2nd CLR	54)	85	+
22	INV	55	÷	86	2nd SI Up
23	In x	57	2nd Eng	87	2nd II Up
24	CE	58	2nd Fix	88	2nd D.M.S
25	CLR	59	2nd Int	89	2nd π
27	2nd INV	60	2nd Deg	90	2nd List
28	2nd log	61	GTO	91	R/S
29	2nd CP	62	2nd Pgm 2nd Ind	92	INV SBR
30	2nd tan	63	2nd Exc 2nd Ind	93	.
32	x [≠]	64	2nd Prd 2nd Ind	94	+/-
33	x ²	65	X	95	=
34	√x	66	2nd Pause	96	2nd Write
35	1/x	67	2nd x [≠]	97	2nd Dsz
36	2nd Pgm	68	2nd Nop	98	2nd Adv
37	2nd P→R	69	2nd Op	99	2nd Prt
38	2nd sin	70	2nd Rad		
		71	SBR		

Bei normaler Ausnutzung des Rechners werden Sie mit den häufiger auftretenden Befehlskodes bald vertraut, so daß die dauernde Bezugnahme auf diese Tabellen nicht mehr erforderlich ist. Die meisten Codes können über die Rechnertastatur oder über die Kodefolie schnell bestimmt werden.



Speicherung der Tastenanweisungen

Die meisten Befehle belegen einen Platz im Programmspeicher. Einige Befehlsfolgen lassen jedoch mehrere Tastenbetätigungen für einen einzelnen Programmspeicherplatz zu. Die Taste **2nd** ist mit dem Folgebefehl gekoppelt, und belegt einen Speicherplatz. Die zwei-stelligen Adressen nach den Befehlen für Datenspeicherung, für den Zugriff auf Software-Programme und für die speziellen Steueroperationen bilden ebenfalls eine Einheit und benötigen nur einen Speicherplatz. **RCL** 16 beansprucht zum Beispiel nur zwei Speicherplätze, ebenso wie **2nd** **Rgm** 12 mit **2nd** **Rgm** in einem Speicherplatz und der Nummer 12 im nächsten. Diesen Vorgang führt der Rechner automatisch aus.

Unbedingte Sprungadressen (Siehe Seite V-64) wie **GTO** 123 werden wie folgt gespeichert: ein Speicherplatz für **GTO**, der folgende für 01 und der nächste für 23. Beim Eintasten dieser Folge nimmt der Rechner die Tastenbefehle auf und platziert sie richtig in den Speicherplätzen, ohne daß Sie besondere Mühe darauf verwenden müssen.

Manchmal, wenn die Taste **Ind** zusammen mit einem anderen Befehl verwendet wird, belegt **Ind** in der Kombination mit diesem Befehl nur einen Speicherplatz, und erhält einen anderen Befehlscode. Die neuen Codes basieren nicht auf der Position des **Ind**-Befehls auf der Tastatur, sondern sind zugewiesene Tastenkodes, die sich aus der Lage der Ziffern auf der Tastatur herleiten. **STO** **2nd** **Ind** hat zum Beispiel den Code 72, der einen Speicherplatz belegt. Diese speziellen Zuordnungen verarbeitet der Rechner intern, und sie sind in der numerischen Reihenfolge der Tastenkodes auf der vorigen Seite aufgelistet. Folgende indirekte Befehle sind davon betroffen:

Tastenfolge	Tastenkodes
2nd Rgm 2nd Ind	62
2nd Ex 2nd Ind	63
2nd Prd 2nd Ind	64
STO 2nd Ind	72
RCL 2nd Ind	73
SUM 2nd Ind	74
GTO 2nd Ind	83
2nd Op 2nd Ind	84

Die indirekten Befehle, die nicht durch diese Kombinationen beeinflusst werden, speichern den Befehlscode von **2nd** **Ind** entsprechend der Lage der IND-Anweisung auf der Tastatur, also 40 nach dem Code des Befehls, mit dem **Ind** verwendet wird. **2nd** **Σ** **2nd** **Ind** wird zum Beispiel wie folgt gespeichert: **2nd** **Σ**, Code 67 in einem Speicherplatz, und **2nd** **Ind**, Code 40, im nächsten. Die Anwendungsmöglichkeiten für indirekte Folgen werden später noch erklärt. Die Tastenfolge **INV** **SBR** ist ein zusammengezogener Befehl, der nur eine Programmzeile mit dem Tastenkode 92 belegt.

REDIGIEREN VON PROGRAMMEN

2nd **Nop** NULL-OPERATION — Im Learn-Modus kann man mit dieser Taste einen ungewollten Befehl löschen, oder zwischen Programmteilen Intervalle für spätere Ergänzungen freihalten. Beim Programmablauf findet an diesen Stellen ganz einfach keine Operation statt. Die Anwendung dieser Taste beeinflusst in keinem Fall eine Tasten- oder Befehlsfolge, noch eine Dateneingabe, es sei denn, man verwendet die Taste als Label.

2nd **Del** — DELETE (LÖSCHEN EINES EINZELBEFEHLS) — Im Learn-Modus wird der angezeigte Befehl gelöscht, und jeder folgende Befehl rückt um einen Speicherplatz auf, so daß die Lücke wieder geschlossen wird. Der Programmzeiger wird bei dieser Operation nicht bewegt.



2nd **Ins** - INSERT (EINFÜGEN EINES BEFEHLS) – Im Learn-Modus werden, beginnend mit dem angezeigten Speicherplatz, alle Befehle in Richtung auf das Ende des Programmspeichers um eine Stelle verschoben. Auf diese Weise wird Raum für das Einfügen eines neuen Befehls geschaffen.

Für die Bearbeitung eines Programms im Programmspeicher haben Sie folgende Möglichkeiten:

1. Durchgehen des Programms in Einzelschritten vor und zurück, wobei die Befehle in den einzelnen Speicherplätzen angezeigt werden.
2. Ersetzen eines Befehls durch einen anderen;
3. Löschen eines ausgewiesenen Befehls und Schließen der Lücke;
4. Raum für das Einfügen eines neuen Befehls schaffen.

Mit diesen Anweisungen können im Programm Korrekturen oder Änderungen mit minimalem Aufwand vorgenommen werden.

Ersetzen eines Befehls durch einen anderen

Im Learn-Modus überschreibt eine Tasteneingabe an beliebiger Stelle in einem Programm den Befehl, der zuvor in der betreffenden Adresse gespeichert war. Wenn Sie das Programm in Einzelschritten vor oder zurück durchprüfen, und Sie bemerken einen ungewollten Befehl, drücken Sie zur Korrektur einfach die richtige Taste. Der angezeigte falsche Befehl wird sofort durch den korrigierten ersetzt.

Löschen eines Befehls

Ein Einzelbefehl kann mit den Tasten **2nd** **Del** aus dem Programm genommen werden. Im Learn-Modus kann man einen angezeigten Befehl löschen, und alle folgenden Befehle rücken um einen Speicherplatz auf, so daß die Lücke im Programm wieder geschlossen wird. Die Folge ist, daß der letzte Speicherplatz des Programmspeichers frei wird und dann mit einer Null belegt ist. Ein beliebig großer Teil des Programms kann auf diese Weise gelöscht werden, wenn es erforderlich ist.

Einfügen eines Befehls

Wenn Sie in einem Programm noch einen Befehl einfügen müssen, drücken Sie die Tasten **2nd** **Ins**. Alle Befehle, beginnend mit dem gerade angezeigten, werden um einen Speicherplatz verschoben, um Raum für einen neuen Befehl zu schaffen. Der neue Befehl muß jetzt nur noch eingetastet werden. Diese Operation läßt sich beliebig wiederholen, um zu einem Programm auch größere Abschnitte hinzuzufügen. Beachten Sie aber, daß jedesmal, wenn diese Operation durchgeführt wird, der Befehl im letzten Speicherplatz des Programmspeichers verloren geht.

Mit der Null-Operation über die Tasten **2nd** **Nop** kann ein Befehl in einem Speicherplatz gelöscht und durch den Kode für einen Leerbefehl ersetzt werden. Dieser Kode wird im Programmablauf einfach ignoriert, es sei denn, er ist als Label gespeichert. Eine Anwendungsmöglichkeit dieser Taste ist, einen bestimmten Platz im Programm, der häufig geändert wird, zu reservieren. Damit können Änderungen im Programm auch ohne Einfügen und Löschen vorgenommen werden, wo auch die Programmspeicherplätze betroffen sind.



Beispiel : $x^2 + 3x - 2$ soll so eingegeben und aufbereitet werden, daß der Ausdruck wiederholt für die verschiedenen x-Werte gelöst werden kann, x sei der Anzeige, wenn der Programmablauf beginnt.

Taste	Anzeige	Bemerkungen
LRN	000 00	Eingabe des Learn-Modus
STO	001 00	Speicherung von x zur späteren Verwendung
1	001 00	
x²	003 00	
-	004 00	Falsche Eingabe
BST	003 75	Einzelschritt zurück auf 003
+	004 00	Die Eingabe - wird durch + ersetzt
4	005 00	Falsche Eingabe
X	006 00	
BST	005 65	Einzelschritte zurück zur falschen Eingabe
BST	004 04	
3	005 65	4 wird durch 3 ersetzt
SST	006 00	Einzelschritte vor bis zum Speicherplatz nach dem richtigen x
X	007 00	
RCL	008 00	
1	008 00	
BST	008 01	Sie bemerken, daß x zweimal eingegeben wurde
BST	007 43	Einzelschritte zurück bis zum Speicherplatz 006
BST	006 65	
2nd DEL	006 43	x wird gelöscht, RCL von 007 auf 006 verschoben, etc.
SST	007 01	Einzelschr. vor bis zum Speicherplatz nach RCL
SST	008 00	Einzelschr. vor bis zum Speicherplatz nach 1
2	009 00	
=	010 00	
BST	009 95	Sie bemerken eine fehlende Eingabe
BST	008 02	Einzelschr. zurück zum Speicherplatz 008
2nd INS	008 00	008 wird frei, und 2 = weitergerückt
-	009 02	Einfügen von -
SST	010 95	Einzelschritt vor bis nach Speicherplatz 2
SST	011 00	Einzelschritt vor bis nach Speicherplatz =
R/S	012 00	Programmstop, Anzeige des Resultats
RST	013 00	



Um zu kontrollieren, ob das Programm richtig eingegeben ist, gehen Sie es noch einmal in Einzelschritten durch. Das vollständige Programm muß wie folgt formuliert sein:

Speicherplatz und Tastenkod	Tastenfolge
000 42	STO
001 01	0 1
002 33	x²
003 85	+
004 03	3
005 65	X
006 43	RCL
007 01	0 1
008 75	-
009 02	2
010 95	=
011 91	R/S
012 81	RST

Bei Korrekturen ist wegen der kombinierten Tastenkodes äußerste Vorsicht angebracht. Beachten Sie nachstehende Tastenfolge.

Speicherplatz und Tastenkod	Tastenfolge
.	
.	
.	
019 95	=
020 42	STO
021 12	1 2
022 61	GTO
.	
.	
.	

Wenn man STO löscht, wird die Registernummer 12 jetzt B (Kode 12). Wenn 12 nun durch 13 ersetzt werden muß, wobei der Zeiger auf 021 eingestellt und 13 eingegeben wird, kommt die 1 in den Speicherplatz 021 und die 3 in 022, und nicht 13 in 021. Um diese letztgenannte Änderung zu erreichen, gibt man ab Speicherplatz 020 **STO** 13 ein, bzw. C (Kode 13) im Speicherplatz 021. Diese geschickte Eingabe von C, um in 021 den Kode 13 zu erhalten, ist eine sehr vorteilhafte Redigertechnik für die Änderung von Programmbefehlen.



Kennzeichnung von Programmteilen

[2nd] [Lbl] — LABEL — Kennzeichnung von Programmsegmenten nur im Learn-Modus. Der Rechner wird mit dieser Taste angewiesen, die nächste Tasteneingabe als Merkzeichen zu verwenden und nicht als Befehl, der dann ausgeführt werden muß. Die Verwendung von Labels ist vergleichbar mit dem Anbringen von Etiketten auf einem Notizbuch. Sie sind besonders vorteilhaft für Programme mit mehreren Segmenten, die einzeln oder nacheinander ablaufen können. Es gibt zwei verschiedene Typen, die allgemeinen Labels und die Programmadress-Labels.

Programmadressstasten als Labels

Eine Reihe von Tasten im oberen Teil der Tastatur, **[A]** bis **[E]** und **[2nd] [A']** bis **[2nd] [E']** werden als Programmadress-Tasten bezeichnet. Die Markierung eines Programmsegments mit einer dieser Tasten ermöglicht es, daß diese Taste im Rechenmodus manuell gedrückt wird, und Sie damit unmittelbar Zugriff auf dieses Programmsegment haben. Tatsächlich bestimmen Sie selbst die Funktion dieser Taste, und sie wirkt wie andere Funktionen auf der Tastatur. Die Folge kann den Programmspeicher vollständig belegen, sie kann aber auch so kurz sein wie im vorigen Beispiel. Wenn dieses Beispiel noch gespeichert ist, geben Sie dem Programm ein Programmadress-Label.

Taste	Anzeige	Bemerkungen
[LRN]	0	Rückkehr in den Rechenmodus
[RST]	0	Rückstellung auf 000
[LRN]	000 42	Eingabe des Learn-Modus
[2nd] [Ins]	000 00	Weiterrücken um 2 Speicherplätze
[2nd] [Ins]	000 00	
[2nd] [Lbl]	001 00	[Lbl] wird in 000 plziert
[A]	002 42	[A] wird in 001 plziert

Das Programm ist jetzt mit dem Merkzeichen A versehen und kann jederzeit durchgeführt werden, wenn man einen x-Wert in die Anzeige eingibt und dann **[A]** drückt.

Taste	Anzeige	Bemerkungen
1 [A]	2.	Der Ausdruck wird für x = 1 berechnet
1 [+/-] [A]	-4.	Für x = -1
6 [A]	52	Für x = 6
3.2 [EE]	3.2 00	
12 [A]	1.024 25	Für x = 3.2 x 10 ¹²
[2nd] [π] [÷]	3.1415927 00	
.03 [=] [A]	1.1278386 04	Für x = π/.03

Jede der Programmadress-Tasten kann einer Folge im Programm zugeordnet werden, die Sie dann nach Belieben durchführen können. Gleich ob eine Programmadress-Taste manuell gedrückt wird, oder ob sie im Programm während des Ablaufs vorkommt — in jedem Fall erfolgt eine Verschiebung zu dem Bereich des Programms, der mit der Programmadress-Taste markiert wurde, und die Verarbeitung dieses Programmteils wird dann aufgenommen.



Allgemeine Labels

Nahezu alle Tasten (einschließlich der Zweitfunktionen), können als Labels verwendet werden. Nur die folgenden Tasten sind Ausnahmen: **[2nd]**, **[LRN]**, **[Ins]**, **[Del]**, **[SST]**, **[BST]**, **[Ind]** und die Ziffern 0 bis 9. Auch die Verwendung von **[R/S]** als Label muß vermieden werden, weil mit diesem Befehl der Programmablauf gestartet wird.

Diese Labels werden wie die Programmadress-Labels zugeordnet. Aber der Zugriff über die Tastatur auf die Programmsegmente, die mit allgemeinen Labels gekennzeichnet sind, ist nicht so einfach wie mit Programmadress-Labels. Man kann zum Beispiel ein Segment mit **[cos]** markieren. Drückt man nun **[2nd]** **[cos]** auf der Tastatur, muß dieser Befehl durchgeführt werden und der Kosinus der Anzeige wird ermittelt. Allgemeine Labels haben ihre Funktion primär innerhalb des Programms in Verbindung mit Sprungoperationen, auch wenn der Zugriff über die Tastatur möglich ist.

Die Folge **[GTO]** **[2nd]** **[cos]** zum Beispiel richtet den Programmzeiger auf den Befehl unmittelbar nach dem Label „cos“. **[SBR]** **[2nd]** **[cos]** wirkt in gleicher Weise: die Verarbeitung wird ebenfalls an der Stelle aufgenommen, an der das Programm mit **[GTO]** **[2nd]** **[cos]** **[R/S]** fortgesetzt worden wäre.

Generell kann man ein Programm mit so vielen verschiedenen Labels (72 stehen zur Verfügung) wie nötig versehen. Kein Label kann mehr als ein Programmsegment markieren, aber jedes Label kann so oft wie nötig aufgerufen werden.

Mit dem Drucker können Sie eine Auflistung aller Labels im derzeit gespeicherten Programm erstellen, einschließlich ihrer Tastenkodes und ihrer Adressen im Programmspeicher. Drücken Sie einfach **[GTO]** 0 oder **[RST]** (um den Programmzeiger auf den Speicherplatz 000 einzustellen), und dann **[2nd]** **[Op]** 8. Die Tabelle wird daraufhin ausgedruckt.

Mit dem Drucker PC-100A können Sie eine Auflistung aller Labels im derzeit gespeicherten Programm erstellen, einschließlich ihrer Tastenkodes und ihrer Adressen im Programmspeicher. Drücken Sie einfach **[GTO]** 0 oder **[RST]** (um den Programmzeiger auf den Speicherplatz 000 einzustellen), und dann **[2nd]** **[Op]** 10. Die Tabelle wird daraufhin ausgedruckt.

Sprungbefehle

Wenn ein Programm abläuft, beginnt nach den bisherigen Informationen die Verarbeitung mit Speicherplatz 000 und wird bis zum nächsten Run/Stop - Befehl folgebunden weitergeführt. Mit einer Reihe von Tasten, den sogenannten Sprungbefehlen, ist eine Unterbrechung der starren Verarbeitungsfolge möglich. Es gibt bedingte und unbedingte Sprünge. Die unbedingten Sprungbefehle verschieben den Programmzeiger sofort auf den geforderten Speicherplatz. Bei den bedingten Sprüngen wird zunächst der Programmstatus getestet, und der Sprung erfolgt nur unter bestimmten Bedingungen.

Unbedingte Sprungbefehle GTO und SBR Go To - Befehl

[GTO] N oder nnn — GO TO-BEFEHL (Sprungbefehl) — In einem Programm lenkt GO TO den Verarbeitungsfluß sofort auf Label N oder auf den Speicherplatz nnn. N kann ein Programmadress-Label oder auch ein allgemeines Label sein. Gibt man den Befehl GO TO manuell über die Tastatur, wird der Programmzeiger auf den Speicherplatz nnn oder auf das mit N markierte Programmsegment eingestellt, die Durchführung des Programms wird damit aber nicht wieder aufgenommen. Die manuelle Eingabe dieses Befehls ist vor allem für den schnellen Zugriff auf einen Programmteil zu Redigier- und anderen Zwecken von Vorteil.

Der GTO-Befehl erfüllt noch eine weitere Funktion, die im Abschnitt D Seite D 8 beschrieben ist.



Unter **absoluter Adressierung** versteht man den Sprung zu einem bestimmten Programmspeicherplatz (nnn)

Beispiel: Schreiben Sie ein Programm für eine Zählung in Viererabständen.

Tastenfolge	Bemerkungen
2nd CP	Löschen des Programmspeichers, Einstellung auf Speicherplatz 000
LRN	Eingabe des Learn-Modus
2nd Lbl SUM	Das Programmsegment wird mit SUM gekennzeichnet
+ 4 =	Zu jedem Ergebnis wird 4 addiert
2nd Pause	Pause und Anzeige der Zahl
GTO SUM	Sprung zu Label SUM und Wiederholung der Folge
LRN	Aufheben des Learn-Modus
GTO SUM	Der Programmzeiger wird manuell auf das Label SUM eingestellt

Jetzt drücken Sie **R S** und Sie sehen 4, dann 8, dann 12 etc. in der Anzeige. Der Rechner führt die Folge + 4 = immer wieder aus, bis Sie mit der Taste **R S** stoppen.

Dieses Programm kann auf mehrere Arten geschrieben werden. Vorausgesetzt sei, daß jede Folge mit Speicherplatz 000 beginnt.

Tastenfolge	Tastenfolge	Tastenfolge	Tastenfolge
2nd Lbl	+	2nd Lbl	+
SUM	4	C	4
+	=	+	=
4	2nd Pause	4	2nd Pause
=	GTO	=	RST
2nd Pause	0	2nd Pause *	
GTO	0	C	
SUM	0		
Allgemeines Label	Absolute Adressierung	Programmadress- Label	Rückstellung

* Wenn das Programm Unterprogramme aufruft, ersetzen Sie diese Zeile durch **GTO** **C** .



Die Methode mit dem allgemeinen Label ist als erste angegeben. Wenn man mit absoluter Adressierung arbeitet, erübrigen sich Labels. Ist aber dieses Segment an späterer Stelle im Programmspeicher, und Sie müssen auf einer niedrigeren Adresse korrigieren, muß in bestimmten Fällen auch die absolute Adresse geändert werden. Über die Taste **C** kehrt der Programmablauf immer wieder zum Programmadress-Label C zurück und die Berechnung wird automatisch durchgeführt. Diese Methode belegt auch das Unterprogrammrücksprung-Register.

Bei der Speicherung einer drei-stelligen absoluten Adresse benötigt man zwei Speicherplätze. **GTO** 126 wird zum Beispiel in drei Plätzen gespeichert: **GTO** 01 26. Wenn man mit Adressen unter 100 arbeitet, muß nur der signifikante Teil der Adresse eingegeben werden. Drückt man **GTO** 7, belegt diese Folge automatisch auch mit der Kurzform-Adressierung drei Speicherplätze, also **GTO** 00 07, sobald eine nicht-numerische Taste gedrückt wird. Der **GTO**-Befehl muß jedoch immer mit einer mindestens ein-stelligen Adresse ergänzt werden, andernfalls wird die nächste Eingabe als Label interpretiert.

Unterprogramme

SBR N oder nnn – **UNTERPROGRAMM** – Ein Unterprogramm ist eine Befehlsfolge, die, getrennt vom Hauptteil des Programms, zur Definition einer mathematischen oder logischen Operation geschrieben werden kann. Das Hauptprogramm oder ein anderes Unterprogramm können diese Folge jederzeit aufrufen und durchführen. Unterprogramme eignen sich für Programmsituationen, die eine oder mehrere Serien von Programmschritten enthalten, die an verschiedenen Stellen wiederholt werden. Anstatt diese Schrittfolge für jede Anwendung zu wiederholen, kann man sie einmal eingeben und dann nach Bedarf aufrufen. Nachdem ein Unterprogramm aufgerufen wurde, und seine Funktion erfüllt hat, wird die Steuerung an die Programmadresse abgegeben, die unmittelbar nach der Folge für den Unterprogrammaufruf steht.

Im Learn-Modus wird mit **SBR** ein Sprung zu dem Unterprogramm mit dem Label N oder der Adresse nnn programmiert. Wenn der Unterprogrammaufruf während des Programmablaufs erfolgt, wird der Verarbeitungsfluß sofort zu dem aufgerufenen Unterprogramm gelenkt. Die Nummer des Speicherplatzes nach dem Unterprogrammaufruf wird im Unterprogrammrücksprung-Register gespeichert. Jedes Unterprogramm endet mit **INV** **SBR**. Der Rücksprung erfolgt zu dem Speicherplatz, der im Unterprogrammrücksprung-Register eingetragen ist, und von dem aus die Verarbeitung fortgesetzt wird. Die Folge **INV** **SBR** wirkt wie der Run/Stop-Befehl, wenn sie nicht in einem Unterprogramm verwendet wird. Das Unterprogrammrücksprung-Register hat eine Kapazität für sechs Rücksprungadressen. Auf diese Weise ist es möglich, daß ein Unterprogramm das andere aufruft. Sobald ein Unterprogramm abgeschlossen ist, wird die zugeordnete Rücksprungadresse aus dem Register gelöscht, um, wenn nötig, Raum für ein weiteres Unterprogramm zu schaffen.

Wird der Unterprogrammbefehl manuell über die Tastatur gegeben, erfolgt der Programmsprung zu dem geforderten Label oder der Adresse und die Verarbeitung wird automatisch aufgenommen.



Beispiel: Berechnen Sie $x^2 - 3x - 2$ für x_1 und x_2 . Speichern Sie x_1 in R_{01} und x_2 in R_{02} und summieren Sie die beiden Ergebnisse in R_{03} .

Weil $x^2 - 3x - 2$ zweimal berechnet werden muß, wird für diese Aufgabe ein Unterprogramm erstellt. Der Beginn bei Adresse 030 ist willkürlich. Drücken Sie **GTO** 30 **LAN** und geben Sie nachstehende Folge ein:

Speicherplatz und Tastenkodes	Tastenfolge	Bemerkungen
030 76	2nd Lb	Programmsegment, das mit Label CE markiert ist
031 24	CE	
032 53	(
033 42	STO	Speicherung von x für d. spätere Anwendung
034 05	5	
035 33	x²	
036 75	-	
037 03	3	
038 65	X	
039 43	RCL	Aufruf von x
040 05	5	
041 75	-	
042 02	2	
043 54)	Berechnung d. Ausdrucks
044 44	SUM	Das Ergebnis wird in Register 03 summiert
045 03	3	
046 92	INV SBR	Rücksprung zum Hauptprogramm

Beachten Sie, daß zur Berechnung des Ausdrucks Klammern verwendet wurden. Auch eine Gleichheitsanweisung wäre möglich gewesen, aber sie hätte alle unvollständigen Operationen im Hauptprogramm und im Unterprogramm abgeschlossen. Gehen Sie kein Risiko ein und setzen Sie Klammern. Jetzt kann das Hauptprogramm geschrieben werden, um die x-Werte richtig einzusetzen und die Ergebnisse anzuzeigen.



Speicherplatz und Tastenkodes	Tastenfolge	Bemerkungen
000 43	RCL	Aufruf von x_1
001 01	1	
002 71	SBR	Aufruf von Unterprogramm CE
003 24	CE	
004 43	RCL	Aufruf von x_2
005 02	2	
006 71	SBR	Aufruf von Unterprogramm CE
007 24	CE	
008 43	RCL	Aufruf des Resultats in Register 03
009 03	3	
010 91	R/S	Halt und Anzeige d. Inhalts von R_{03}
011 81	RST	Rückstellung auf 000 für die nächsten Variablen; auch das Unterprogrammrückprung-Register wird wieder auf Null gestellt.

Wenn das Unterprogramm aufgerufen wird, verzweigt der Verarbeitungsfluß zum Label CE (Speicherplatz 030), das Unterprogramm wird abgeschlossen, und die Rücksprungadresse ist der Speicherplatz 004. Die Rücksprungstelle des nächsten Unterprogramms ist der Speicherplatz 008. Der Rücksprung erfolgt immer vom Unterprogramm zu dem Speicherplatz, der unmittelbar nach dem Befehl folgt, der das Unterprogramm aufruft.

Dazu kommt, daß Unterprogramme, die vom Hauptprogramm aufgerufen wurden, selbst weitere Unterprogramme aufrufen können. Das Unterprogrammrückprung-Register kontrolliert automatisch, wohin nach Abschluß eines Unterprogramms der Rücksprung erfolgen muß.

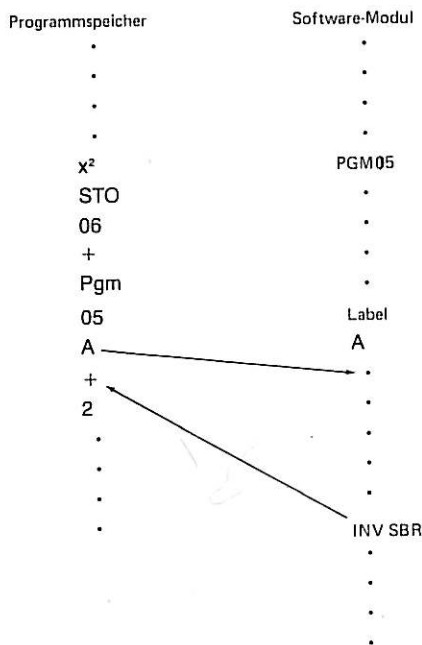
Wenn ein Programm eingetastet ist, und man drückt **RST**, um auf Speicherplatz 000 zurückzukommen, wird damit auch das Unterprogrammrückprung-Register auf Null gestellt.

SOFTWARE-PROGRAMME ALS UNTERPROGRAMME

Jedes Programm eines im Rechner eingesetzten Software-Moduls kann als Unterprogramm aufgerufen werden. Die Programme auf jedem Modul sind speziell zu diesem Zweck formuliert, wobei jeder Teil eines Programms ein Unterprogramm ist. Jeder Teil ist mit einem Label gekennzeichnet, endet mit **INV SBR** und enthält meistens keine Run/Stop-Befehle. Informationen über die Kennzeichnung jedes Programmtails siehe die Bücher zu den Softwareprogrammen.



Sie wissen, wie der Zugriff auf Software-Programme über die Tastatur möglich ist. Wenn ein Programm im Programmspeicher abläuft, können Sie ein Software-Programm aufrufen und beliebige Teile davon als Unterprogramm verwenden. Ist das Segment mit einer Programmadress-Taste markiert, überträgt die Folge **[2nd] [Pgm] mm,N** im Programmspeicher die Verarbeitung auf das Software-Programm mm, und die Ausführung wird im Software-Programm bei Label N fortgesetzt. Der Zugriff auf ein allgemeines Label in einem Software-Programm erfolgt mit den Tasten **[2nd] [Pgm] mm [SBR] N**. Wenn das Software-Programmsegment mit dem Label N seinen Zweck erfüllt hat, richtet sich der Programmzeiger auf den Befehl, der der Stelle der Abweichung vom Programmspeicher direkt folgt, und die Verarbeitung wird von hier aus wieder aufgenommen.



In einem Programm wirkt die Befehlsfolge **[2nd] [Pgm] mm,N** wie **[SBR] N** : die Adresse nach diesem Befehl wird im Unterprogrammrücksprung-Register gespeichert, und die Verarbeitung wird sofort auf den Programmabschnitt mit dem Label N gelenkt. Sobald die Folge nach Label N abgeschlossen ist, wird die Verarbeitung an der Adresse fortgesetzt, die als letzte im Unterprogrammrücksprung-Register gespeichert wurde. Nach **[2nd] [Pgm] mm** muß **[SBR]** oder eine Programmadresstaste folgen. Eine andere Tastenfolge ist unzulässig und kann zu fehlerhaften Ergebnissen führen.



Die Anwendung der Software-Programme als Unterprogramme erhöht die Rechenkapazität beträchtlich. Es gibt jedoch einige bedenkliche Aspekte bei der Programmierung, wenn zwei separate Programme kombiniert werden. Die Programme dürfen sich niemals in der Anwendung gemeinsamer Datenspeicherregister oder Flags gegenseitig überlagern. Ein Software-Programm kann auch andere Unterprogramme aufrufen, Sie müssen aber sicherstellen, daß nicht mehr als 6 Unterprogrammebenen beansprucht werden. Im allgemeinen überschreiten Sie diese Kapazität nicht, wenn Sie nicht mehr als drei Unterprogrammebenen im Programmspeicher verwenden. Dazu kommt, daß auch das Anzeigeformat des Software-Programms auf Festkomma, auf Exponentialform oder auf technische Notation eingestellt sein kann. In der folgenden Tabelle sind die verschiedenen Anwendungsmöglichkeiten der Unterprogramme kurz zusammengefaßt.

UNTERPROGRAMMBEFEHLE

Von	Zu	Tastenfolge
Programmspeicher oder Software-Programm	Software-Programm	2nd Pgm mm N (Programmadress-Label
		2nd Pgm mm SBR N (allgemeines Label) oder
		2nd Pgm mm SBR nnn
		2nd Pgm mm R/S
Programmspeicher Software-Programm	Programmspeicher	SBR nnn oder N
	Programmspeicher	2nd Pgm 00 N (oder nnn) oder
		2nd Pgm 00 SBR N oder
		RST

Bedingte Sprünge (Testbefehle)

Diese Befehle verzweigen das Programm nur dann, wenn der Wert im Anzeigeregister eine bestimmte Bedingung erfüllt: Welchen Status hat der derzeitige Wert (im Anzeigeregister) im Vergleich zu einem anderen Wert, (der im T-Register gespeichert ist), oder sind bereits genug Schritte verarbeitet (Dekrement und Überspringen bei Null) ?

T-REGISTER VERGLEICHE

x=t	Austausch des Anzeigeregister-Wertes x gegen den Wert t des T-Registers;
2nd x=1 N oder nnn	Frage: „Ist der Wert des Anzeigeregisters gleich dem Wert des T-Registers?“
INV 2nd x=1 N oder nnn	Frage: „Ist der Wert des Anzeigeregisters ungleich dem Wert des T-Registers?“
2nd x=1 N oder nnn	Frage: „Ist der Wert des Anzeigeregisters größer oder gleich dem Wert des T-Registers?“
INV 2nd x=1 N oder nnn	Frage: „Ist der Wert des Anzeigeregisters kleiner als der Wert des T-Registers?“

Kann eine der obigen Fragen mit „ja“ beantwortet werden, verzweigt der Verarbeitungsfluß zu der Adresse, die dem Befehl unmittelbar folgt. Ist die Antwort „nein“, wird die entsprechende Adresse übersprungen, und die Verarbeitung wird mit dem nächsten Befehl fortgesetzt.



Diese Testbefehle wirken nicht auf unvollständige Operationen. Sie können also beliebig in einem Programm durchgeführt werden.

Beispiel:

Speicherplatz und Tastenkod	Tastenfolge
022 43	RCL
023 01	1
024 95	=
025 77	2nd $x \geq 1$
026 12	B
027 61	GTO
028 00	3
029 03	
030 76	2nd Lbl
031 12	B
032 91	R/S

Das Ergebnis, das man im Speicherplatz 024 erhält, wird im Speicherplatz 025 getestet, ob es größer oder gleich dem im T-Register gespeicherten Wert ist. Ist die Antwort ja, wird der Verarbeitungsfluß zum Label B gelenkt, wo das Programm unterbrochen und das Ergebnis angezeigt wird. Ist die Antwort nein, wird der Befehl, zum Label B zu verzweigen, übergangen. Mit dem folgenden Sprungbefehl GTO 003 wird die Fortsetzung der Verarbeitung bei Adresse 003 veranlaßt.

ANMERKUNG: Bedingte Verzweigungen speichern die Rücksprungadresse nicht im Unterprogrammrücksprungs-Register. Wenn Sie ein Unterprogramm aufrufen müssen und auf einen bedingten Test zurückgehen, soll der Test der erste Schritt im Unterprogramm sein. **2nd** **$x \geq 1$** **x^2** speichert zum Beispiel keine Rücksprungadresse. Bei **SBR** **x^2** und dann **2nd** **Lbl** **x^2** **2nd** **$x \geq 1$** wird dagegen die Rücksprungadresse gespeichert.

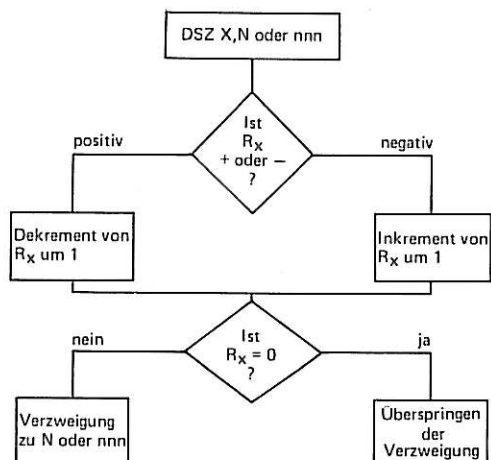
DEKREMENT UND ÜBERSPRINGEN BEI NULL

2nd **0Sz** X, N oder nnn – DEKREMENT UND ÜBERSPRINGEN BEI NULL – Die Größe des Datenregisterinhalts X (0 bis 9) wird um 1 vermindert, und die Verarbeitung verzweigt zum Label N oder zum Speicherplatz nnn, wenn $R_x \neq 0$. Bei R_x gleich Null wird die Verzweigungsadresse übersprungen. R_x steht hier für den Inhalt des Datenregisters X.

INV **2nd** **0Sz** X, N oder nnn verzweigt zum Label N oder zum Speicherplatz nnn, wenn $R_x = 0$.



Dieser leistungsfähige Programmierbefehl ist eine Kombination aus Zähl- und Testbefehl. Wenn die Wiederholung einer Folge y mal erforderlich ist, speichern Sie y in einem der Datenregister 0 bis 9 und programmieren einen DSZ-Testbefehl in die iterative Folge. Nach y Iterationen wird die Schleife beendet und das Programm fortgesetzt. Der DSZ-Befehl operiert wie folgt:



programmierter Befehl

Ist R_x positiv oder negativ?

Die Größe von R_x wird um 1 reduziert

Ist $R_x = 0$?

Bei R_x ungleich 0 Verzweigung zu N oder nnn;
Bei R_x gleich 0 Überspringen der Verzweigungs-
adresse und Fortsetzung der Verarbeitung.

Die Skizze zeigt, daß mit dem DSZ-Befehl zu Beginn einer Folge zunächst gezählt und der Rechengang dann ausgeführt wird. DSZ am Ende einer Folge berechnet zuerst die Funktion, dann findet die Zählung statt. Für die richtige Anzahl von Durchläufen einer Folge bedeutet dies:

Zuerst Eingabe von y in das Register X und Durchführung des DSZ-Befehls am Ende, oder Durchführung des DSZ-Befehls am Anfang, aber dann wird zunächst $y + 1$ im Register X gespeichert.



Beispiel: Schreiben Sie ein Programm zur Berechnung von $F!$ (F-Fakultät), wobei $F! = 1 \times 2 \times 3 \times \dots \times F$. (0! ist als 1 definiert)

Speicherplatz und Tastenkodes	Tastenfolge	Bemerkungen
000 76	2nd Lbl	
001 15	E	
002 42	STO	F wird in Register 00 gespeichert
003 00	0	
004 29	2nd CP	Löschen des T-Registers
005 67	2nd z=1	Test : $F = 0?$
006 11	A	Wenn ja, Verzweigung zu A
007 76	2nd Lbl	
008 12	B	
009 43	RCL	Aufruf von F
010 00	0	
011 65	X	
012 97	2nd 057	Verminderung von R_{00} um 1
013 00	0	
014 12	B	Ist R_{00} nicht im Bereich ± 1 , Verzweigung zu B
015 76	2nd Lbl	Ist R_{00} im Bereich ± 1 , Fortsetzung d. Ablaufs bis zum Ende
016 11	A	
017 01	1	
018 95	=	
019 91	R/S	Stop und Anzeige von $F!$

Um diese Folge durchzuführen, gibt man eine Zahl für F ein und drückt dann **E**. F muß kleiner als 70 sein, andernfalls wird die Kapazität des Rechners überschritten, weil $70! > 9.9999999 \times 10^{99}$.

FLAGS

2nd **ST/IF** y – SET FLAG – Dieser Befehl setzt ein Flag Nummer y, wobei $0 \leq y \leq 9$. Mit der Folge **INV** **2nd** **ST/IF** wird Flag y auf Null gesetzt, d. h. rückgesetzt.

2nd **IF/IF** y, N oder nnn – FLAG-TEST – Flag y wird getestet, ob es gesetzt ist. Wenn ja, erfolgt ein Programmsprung zu Label N oder zum Speicherplatz nnn

INV **2nd** **IF/IF** y;N oder nnn – UMKEHRUNG DES FLAG-TESTS – Flag y wird getestet und das Programm verzweigt dann zu Adresse N oder nnn, wenn Flag y nicht gesetzt ist.



Für die Anwendung im Programm stehen 10 Flags (0 bis 9) zur Verfügung. Diese Flags sind zu Beginn auf Null gesetzt. Man verwendet sie zur Kontrolle des Rechenwegs im gesamten Programmablauf oder zur manuellen Steuerung von Programmoptionen vor der Ausführung. Gibt man die Anweisungen **RST** oder **[2nd] [CP]** manuell über die Tastatur, werden alle Flags auf Null gesetzt und das Unterprogrammrücksprung-Register wird gelöscht. Ein RST-Befehl im Programm setzt ebenfalls alle Flags auf Null.

Beispiel: Erstellen Sie eine Programmfolge, mit der alle eingehenden Zahlen summiert und die positiven Zahlen ausgedruckt werden. Die Summe soll nach jeder Eingabe ausgewiesen werden.

Speicherplatz und Tastenkode	Tastenfolge	Bemerkungen
018 76	[2nd] [LbI]	
019 11	[A]	
020 22	[INV]	
021 86	[2nd] [St/If]	Flag 3 muß auf Null gesetzt werden
022 03	[3]	
023 77	[2nd] [x=1]	„Ist die Zahl Null oder positiv?“
024 12	[B]	Wenn ja, Sprung zu Label B
025 86	[2nd] [St/If]	Ist die Zahl negativ, wird Flag 3 gesetzt
026 03	[3]	
027 76	[2nd] [LbI]	
028 12	[B]	
029 44	[SUM]	Summierung aller Zahlen
030 12	[1] [2]	
031 87	[2nd] [St/If]	„Ist Flag 3 gesetzt?“
032 03	[3]	
033 13	[C]	Wenn ja (negative Zahl), Sprung zu C
034 99	[2nd] [Pi]	Wenn nein, Ausdruck der Zahl
035 76	[2nd] [LbI]	
036 13	[C]	
037 43	[RCL]	
038 12	[1] [2]	
039 91	[R/S]	

Kontrollieren Sie, daß vor Eingabe einer Reihe von Zahlen das Datenregister 12 nicht belegt ist. Wenn die Zahl eingegeben und **[A]** gedrückt wird, wird diese Zahl zum Inhalt des Datenregisters 12 addiert, eine positive Eingabe wird ausgedruckt und die Gesamtsumme aus allen Eingaben angezeigt.



FLAGS UND FEHLERBEDINGUNGEN

Flag 7 ist reserviert, um eine Programmoperation im Hinblick auf den Fehlerstatus eines Programms zu bestimmen. Im allgemeinen wird der Ablauf fortgesetzt, auch wenn eine Fehlerbedingung vorkam. Gleich, ob Flag 8 manuell oder im Programm gesetzt wird, in jedem Fall wird die Ausführung des Programms unterbrochen, wenn eine Fehlerbedingung auftritt.

2nd Op 18 veranlaßt, daß Flag 7 gesetzt wird, wenn in einem Programm keine Fehlerbedingung existiert.

2nd Op 19 veranlaßt, daß Flag 7 gesetzt wird, wenn eine Fehlerbedingung in einem Programm vorliegt. Flag 7 kann jetzt zur Bestimmung des Fehlerstatus Ihres Programms kontrolliert werden, und Sie haben die Möglichkeit zu folgerichtigen Reaktionen. Wenn einer dieser Tests falsch ist, wird Flag 7 nicht geändert.

2nd Op 40 (nur für den TI-58C) weist den Rechner an, Flag 7 zu setzen, wenn der Drucker angeschlossen ist. Flag 7 kann getestet werden, um das Vorhandensein einer Druckerfunktion innerhalb eines Programms festzustellen. Wenn der Drucker nicht angeschlossen ist, wird Flag 7 nicht geändert.



Indirekte Adressierung

[2nd] [Ind] XX – **INDIREKTER INDEX** – Wenn man diesen Befehl einer der folgenden Operationen nachstellt, wird der Inhalt des Datenregisters XX aufgerufen, und dieser Inhalt ist die korrekte Adresse für eine Verzweigung oder für das tatsächliche Datenregister, das im weiteren Ablauf zu verwenden ist.

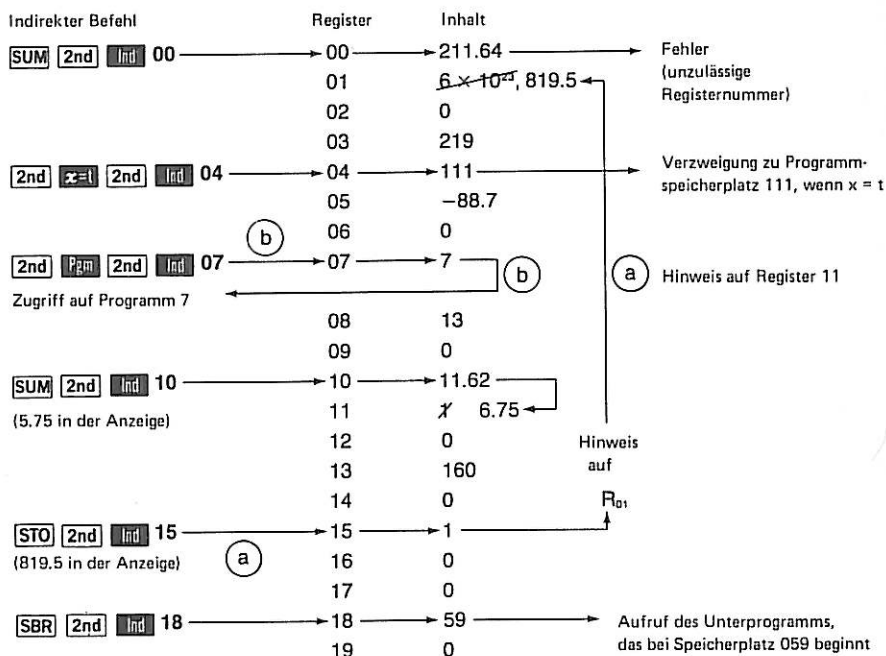
Tastenfolge	Tastenkodes	Zweck
[STO] [2nd] [Ind] XX	72 XX	Indirekte Speicherung
[RCL] [2nd] [Ind] XX	73 XX	Indirekter Aufruf
[2nd] [ExC] [2nd] [Ind] XX	63 XX	Indirekter Austausch
[SUM] [2nd] [Ind] XX	74 XX	Indirekte Speicheraddition
[INV] [SUM] [2nd] [Ind] XX	22 74 XX	Indirekte Speichersubtraktion
[2nd] [Prd] [2nd] [Ind] XX	64 XX	Indirekte Speichermultiplikation
[INV] [2nd] [Prd] [2nd] [Ind] XX	22 64 XX	Indirekte Speicherdivision
[GTO] [2nd] [Ind] XX	83 XX	Indirekter Sprungbefehl
[2nd] [PgM] [2nd] [Ind] XX	62 XX	Indirekter Programmaufruf
[2nd] [Op] [2nd] [Ind] XX	84 XX	Indirekte Steueroperation
[SBR] [2nd] [Ind] XX	71 40 XX	Indirektes Unterprogramm
[2nd] [$x \geq t$] [2nd] [Ind] XX	77 40 XX	Indirekter $x \geq t$ -Test
[INV] [2nd] [$x \geq t$] [2nd] [Ind] XX	22 77 40 XX	Indirekter $x < t$ -Test
[2nd] [$x = t$] [2nd] [Ind] XX	67 40 XX	Indirekter $x = t$ -Test
[INV] [2nd] [$x = t$] [2nd] [Ind] XX	22 67 40 XX	Indirekter $x \neq t$ -Test
[2nd] [Fix] [2nd] [Ind] XX	58 40 XX	Indirekte Festkomma-Einstellung
[2nd] [St flg] [2nd] [Ind] XX	86 40 XX	Indirekte Flagsetzung
[INV] [2nd] [St flg] [2nd] [Ind] XX	22 86 40 XX	Indirekte Flagrücksetzung
[2nd] [DSZ] [2nd] [Ind] XX, N oder nnn	97 40 XX	Indirekter Register-DSZ-Test
[2nd] [DSZ] X [2nd] [Ind] XX	97 X 40 XX	Indirekter Adressen-DSZ-Test
[2nd] [DSZ] [2nd] [Ind] XX [2nd] [Ind] yy	97 40 XX 40 yy	Indirekter Register- und Adressen-DSZ-Test
[INV] [2nd] [DSZ] [2nd] [Ind] XX, N oder nnn	22 97 40 XX	Indirekter Register-/Überspringen bei nicht Null-Test
[INV] [2nd] [DSZ] X [2nd] [Ind] XX	22 97 X 40 XX	Indirekter Adressen-/Überspringen bei nicht Null-Test
[INV] [2nd] [DSZ] [2nd] [Ind] XX [2nd] [Ind] yy	22 97 40 XX 40 yy	Indirekter Register- und Adressen-/Überspringen bei nicht Null-Test
[2nd] [flg] [2nd] [Ind] XX, N oder nnn	87 40 XX	Indirekte Flag-Nummer, Flag-Test
[2nd] [flg] X [2nd] [Ind] yy	87 X 40 yy	Indirekte Adresse, Flag-Test
[2nd] [flg] [2nd] [Ind] XX [2nd] [Ind] yy	87 40 XX 40 yy	Indirekte Flagnummer und Adresse, Flag-Test
[INV] [2nd] [flg] [2nd] [Ind] XX, N oder nnn	22 87 40 XX	Indirekte Flagnummer, Flag-Testumkehrung
[INV] [2nd] [flg] [2nd] X [2nd] [Ind] XX	22 87 X 40 XX	Indirekte Adresse, Flag-Testumkehrung
[INV] [2nd] [flg] [2nd] [Ind] XX [2nd] [Ind] yy	22 87 40 XX 40 yy	Indirekte Flagnummer und Adresse, Flag-Testumkehrung

Beachten Sie, daß den indirekten Speicheroperationen, den PGM-, Op- und Sprung-Befehlen Codes zugeordnet sind, die keine Beziehung zu deren Position auf der Tastatur haben. Die Zuordnung dieser Codes erfolgte unter dem Aspekt der Einsparung von Programmspeicherplätzen.



Der indirekte Befehl bewirkt eine Adressenmodifikation in der Form, daß die endgültige Adresse im Datenregister XX aufgefunden wird. Mit der Folge **RCL 2nd Ind 04** wird also nicht der Inhalt von Register 04 aufgerufen, sondern R_{04} gibt nur die geforderte Speicheradresse an. Wenn im Register 04 die Zahl 111 gespeichert ist, wird 111 die eigentlich benötigte Adresse. Die indirekte Adresse zeigt die tatsächliche Adresse auf.

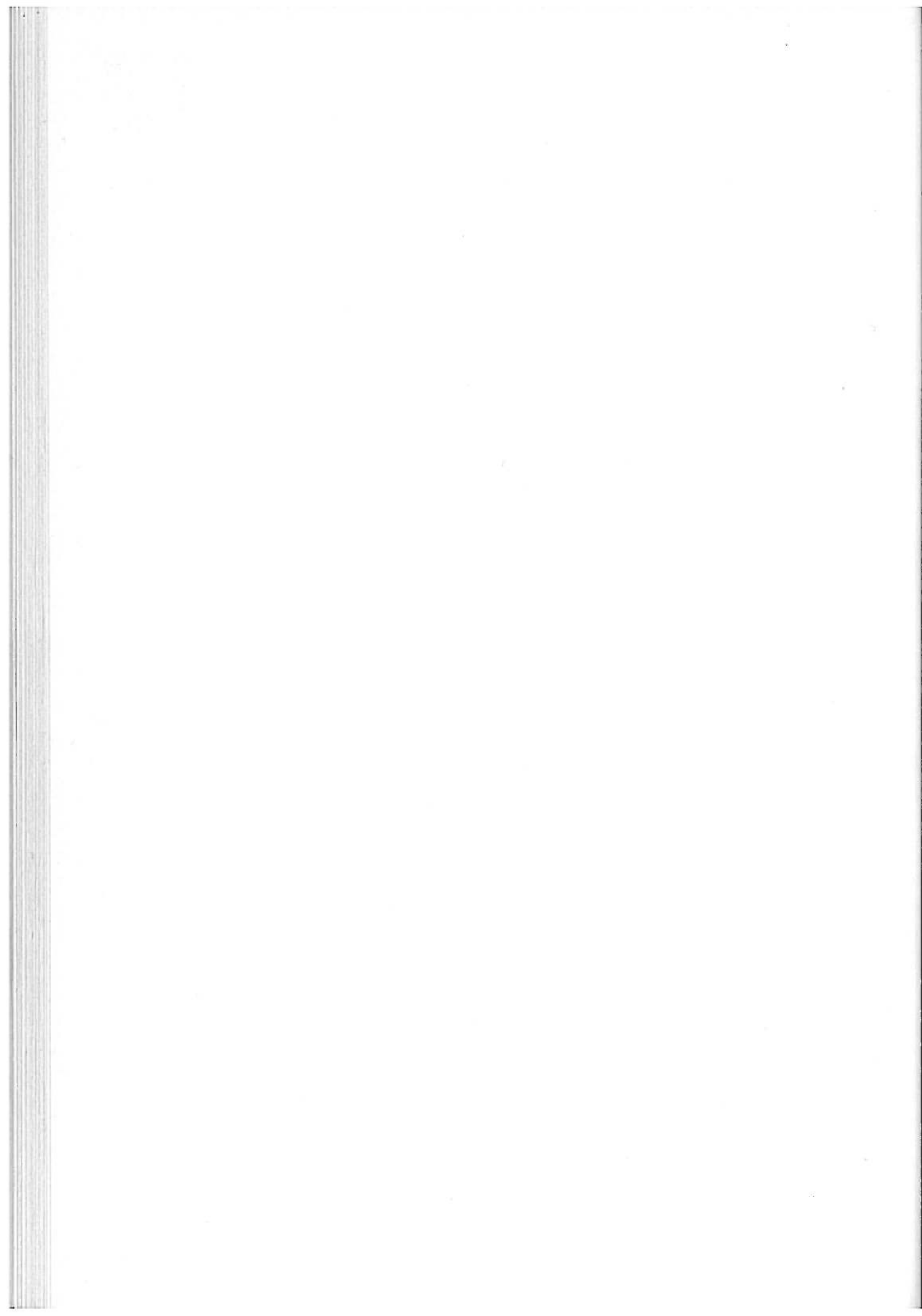
Das folgende Diagramm stellt dieses Konzept graphisch dar.



Das Diagramm zeigt die Wirkung von **2nd ~~cc=t~~ 2nd Ind 04**. Die Folge ist, daß der Wert in R_{04} (111) als die neue Adresse für den **~~cc=t~~**-Befehl verwendet wird. Die Konsequenz aus **STO 2nd Ind 15** wird in dem Ablauf der Vorgänge dargestellt, der mit (a) markiert ist. Das Resultat hier ist, daß 819.5 in R_{01} gespeichert wird, das zuvor den Wert 6×10^{23} enthielt. Schließlich ist die Wirkung von **2nd Pgm 2nd Ind 07** in diesem Beispiel mit (b) markiert. Der Zeiger richtet sich auf das Datenregister 07 mit der Folge, daß der Wert 7 als Zugriff auf das Software-Programm Nummer 7 verwendet wird.

Voraussetzung ist, daß jeder Hinweis, der als Konsequenz eines indirekten Befehls verwendet wird, eine wirklich realisierbare Adresse aufzeigt. Demnach wäre das Ergebnis aus **SUM 2nd Ind 00** in dem obigen Diagramm ein Fehler, — es gibt kein Datenregister mit der Adresse 211.64. Wäre der Inhalt von Register 00 die Zahl 111.64 gewesen, würden die Werte im Register 11 summiert, weil nur der ganzzahlige Teil einer indirekten Adresse benutzt wird.

Ist der Wert im indirekten Register kleiner als Null, nimmt der Rechner das Register 00 an. Liegt der Wert nicht im Bereich der Verteilung, wird die Verarbeitung unterbrochen und die Anzeige blinkt.





Der wahlweise lieferbare Drucker PC-100A, PC-100B oder PC-100C * kann in Verbindung mit Ihrem programmierbaren Rechner eine Reihe verschiedener Druckaufgaben durchführen. Im einzelnen sind folgende Operationen möglich, wenn der Rechner auf den Drucker aufgesetzt ist :

1. Ausdruck des Inhalts der Anzeige zu jeder Zeit.
2. Auflisten des Programms im Programmspeicher und der Inhalte aller Datenregister.
3. Ausdruck der Ergebnisse von jeder beliebigen Stelle des ablaufenden Programms.
4. Ausdruck jedes Schrittes der Rechneroperation durch Protokollierung im Parallelbetrieb. Die Berechnungen können dabei manuell über die Tastatur oder in einem Programm durchgeführt werden.
5. Auflistung aller Labels und deren Adressen.
6. Ausdruck alphanumerischer Nachrichten, wann immer Bedarf besteht.
7. Erstellen einer Aufzeichnung von Daten (Plotten), die manuell eingegeben werden können oder die automatisch aus einem Programm kommen.
8. Sie können den Schreibtisch verlassen, ohne den Rechner wegschließen zu müssen. Der Drucker sichert den Rechner und versorgt ihn mit Strom.
9. Die aufladbare Batterie des Rechners kann geladen werden, während der Drucker arbeitet.

Die Bedienungsanleitung für diesen vielseitigen Drucker ist im Lieferumfang enthalten, hier bekommen Sie aber noch mehrere zusätzliche wichtige Informationen.

Die Kontrollsymbole und das Programm für die Druckkopfreinigung sind bei diesen Rechnern anders und werden später noch behandelt.

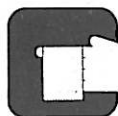
Der Rechner-Wählschalter (nur PC-100A), der sich im Ladefach des Druckers befindet, muß auf "OTHER" geschoben werden, wenn der Drucker in Verbindung mit den Rechnern TI-58/TI-58C oder TI-59 verwendet wird. Die PC-100B und PC-100C sind so ausgelegt, daß er nur mit den programmierbaren Rechnern TI-58/58C oder TI-59 benutzt werden kann.

Die Grundanweisungen für den Druckerbetrieb, das Aufsetzen des Rechners, das Austauschen des Papiers und die Service-Informationen erhalten Sie zusammen mit dem Drucker.

Der Abschnitt, den Sie jetzt lesen, beschreibt die Drucker-operationen, die in der Rechner/Drucker-Kombination durchgeführt werden können. Wenn Sie den Drucker benutzen, wenden Sie für die Reinigung der Druckköpfe das Programm auf Seite VI-12 dieses Handbuchs an, wie es in der Bedienungsanleitung für den Drucker beschrieben ist.

WICHTIG : Die Constant Memory - Eigenschaft des TI-58C erlaubt die Entfernung des Batteriepakets, um den Rechner auf den Drucker aufzusetzen, ohne daß dabei Programm- und Datenspeicher beeinflußt werden (achten Sie aber darauf, ein laufendes Programm zu unterbrechen den Rechner abzuschalten und das Netzadapter/Ladegerät abzutrennen. Wenn Sie einen PC-100A oder PC-100B benutzen, muß der Einschalter des Druckers auf ON stehen und der Schlüssel in der Absperrposition sein, um die Inhalte von Programm- und Datenspeicher zu erhalten.

*Anmerkung: Der PC-100 kann nicht mit den programmierbaren Rechnern TI-58, TI-58C und TI-59 arbeiten.



WAHLWEISER AUSDRUCK NACH BEDARF

Wenn man **2nd** **Prt** manuell drückt, oder wenn dieser Befehl in einem Programm vorkommt, wird der Wert in der Anzeige ausgedruckt. Ob Sie die Taste **Prt** auf dem Drucker oder die Folge **2nd** **Prt** auf der Rechner tastatur drücken, ist beliebig.

Beachten Sie folgendes Programm, bei dem Vielfache von 4 ausgedruckt werden:

Speicherplatz und Tastencode	Tastenfolge
000 85	+
001 04	4
002 95	=
003 99	2nd Prt
004 81	RST

Geben Sie einen Anfangspunkt ein und drücken Sie **RST** **R/S**, und Sie erhalten folgenden Ausdruck:

Ergebnisse

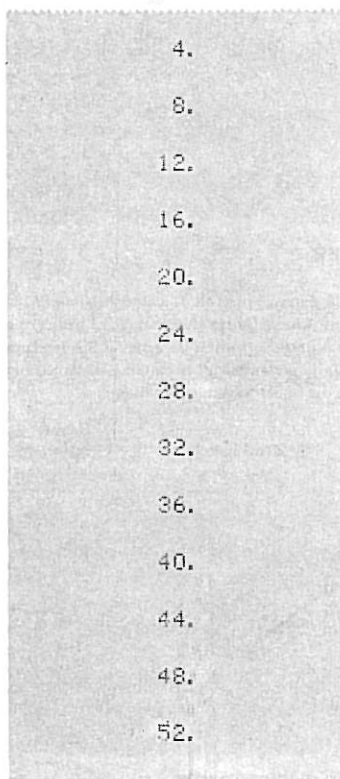
4.
8.
12.
16.
20.
24.
28.
32.
36.
40.
44.
48.
52.
56.
60.
64.
68.
72.
76.
80.
84.
88.
92.
96.
100.



Der Modul für die Software-Programme enthält auch eine ganze Gruppe von Druckbefehlen. Wenn also ein Drucker zur Verfügung steht, können Eingaben und Resultate mehrerer Software-Programme automatisch ausgedruckt werden.

Einzelne Gruppen gedruckter Ergebnisse können Sie mit der Papiervorschub-Taste **2nd** **Adv** auf der Rechnertastatur von anderen abgrenzen. Wenn dieser Befehl in einem Programm vorkommt, wird das Papier um eine Zeile ohne Drucken weitergeschoben. Um zum Beispiel die Vielfachen von 4 abzugrenzen, fügt man die Anweisung **2nd** **Adv** in einem beliebigen Speicherplatz vor **RST** ein, und Sie erhalten folgenden Ausdruck. Der Papiervorschubbefehl hat keinen Einfluß auf Berechnungen.

Ergebnisse



4.
8.
12.
16.
20.
24.
28.
32.
36.
40.
44.
48.
52.

Um mehr Leerzeilen zu erhalten, wird der Befehl mehrere Male wiederholt.

Die Taste **ADV** auf dem Drucker erzeugt ebenfalls Leerzeilen. Das Papier wird weitergeschoben, solange die Taste gedrückt bleibt.



AUFLISTEN IHRER PROGRAMME

Zum Auflisten eines Programms drückt man einfach **2nd** **List** auf der Tastatur. Das Programm wird dann von der derzeitigen Position des Programmzeigers an aufgelistet. Mit der Taste **R/S** kann die Auflistung jederzeit manuell unterbrochen werden. Für eine vollständige Programmauflistung drückt man **RST** **2nd** **List**. Nachstehend die Auflistung des Programms zur Berechnung der Vielfachen von 4:

Programm- speicherplatz	Tasten- kode	Tasten- symbol
----------------------------	-----------------	-------------------

000	85	+
001	04	4
002	95	=
003	99	PRT
004	98	ADV
005	81	RST
006	00	0
007	00	0

AUFLISTEN DER DATENREGISTER

Die Folge **INV** **2nd** **List** (bei manueller Eingabe oder als Teil eines Programms) listet die Inhalte aller Datenregister auf, beginnend mit der Registeradresse, die in der Anzeige ausgewiesen ist. Die Auflistung wird so lange fortgesetzt, bis auch der Inhalt des Datenregisters mit der höchsten Adresse aufgelistet ist, oder bis Sie den Druckvorgang mit der Taste **R/S** unterbrechen. Der Rechner ist jetzt wieder auf Tastatursteuerung geschaltet. Eine Auflistung von willkürlichen Registerinhalten ab Register 50 bis zur Verteilung bei 59 sehen Sie in der folgenden Abbildung.

Registerinhalt	Register- adresse
----------------	----------------------

14.12181818	50
665.8568182	51
110.9761364	52
0.	53
0.	54
-5.5488068-12	55
0.	56
0.	57
1.	58
0.085106383	59

Die Auflistung wurde mit der Tastenfolge **50** **INV** **2nd** **List** eingeleitet.

Wenn diese Operation in einem Programm durchgeführt wird, werden die Berechnungen eingestellt, die geforderten Datenregisterinhalte aufgelistet, und das Programm wird unterbrochen, wobei die Steuerung wieder an die Tastatur gegeben wird.



PROTOKOLLIEREN IHRER BERECHNUNGEN (PARALLEL BETRIEB)

Mit der Taste **TRACE** auf dem Drucker veranlaßt man, daß jeder Schritt einer Berechnung ausgedruckt wird. Der berechnete Wert und der entsprechende Befehl werden angezeigt. Dies gilt sowohl für manuelle Berechnungen über die Tastatur als auch für Berechnungen in Programmen.

Die Taste **TRACE** ist ein selbsthaltender Schalter, der beim Drücken einrastet, und mit dem die Protokollierung aller Berechnungen veranlaßt wird. In diesem Parallelbetrieb werden jede neue Funktion und jedes neue Ergebnis automatisch ausgedruckt. Eine Zahleneingabe wird nur dann gedruckt, wenn eine Operation oder eine Funktion folgt. Die Operation im Drucker-parallelbetrieb wird solange fortgesetzt, bis die Taste **TRACE** wieder gelöst wird.

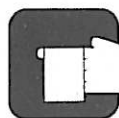
Mit dem Programm zur Berechnung der Vielfachen von 4 im Programmspeicher drücken Sie die Taste **TRACE** auf dem Drucker, und dann **CLR** **RST** **R/S** auf dem Rechner, um das folgende "Protokoll" zu den Berechnungen zu erhalten.

Anzeige- register	Kontroll- symbole
0.	+
4.	=
4.	
4.	PRT
	RST
4.	+
4.	=
8.	
8.	PRT
	RST
8.	+
4.	=
12.	
12.	PRT
	RST
12.	+
4.	=
16.	
16.	PRT

Das Programm (und folglich auch die Protokollierung) wurden mit der Taste **R/S** gestoppt.

KONTROLLSYMBOLE IM DRUCKER-PARALLEL BETRIEB

Die meisten der ausgedruckten Symbole können ohne Schwierigkeiten identifiziert werden, während andere nicht auf den ersten Blick zugeordnet werden können. Auf der nächsten Seite finden Sie eine vollständige Liste der Kontrollsymbole und der dazugehörigen Tastenfolgen.



Druck- symbol	Tastenfolge
A - E	A - E
A' - E'	2nd A - 2nd E
ADV	2nd Adv
BST	siehe Anmerkung unten
CE	CE
CLR	CLR
CP	2nd CP
CMS	2nd CMS
COS	2nd COS
DEG	2nd DEG
DEL	siehe Anmerkung unten
DMS	2nd DMS
DSZ	2nd DSZ
EE	EE
ENG	2nd ENG
EQ	2nd EQ
EX*	2nd EX 2nd Ind
EXC	2nd EXC
FIX	2nd FIX
GE	2nd GE
GO*	GTO 2nd Ind
GRD	2nd GRD
GTO	GTO
IEQ	INV 2nd EQ †
IGE	INV 2nd GE †
IS+	INV 2nd IS+ †
ICOS	INV 2nd COS †
IDMS	INV 2nd DMS †
IDSZ	INV 2nd DSZ †
IFF	2nd IFF
IFIX	INV 2nd FIX †
IFF	INV 2nd IFF †
IINT	INV 2nd INT †
ILNX	INV 1nX †

Druck- symbol	Tastenfolge
ILOG	INV 2nd LOG †
IND	2nd Ind
INS	siehe Anmerkung unten
INT	2nd INT
INV	INV
IPD*	INV 2nd PRD 2nd Ind †
IP/R	INV 2nd P=R †
IPRD	INV 2nd PRD †
ISBR	INV SBR †
ISIN	INV 2nd SIN †
ISM*	INV SUM 2nd Ind †
ISTF	INV 2nd STF †
ISUM	INV SUM †
ITAN	INV 2nd TAN †
IX	INV 2nd X †
IXI	2nd IXI
IY*	INV Y* †
LBL	2nd LBL
LNx	1nX
LOG	2nd LOG
LRN	siehe Anmerkung unten
LST	2nd List
NOP	2nd Nop
OP	2nd Op
OP*	2nd Op 2nd Ind
PAU	2nd Pause
PD*	2nd PRD 2nd Ind
PG*	2nd Pgm 2nd Ind
PGM	2nd Pgm
P/R	2nd P=R
PRD	2nd PRD
PRT	2nd Prt
RAD	2nd Rad
RC*	RCL 2nd Ind

Druck- symbol	Tastenfolge
RCL	RCL
R/S	R/S
RST	RST
RTN	INV SBR
SBR	SBR
SIN	2nd SIN
SM*	SUM 2nd Ind
SST	siehe Anmerkung unten
ST*	STO 2nd Ind
STF	2nd STF
STO	STO
SUM	SUM
TAN	2nd TAN
WRT	2nd Wrt
X \approx T	X\approxT
X ²	X²
X	2nd X
IXI	2nd IXI
1/X	1/X
\sqrt{x}	\sqrt{x}
Y*	Y*

SYMBOLE

$\Sigma+$
 π
)
(
-
+
 \times
+
=
.
+/-

ANMERKUNG: Dieser Befehl ist nur sichtbar, wenn die Taste während der Auflistung eines Programms vorkommt. Da der Tastenkod nicht in den Programmspeicher eingebracht werden kann, wenn man diese Taste drückt, kann der Tastenkod nur ein Rest aus dem Redigieren eines anderen Befehls sein, und muß korrigiert werden.

Nur TI-58C: Das Symbol Δ wird während der "Trace"-Operation ausgedruckt, wenn eine Verzweigung stattfindet.

† nur bei Protokollierung.

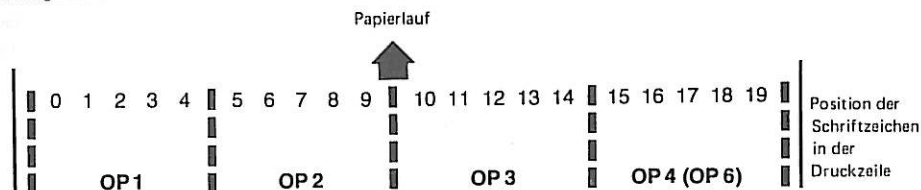


SPEZIELLE STEUEROPTIONEN ZUM DRUCKEN

Die Steueroptionen 00 bis 08 sind ausschließlich für die Anwendung zusammen mit dem Drucker bestimmt.

Alphanumerisches Drucken — **Op** 00 bis 06

Mit den ersten sieben Steueroptionen können alphanumerische Nachrichten erzeugt und ausgedruckt werden. In jede Druckzeile passen 20 Schriftzeichen. Sie werden in Gruppen von je 5 Schriftzeichen zusammengestellt und gespeichert, wie unten dargestellt.



Jedes gedruckte Schriftzeichen wird durch einen zwei-stelligen Kode, entsprechend der folgenden Tabelle dargestellt. Der Kode wird aus der Reihe-Spalte-Adresse hergeleitet.

	Einerstelle							
	0	1	2	3	4	5	6	7
Zehnerstelle	0		0	1	2	3	4	5
	1		7	8	9	A	B	C
	2		-	F	G	H	I	J
	3		M	N	O	P	Q	R
	4		.	U	V	W	X	Y
	5		x	+	-	/	=	<
	6		↑	↓	↖	↗	×	÷
	7		≡	?	÷	°	II	III

A ist zum Beispiel Kode 13 und + ist Kode 47. Die Kodes für 5 Schriftzeichen (10 Stellen) können gleichzeitig in die Anzeige eingegeben werden. Wenn Sie nicht alle 10 möglichen Stellen genau spezifizieren, werden vor den eingegebenen Stellen automatisch Nullen angenommen. (Ein Null-Paar ist der Kode für eine Leerstelle.) Um nach einigen Schriftzeichen Leerstellen zu bekommen, geben Sie einfach nach den Kodes für die Schriftzeichen Null-Paare ein.



Sobald die Anzeige eine Reihe von Kodes für die Schriftzeichen enthält, drückt man **[2nd] [Op] 01, 02, 03 oder 04**, um den Rechner exakt anzuweisen, an welcher Stelle der Zeile die Schriftzeichen zu drucken sind.

- [2nd] [Op] 01** — für das äußere linke Viertel der Zeile
- [2nd] [Op] 02** — für das innere linke Viertel der Zeile
- [2nd] [Op] 03** — für das innere rechte Viertel der Zeile
- [2nd] [Op] 04** — für das äußere rechte Viertel der Zeile

Mit **[2nd] [Op] 00** löscht man das Druckerregister. **[2nd] [Op] 05** ist die Anweisung, den Inhalt des Druckerregisters auszu-drucken. Die Beschriftung kann zur Kennzeichnung von Teilen des Papierbandes im Rechenmodus oder auch in einem Pro-gramm verwendet werden.

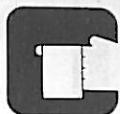
Beispiel: Geben Sie einem Papierstreifen die Überschrift " π^2 VS X% TESTS 3/22"

Symbol	π^2	VS	X%	TESTS	3/22
Kode	53 70 00 42 36	00 44 61 00 37	17 36 37 36 00	04 63 03 03	

Taste	Anzeige	Bemerkungen
[CLR] [2nd] [Op] 00	0.	Löschen des Druckerregisters
5370004236 [2nd] [Op] 01	5370004236.	Speicherung von " π^2 vs", für den Druck im äußeren linken Viertel
44610037 [2nd] [Op] 02	44610037.	Speicherung von "X% T", für den Druck im inneren linken Viertel
1736373600 [2nd] [Op] 03	1736373600.	Speicherung von "ESTS", für den Druck im inneren rechten Viertel
463030300 [2nd] [Op] 04	463030300.	Speicherung von "3/22", für den Druck im äußeren rechten Viertel
[2nd] [Op] 05	463030300.	Ausdruck der kompletten Beschriftung

Beachten Sie, daß für das innere linke Viertel der Druckzeile zunächst eine Leerstelle nötig ist. Nicht-ausgetauschte führende Nullen im Druckerregister erzeugen diese Leerstelle. Im äußeren rechten Viertel ist 04 der erste erforderliche Schriftzeichen-kode, aber nur 4 muß eingegeben werden. Die Druckzeilen-Viertel können in beliebiger Reihenfolge eingegeben und durch einen anderen Beschriftungssatz überschrieben werden. Die Register, die zur Speicherung dieser Beschriftung belegt werden, sind sonst für die 5-te, 6-te, 7-te und 8-te unvollständige Operation reserviert. Wenn Sie also eine Beschriftung zusammen-stellen, dürfen nicht mehr als 4 Operationen unvollständig belassen werden.

Beachten Sie, daß alle Sonderformen der Anzeige, also Festkomma-Einstellung, Exponentialform und technische Notation aufgehoben sind, ehe Sie alphanumerische Nachrichten eingeben. Beachten Sie auch, daß beim TI-59 die Befehle **[2nd] [Op] 00** bis **[2nd] [Op] 05** den Bruchteil der Zahl im Anzeigeregister auf die gleiche Weise löschen wie die Ganzzahl-taste. Nur **[2nd] [Op] 01** bis **[2nd] [Op] 04** löschen den Bruchteil der angezeigten Zahl beim TI-58C.



Eine weitere spezielle Steueroperation ist **2nd Op 06**: Damit wird der derzeitige Wert in der Anzeige zusammen mit den 4 Schriftzeichen im äußeren rechten Viertel der Zeile ausgedruckt — eine vorteilhafte Möglichkeit, Programmsergebnisse zu kennzeichnen.

Beispiel: Erstellen Sie ein Programm zur Berechnung und Kennzeichnung des π -Näherungswertes $22/7$.

Speicherplatz
und Tastenkod

000 03
001 03
002 02
003 04
004 69
005 04
006 02
007 02
008 55
009 07
010 95
011 69
012 06
013 91

Tastenfolge

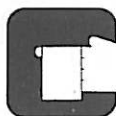
3 } P
3 }
2 } I
4 }
2nd Op
0 4
2
2
÷
7
=
2nd Op
6
R/S

Speicherung von PI für den Druck

$22 \div 7$

Druck

Wenn das Programm abläuft, erzeugt der Drucker die Zeile 3.142857143 Pi.



Aufzeichnen von Daten (Plotten) — Op 07

Die Steueroperation 07 zeichnet den derzeitigen Anzeigewert (0 bis 19) in der Schriftzeichenposition 0 bis 19 auf (Beginn mit 0 in der Folge von links nach rechts). Die Operation, primär für die Anwendung in Programmen bestimmt, ermöglicht die Aufzeichnung von Kurven und Histogrammen. Pro Zeile wird ein Sternchen (*) gedruckt. Der ganzzahlige Teil des aufzuzeichnenden X-Wertes muß im Bereich $0 \leq X < 20$ liegen. Liegt der Anzeigewert nicht in diesem Bereich, wird der Wert nicht aufgezeichnet und die Anzeige blinkt, wenn das Programm unterbricht. Nur der ganzzahlige Teil wird aufgezeichnet.

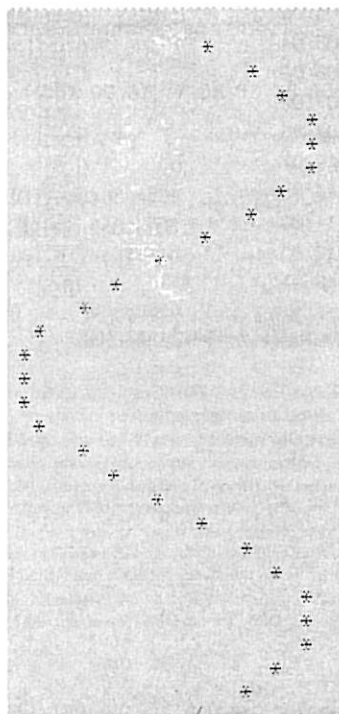
Beispiel: Erstellen Sie ein Programm zur Aufzeichnung einer Sinus-Kurve, wobei die Werte alle 18° gebildet werden.

Tastenfolge

2nd [Lb] [A]
[RCL]
[1]
2nd sin
+
[1]
[)]
[X]
[9]
[.]
[9]
[=]
2nd Op
[0] [7]
[1]
[8]
[SUM]
[1]
[A]

Resultat

0 1 3 5 7 9 11 13 15 17 19 Werte



Beachten Sie, daß zu allen Sinuswerten in den Speicherplätzen 004, 005 die Zahl 1 addiert wird, und daß damit der Sinus von allen Winkeln positiv ist. Die Werte können jetzt im Bereich 0 bis 2 liegen. Wenn die Werte maßstäblich um 9.9 vergrößert werden, erweitert sich der Bereich auf 0 bis 19.8, ideal für Aufzeichnungszwecke. Das Programm läuft in der Form ab, daß zunächst ein Anfangswinkel in Register 01 gespeichert wird, und dann drückt man die Taste [A]. Das Programm läuft, bis Sie mit [R S] unterbrechen.

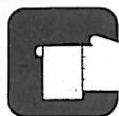


Auflisten der im Programm verwendeten Labels — **Op 08**

Mit der Tastenfolge **2nd** **Op 08** erhält man eine fortlaufende Auflistung aller Labels und der Speicherplätze, die sie im Programmspeicher belegen. Die Auflistung beginnt mit der augenblicklichen Position des Programmzeigers. Um also alle Labels zu erfassen, drückt man vor **2nd** **Op 08** zunächst **GTO 0** oder **RST**. Siehe das Beispiel einer Auflistung unten:

Speicher- Tasten- Kontr.-
platz kode Symbole

001	11	A
018	12	B
062	19	D'
129	13	C
205	88	DMS
239	70	RAD
273	80	GRD
282	14	D
288	15	E
294	16	A'
300	10	E'
306	28	LOG



DRUCKKOPFREINIGUNG

Das Verfahren für die Reinigung des Druckkopfes ist in der Bedienungsanleitung PC-100/A/B/C ausführlich beschrieben. Das folgende Programm muß für die programmierbaren Rechner T1-58, T1-58C und T1-59 verwendet werden.

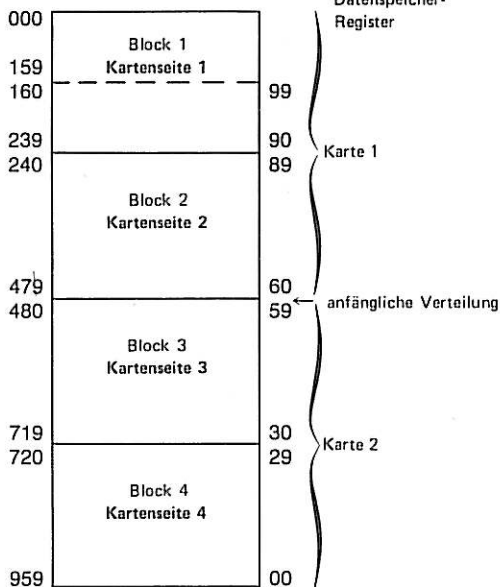
Speicherplatz und Tastenkod	Tastenfolge
000 04	4
001 42	STO
002 00	0 0
003 09	9
004 42	STO
005 06	6
006 52	EE
007 01	1
008 00	0
009 94	+/-
010 22	INV
011 52	EE
012 35	1/x
013 76	2nd Lbl
014 11	A
015 84	2nd Op 2nd Ind
016 00	0
017 97	2nd Dsz
018 00	0
019 11	A
020 76	2nd Lbl
021 12	B
022 69	2nd Op
023 05	5
024 97	2nd Dsz
025 06	6
026 12	B
027 91	R/S

Um diese Folge ablaufen zu lassen, drücken Sie **RST** **R S**. Wenn nötig, wiederholen Sie das Programm.



Auf den leeren Magnetkarten, die im Lieferumfang Ihres Rechners enthalten sind, können Sie jedes gespeicherte Programm oder gespeicherte Daten dauerhaft aufzeichnen. Wie Sie bereits wissen, ist der Rechner mit 120 Speicherregistern ausgestattet, die zwischen Programmspeicher und Datenspeicher verteilt werden können. Diese Register sind in vier Blöcke mit je 30 Registern aufgeteilt. Eine Karte kann zwei dieser Blöcke aufzeichnen, einen Block pro Seite.

Programmspeicher-
plätze



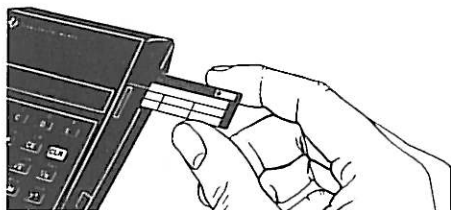
Speicherbereich

ANMERKUNG : Zum Lesen einer Magnetkarte löschen Sie die Anzeige mit [CLR] wählen wenn nötig Fließkomma mit [2nd] [Fix] 9 und schieben die Karte in den unteren Schlitz auf der rechten Seite des Rechners. Der Kartenleser schaltet sich automatisch ein und liest die Karte. Siehe Seite VII-5 für weitere Einzelheiten.



AUFZEICHNUNG (SCHREIBEN) AUF MAGNETKARTEN

Die Aufzeichnung auf Magnetkarten erfolgt über die Tasten **[2nd] [Write]**. Um den Inhalt von Block n (n = 1, 2, 3 oder 4) auf die Kartenseite n zu schreiben, drückt man n **[2nd] [Write]**, und schiebt die Karte mit der bedruckten Seite nach oben in den unteren Schlitz an der rechten Seite des Rechners. Beachten Sie, daß Fix 0 die einzige Festkomma-Einstellung ist, bei der man eine Magnetkarte beschreiben kann. Wenn Sie bezüglich des Anzeigeformats unsicher sind, drücken Sie vor Aufzeichnung der Tastenfolge **[INV] [2nd] [Fix]**. Fließkomma ist erlaubt.



Der Inhalt des Registerblocks wird damit auf die Magnetkarte übertragen, gleich, ob der Block mit Programminformationen oder mit Daten oder mit beidem belegt ist. Die Kartenseite, die Sie beschreiben, trägt die Blocknummer, in der die Informationen gespeichert sind. Wenn **[2nd] [Write]** gedrückt wird, berücksichtigt der Rechner nur den ganzzahligen Teil der ausgewiesenen Zahl, und Bruchteile werden ignoriert.

2.31 **[2nd] [Write]** zum Beispiel überträgt den Inhalt von Block 2 auf eine Kartenseite, die mit 2 gekennzeichnet wird. Bei jeder Zahl für n, die kleiner als 1 oder größer oder gleich 5 ist, blinkt die Anzeige, und die Aufzeichnung findet nicht statt. Achten Sie also darauf, daß der Wert 1, 2, 3 oder 4 in der Anzeige ist, wenn Sie eine Magnetkarte beschreiben.

Wenn Sie eine Karte in den Rechnermechanismus einschieben, darf sie nicht mehr behindert oder festgehalten werden, sobald sie vom Transportsystem erfaßt ist. Bis zum Abschluß der Aufzeichnung bleibt die Anzeige dunkel, anschließend wird die Nummer des aufgezeichneten Blocks ausgewiesen. Wenn nach dem Beschreiben die Blocknummer blinkt, löschen Sie die Anzeige und wiederholen den Aufzeichnungsvorgang. Blinkt die Anzeige dann immer noch, kann die Magnetkarte schadhafte sein. Versuchen Sie eine andere Karte.

Bei der Aufzeichnung von Daten anstelle eines Programms ist zu bedenken, daß das Register 00 in Block 4 ist, und die fortlaufende Numerierung der Datenregister geht auf Block 3 über, etc.

1	◀	▶	TEXAS INSTRUMENTS		▶	2
Produktverteilung (Sojabohnen)					239,89	
	Grenze					
Bohnen	Mehl	Öl	Abfall	Anfang		

Kennzeichnen Sie eine Magnetkarte immer entsprechend der auf ihr gespeicherten Informationen. In den oberen Ecken der Karte ist Platz für die Angabe der Blocknummern, die auf dieser Karte aufgezeichnet sind. Der Pfeil in diesen Feldern zeigt an, in welcher Richtung die Karte in den Rechner bei der Aufzeichnung des angegebenen Blocks eingeschoben wurde. Der Raum über der Kartenmitte ist frei für den Programmtitel und für andere relevante Informationen wie die erforderliche Speicherbereichsverteilung. Darunter sind zwei Reihen mit Kästchen. Die unteren fünf Kästchen kann man verwenden, um die Funktion der Programmadress-Tasten **[A]** bis **[E]** im aufgezeichneten Programm zu markieren. Die obere Kästchenreihe kann in gleicher Weise für die Tasten **[2nd] [A]** bis **[2nd] [E]** genutzt werden.

Wenn ein Registerblock aufgezeichnet wird, ist zu beachten, daß auch die Nummer des Blocks und die augenblickliche Speicherbereichsverteilung magnetisch auf der Karte gespeichert werden.



Aufzeichnung auf Magnetkarten

Anzeige, wenn [2nd] [Write] gedrückt und die Karte ein- geschoben wird	Normales Programm	Geschütztes Programm
1, 2, 3, 4	Beschreiben einer Kartenseite mit den Informationen des Blocks (Programm und/oder Daten) einschl. der angegebenen Blocknummer und der augenblicklichen Speicherbereichsverteilung.	Wenn der Block nur Programminformationen enthält, läuft die Karte durch, wird aber nicht beschrieben – die Nummer in der Anzeige blinkt. Wenn der Block auch Daten enthält, wird der Block mit negativer Blocknummer zwar aufgezeichnet, aber nicht geschützt.
-1, -2, -3, -4	Schützen und Beschreiben der Kartenseite mit den Informationen des Blocks incl. der angegebenen Blocknummer und der augenblicklichen Speicherbereichsverteilung.	Wenn der Block nur Programminformationen enthält, läuft die Karte durch, wird aber nicht beschrieben – die Nummer in der Anzeige blinkt. Wenn der Block auch Daten enthält, wird der Block mit negativer Blocknummer aufgezeichnet.
Eine beliebige andere Zahl	Die Karte läuft durch, wird aber nicht beschrieben. Rechts in der Anzeige blinken zwei Ziffern.	Wie beim normalen Programm.

Wenn beim Versuch, eine Karte zu beschreiben, ein Wert in der Anzeige blinkt, läuft die Karte zwar durch, die Aufzeichnung findet nicht statt und die beiden rechten Ziffern in der Anzeige blinken.

Die Anzeige des Rechners darf beim Aufzeichnen auf Magnetkarte nicht auf Festkomma eingestellt sein.

Jedesmal, wenn die Anzeige blinkt, läuft der Transportmechanismus weiter, bis die Karte herausgezogen wird.

Das Aufzeichnen kann auch programmiert gesteuert werden. Wenn der Programmzeiger den Befehl n [2nd] [Write] erreicht, erwartet der Rechner eine Magnetkarte zum Aufzeichnen. Sobald das geschehen ist, wird das Programm beim nachfolgenden Befehl fortgesetzt.

Diese Maßnahme ist z.B. dann vorteilhaft, wenn errechnete Zwischenwerte aufgezeichnet werden sollen.

Bitte achten Sie auf den Unterschied zwischen Magnet- und Labelkarten. Die Magnetkarten sind durch die Pfeile in den Eckfeldern zu erkennen.



DATENSCHUTZ FÜR PROGRAMME

Wenn ein Programm, das auf Magnetkarte geschrieben werden soll, Informationen enthält, die nicht bekannt werden dürfen, können Sie diese Informationen schützen: Sie geben vor dem Beschreiben Ihrer Karte eine negative Blocknummer ein. Die Folge n [+/-] **2nd** **Write** beschreibt eine Karte, deren Programm dann nur wieder in den Programmspeicher eingelesen und durchgeführt werden kann. Wenn das Programm wieder in den Rechner gelesen ist, gelten für die Anwendung folgende Einschränkungen:

- Keine Auflistung oder Protokollierung mit einem Drucker
- Keine Auflistung der Labels
- Keine Redigierung, keine Änderung der Speicherbereichsverteilung, keine erneute Aufzeichnung des Programms
- Keine Durchführung in Einzelschritten, kein Festhalten der Pausentaste während des Ablaufs

Sie können den Rechner nicht verlassen, eine geschützte Kartenseite in den falschen Speicherblock einzulesen. Wenn eine Seite der geschützten Programmkarte in den Programmspeicher eingelesen wird, wird ein internes Programmschutz-Flag gesetzt, das die oben genannten Einschränkungen veranlaßt. Die einzige Möglichkeit, diese Einschränkungen zu beseitigen (das Flag rückzusetzen) besteht darin, die Tasten **2nd** **CP** zu drücken oder den Rechner auszuschalten. Ein geschütztes Programm im Programmspeicher kann überschrieben werden, das Schutzflag wird dabei aber nicht rückgesetzt. Das neue Programm ist jetzt wie das vorige geschützt.

Die Inhalte der Datenregister können nicht geschützt werden. Ebenso wenig ist der Datenschutz für einen Block möglich, der durch die Speicherbereichsverteilung irgendwo in der Mitte abgeteilt ist, denn wenn ein Block Datenregister enthält, befinden sie sich am Anfang des Blocks. Aus diesem Grund betrachtet der Rechner den gesamten Block als Datenspeicherbereich.

Ein ungeschütztes Programm kann auf eine Karte übertragen werden, auf der zuvor ein geschütztes Programm gespeichert war, wenn man nach dem normalen Aufzeichnungsverfahren vorgeht. Das neue Programm auf der Karte ist dann nicht geschützt.



LESEN VON MAGNETKARTEN

Das Transportsystem des Rechners zieht eine Magnetkarte automatisch durch den Rechner, wenn sie im Rechenmodus in den Kartenschlitz eingeschoben wird. Ob die Karte gelesen wird oder nicht, hängt ab vom Inhalt des Anzeigeregisters und von der Speicherbereichsverteilung:

1. Mit Null in der Anzeige kann jeder Block gelesen werden, vorausgesetzt, daß dieser Block bei der augenblicklich gewählten Verteilung im Programmspeicher eingeschlossen ist, und diese der Verteilung beim Aufzeichnen entspricht. Ist die Verteilung falsch, wird die Karte nicht gelesen, und in der Anzeige blinkt die Nummer des Blocks, die auf der eingegebenen Karte aufgezeichnet ist.
2. Mit n in der Anzeige kann der Rechner nur Block n lesen. Wenn auf der Karte eine andere Blocknummer gespeichert ist, blinkt die gelesene Nummer in der Anzeige, und die Karte wird nicht gelesen. Bei falsch gewählter Verteilung blinkt wiederum die Nummer des eingegebenen Blocks in der Anzeige und die Karte wird nicht gelesen.
3. Mit $-n$ in der Anzeige wird jeder Block, der in den Rechner eingelesen wird, im Block n aufgenommen. Wenn eine Karte eingelesen ist, wird die Blocknummer zur Anzeige gebracht, die magnetisch auf der Karte gespeichert ist. Eine geschützte Karte (negative Blocknummer) kann nur in den Block eingebracht werden, der auf der Karte angegeben ist.

Wenn nach Eingabe einer Karte in der Anzeige eine Null blinkt, hat der Rechner einen Lesefehler erkannt, und der Lesevorgang muß wiederholt werden.

Wenn bei Eingabe einer Karte eine andere Zahl als $\pm n$ ($n = 1, 2, 3$ oder 4) in der Anzeige ist, wird die Karte nicht gelesen, und die beiden rechten Ziffern der Anzeige blinken.

PROGRAMMIERTER LeseBEFEHL

Während ein Programm abläuft, weist der Befehl **INV** **2nd** **Write** den Rechner an, in Übereinstimmung mit den obigen Regeln eine Magnetkarte zu lesen. (Die entsprechende Blocknummer oder der Befehl **CLR** muß im Programm vor der Tastenfolge **INV** **2nd** **Write** angegeben sein). Die Karte wird in den Kartenleser des Rechners eingeschoben, aber erst gelesen, wenn der Befehl **INV** **2nd** **Write** im Programm vorkommt. Damit können bei Bedarf Daten zugeführt werden.

Beachten Sie: Eine Karte, die in den Kartenleser eingeschoben ist, wird automatisch gelesen, wenn ein Programm als Folge von **INV** **SBR** oder **R/S** unterbrochen wird.



Lesen von Magnetkarten

Anzeige bei Eingabe der Karte	Karte mit normalem Programm	Karte mit geschütztem Programm
0	Einlesen der Information in den Block, dessen Nummer auf der Karte gespeichert ist, vorausgesetzt, die augenblicklich gewählte Verteilung entspricht der Verteilung der Karte. Ist die Verteilung falsch, läuft die Karte durch, wird aber nicht gelesen — in der Anzeige blinkt die Nummer der durchgelaufenen Kartenseite.	Wie beim normalen Programm
1, 2, 3, 4	Einlesen der Kartenseite mit der angegebenen Blocknummer — die Nummer der Kartenseite wird angezeigt. Bei Eingabe einer anderen Seite oder bei falscher Verteilung läuft die Karte durch, wird aber nicht gelesen — in der Anzeige blinkt die Nummer der durchgelaufenen Kartenseite.	Wenn die Nummer der durchlaufenden Kartenseite mit der Anzeige identisch ist, wird die Karte gelesen und die Nummer als negative Zahl ausgewiesen. Bei Eingabe einer anderen Kartenseite oder bei falscher Verteilung läuft die Karte durch, wird aber nicht gelesen — in der Anzeige blinkt die Nummer der durchgelaufenen Kartenseite.
-1, -2, -3, -4	Die gelesene Kartenseite wird zwangsweise in den Block mit der entsprechenden Nummer eingebracht — ohne Rücksicht auf Datenschutz oder Verteilung. Ein geschütztes Programm kann nicht in einen beliebigen Block eingelesen werden.	Einlesen der Kartenseite mit der angegebenen Blocknummer — Anzeige der Nummer der Kartenseite. Bei Eingabe einer anderen Seite oder bei falscher Verteilung läuft die Karte durch, wird aber nicht gelesen — in der Anzeige blinkt die Nummer der durchgelaufenen Kartenseite.
Eine andere Zahl	Die Karte läuft durch, wird aber nicht gelesen — in der Anzeige blinken die beiden rechten Ziffern.	Wie beim normalen Programm

Wenn beim Versuch, eine Karte einzulesen, ein Wert in der Anzeige blinkt, läuft die Karte durch, wird aber nicht gelesen. Die beiden rechten ganzen Zahlen blinken.

Die Anzeige des Rechners darf beim Lesen von Magnetkarten nicht auf Festkomma eingestellt sein.

Achten Sie darauf, daß das Batteriepaket voll aufgeladen oder an das Wechselstromnetz angeschlossen ist, ehe Sie mit langen Berechnungen beginnen. Diese Vorsichtsmaßnahme hat entscheidende Wirkung auf die Genauigkeit beim Lesen und Beschreiben von Magnetkarten.

Eine blinkende 0 in der Anzeige weist darauf hin, daß die Karte nicht richtig gelesen wurde und noch einmal eingegeben werden sollte.



PFLEGE DER MAGNETKARTEN

ACHTUNG: Magnetkarten mit aufgezeichneten Informationen können beschädigt oder geändert werden, wenn man sie Staub oder Fremdkörpern, Dauermagneten oder elektromagnetischen Feldern (wie Elektromotoren, Leistungstrafos etc.) aussetzt.

Magnetkarten haben die Fähigkeit, die gespeicherten Informationen unbegrenzt lange zu erhalten. Die aufgezeichnete Information unterliegt keinen Alterungsprozessen, und kann nur durch ein externes Magnetfeld geändert werden. Während sich die magnetischen Signale keineswegs verschlechtern, sind bei den physikalischen Eigenschaften der Karte und beim Kartentransportsystem im Rechner Schäden möglich.

Die Behandlung der Magnetkarten

Die sorgfältige Behandlung der Magnetkarten ist sehr wichtig. Eine beschädigte, geknickte oder verkratzte Karte kann für den beabsichtigten Zweck wertlos sein. Die schlechte Qualität einer Magnetkarte ist in aller Regel auf eine Serie von Mißgeschicken oder auf unsachgemäße Behandlung zurückzuführen.

Eine ganze Reihe von Verunreinigungen muß vermieden werden. Schützen Sie die Karten vor allem gegen die häufigsten Verschmutzungen wie Zigarettenasche, Speisereste, Staub und ölige Flüssigkeiten. Eine Karte kann schon verschmutzt werden, wenn man sie auf einer verunreinigten Fläche ablegt oder wenn man mit den Händen Schmutzpartikel auf die Karte überträgt. Selbst das Hautfett haftet auf den Karten und begünstigt die Ansammlung von Staub und Fremdkörpern. Beachten Sie, daß mit einer einzigen verschmutzten Karte nicht nur der Lesemechanismus, sondern auch andere, später verwendete Karten verunreinigt werden können. Extreme Verschmutzungen mit öligen Stoffen können zum Ausfall des Lesemechanismus führen und der Rechner muß dann in einer der Texas Instruments Service-Abteilungen repariert werden. Die folgenden einfachen Anweisungen garantieren maximale Betriebsdauer der Magnetkarten:

1. Karte möglichst nur an den Ecken anfassen.
2. Karten von Magneten und scharfen Gegenständen fernhalten, die den Oxidbelag beschädigen könnten.
3. Nicht benutzte Karten in der PVC-Tragtasche oder in einem anderen geeigneten Behälter aufbewahren.
4. Verschmutzte Karten sofort reinigen.
5. Versuchen Sie nie, deutlich beschädigte oder verunreinigte Karten einzulesen oder zu beschreiben.



Reinigung der Magnetkarten

Eine verschmutzte Karte kann leicht ohne spezielle Reinigungs- oder Lösungsmittel gereinigt werden. Verwenden Sie unter keinen Umständen Flüssigkeiten auf Petroleumbasis, wie Feuerzeugbenzin, um die Magnetkarten zu reinigen. Staub und Fremdpartikel lassen sich von der Karte mit einem weichen trockenen Tuch entfernen. Andere Verschmutzungen können mit warmem Wasser, dem etwas mildes flüssiges Reinigungsmittel beigemischt ist, abgewaschen werden. Danach spült man die Karte ab und trocknet sie mit einem weichen Tuch.

Beschriftung der Magnetkarten

Die mitgelieferten leeren Magnetkarten haben Beschriftungsfelder für Zahlen, Symbole und Kurztitel Ihrer eigenen Programme. Sie können die Karten mit einem weichen Bleistift oder mit einem Filzstift mit abwaschbarer Tinte beschriften, wenn die Notizen später wieder entfernt werden sollen. Wenn Sie einen Filzstift mit wasserfester Tinte verwenden, wird die Karte natürlich dauerhaft beschriftet. Verlangen Sie in Ihrem Schreibwarengeschäft am besten Filzstifte, die für die Diaschriftung benutzt werden. Die meisten Händler führen eine Vielzahl von Farben mit abwaschbarer oder wasserfester Tinte.

ANWENDUNG DER REINIGUNGSKARTE FÜR DIE ANTRIEBSROLLE

Die Reinigungskarte für die Antriebsrolle verwendet man jeweils nach etwa 500 Lesevorgängen, oder wenn die Karte nicht mehr gleichmäßig durch den Rechner gezogen wird. Drücken Sie **1** **2nd** **Write** und schieben Sie die Karte ein. Halten Sie die Karte an ihrer Schmalseite fest und schieben Sie sie bei laufendem Transportmechanismus vor und zurück. Drei oder vier Sekunden genügen zur Reinigung der Rolle. Ziehen Sie die Karte zurück und drücken Sie **R/S**. Wenn das Transportsystem nach Herausnehmen der Karte weiterläuft, schalten Sie den Rechner kurz aus und wieder ein.



ANWENDUNG DER MAGNETKOPF-REINIGUNGSKARTE

Die mitgelieferte, besonders gekennzeichnete Magnetkopf-Reinigungskarte hat statt der üblichen Oxidbeschichtung eine Schleifschicht. Mit dieser Karte werden aufgestautes Oxid oder Fremdpartikel vom magnetischen Lese-/Schreibkopf im Rechner entfernt. Die Karte darf nicht generell bei jeder Störung benutzt werden, da zu häufige Anwendung die Eigenschaften des Lese-/Schreibkopfes ändern kann. Der Anhang A enthält im Kapitel ABHILFE BEI STÖRUNGEN Anweisungen, wie die Reinigungskarte für die Beseitigung von Schwierigkeiten sinnvoll eingesetzt werden kann. Die Reinigungskarte wird wie eine normale Magnetkarte in den unteren Schlitz des Rechners geschoben, und dann vom Transportsystem durch den Rechner gezogen. Drücken Sie **[CLR]**, wenn nach Durchlauf der Karte die Anzeige blinkt. Die Magnetkopfreinigungskarte darf nur äußerst sparsam und nur einmal pro Störung benutzt werden. Achten Sie darauf, daß die Karte bei Anwendung sauber ist.

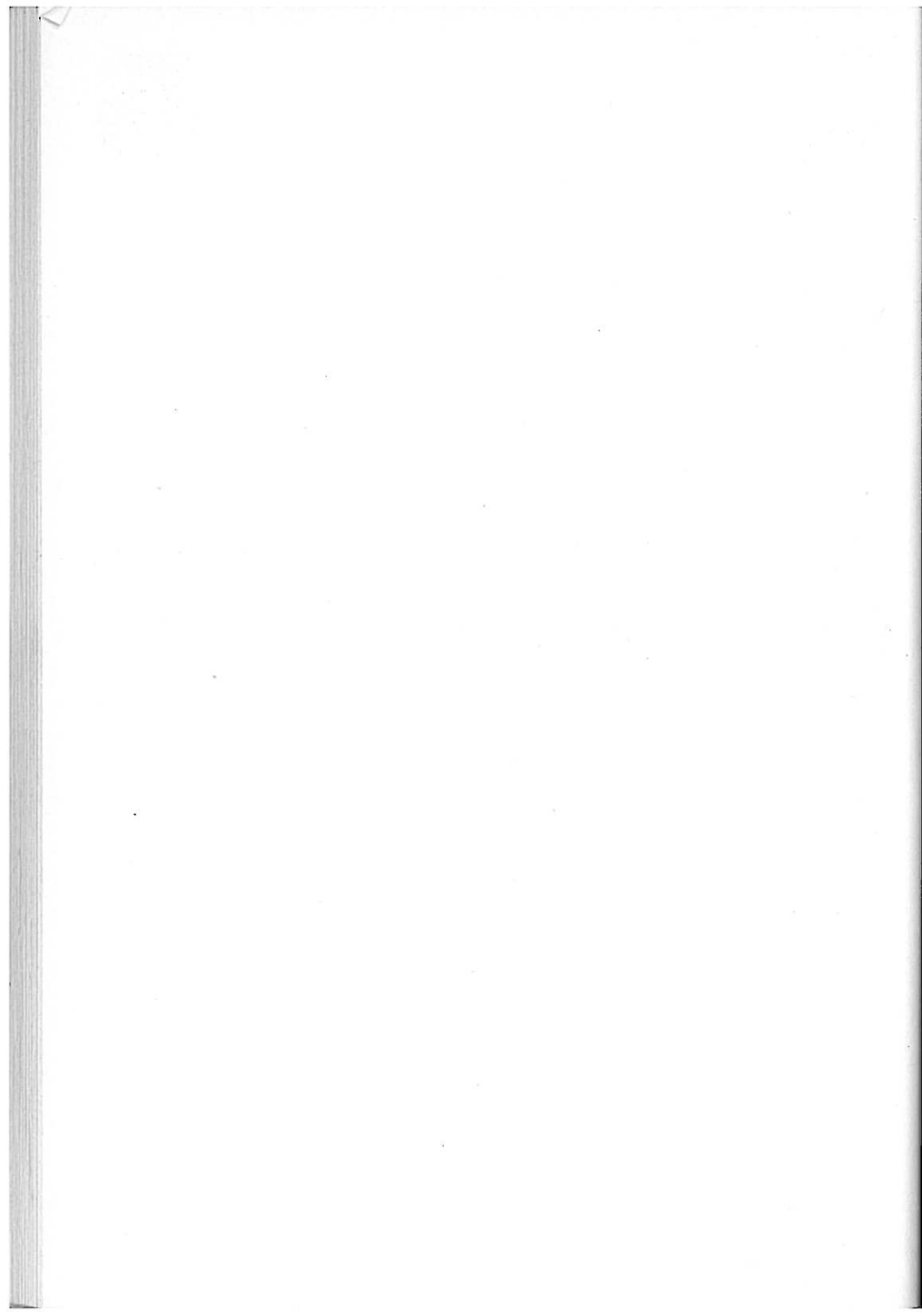
LESE/SCHREIBTEST.

Zum Testen der Lese/Schreibeinheit dient der folgende Arbeitsablauf : das Diagnose-Programm des Standardmoduls wird in den Arbeitsspeicher des Rechners geladen und auf eine Magnetkarte geschrieben. Dann wird der Rechner aus- und wieder eingeschaltet, die beschriebene Karte eingelesen und der Diagnose-test durchgeführt.

Dazu ist folgende Tastenfolge notwendig :

[2nd] **[PgM]** 01 **[2nd]** **[Op]** 09 1 **[2nd]** **[Write]** . Einschreiben Der Magnetkarte. 1 wird angezeigt. Aus- und Einschalten des Rechners. Einlesen der Magnetkarte. 1 wird angezeigt. Dann Drücken der Tasten **[SBR]** **[=]** . Nach einigen Sekunden erscheint 1 in der Anzeige.

Bei Auftreten einer Störung im Arbeitsablauf siehe Abschnitt A.



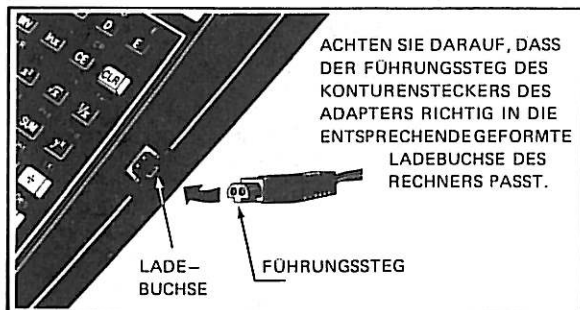


WARTUNGS- UND SERVICE-INFORMATIONEN

BATTERIE- UND NETZBETRIEB

Normalbetrieb – Um mit dem Rechner eine maximale Zeitspanne netzunabhängig arbeiten zu können, schließen Sie das AC 9900 H Adapter/Ladegerät an eine 220V 50Hz Steckdose an, verbinden damit den Rechner, und laden das Batteriepaket mindestens 4 Stunden bei ausgeschaltetem oder 10 Stunden bei eingeschaltetem Rechner. Das Adapter/Ladegerät und das Batteriepaket können im Netzbetrieb warm werden. Dies ist jedoch normal und hat keine weiteren Auswirkungen.

Vorsicht: Der Rechner kann beschädigt werden, wenn das Netzadapter/Ladegerät angeschlossen, aber das Batteriepaket nicht eingesetzt ist.



Bei voll aufgeladenem Batteriepaket arbeitet der Rechner etwa 2 bis 3 Stunden, bis ein erneutes Aufladen erforderlich ist. Zögern Sie jedoch nicht, das Adapter/Ladegerät anzuschließen, wenn Sie vermuten, daß das Batteriepaket fast entladen ist. Ein nahezu entladenes Batteriepaket kann alle Rechenoperationen negativ beeinflussen. Typische Zeichen für ein entladenes Batteriepaket sind eine schwache, unkontrolliert aufleuchtende oder völlig dunkle Anzeige, oder das Magnetkarten-Transportsystem beginnt zu laufen. Wenn die vollständige Entladung während eines Lesevorgangs eintritt, können die Programminformationen auf der Karte gelöscht oder geändert werden.

Wenn Sie beim TI-58C die Anzeichen eines entladenen Batteriepakets beobachten, verhindert ein sofortiges Ausschalten des Rechners daß Programm- und Datenspeicher-Inhalte verloren gehen.

Setzen Sie ein neues Batteriepaket ein oder schließen Sie das Ladegerät so schnell wie möglich an. Unterbrechen Sie immer ein laufendes Programm, trennen Sie das Netzadapter/Ladegerät ab und schalten Sie den Rechner aus, ehe Sie das Batteriepaket entfernen.

Die Lebensdauer der Nickel-Cadmium-Zellen des Batteriepakets ist zwar schwer vorauszusagen, aber bei normalem Gebrauch halten die Batterien etwa 2 bis drei Jahre oder 500 bis 1000 Ladezyklen.

Periodisches Aufladen – Obwohl der Rechner unbegrenzt mit angeschlossenem Netzadapter/Ladegerät arbeitet, kann das aufladbare Batteriepaket mit der Zeit seine Speicherkapazität verlieren, wenn es nicht gelegentlich entladen wird. Im Interesse einer langen Lebensdauer der Batterien empfiehlt es sich, den Rechner mindestens zweimal im Monat netzunabhängig zu betreiben, die Batterien dabei fast völlig zu entladen und dann entsprechend wieder aufzuladen.

Tief-Entladen der Batterien – Bleibt der Rechner längere Zeit eingeschaltet, wenn das Batteriepaket bereits entladen ist (etwa zufällig über Nacht), schließen Sie das Adapter/Ladegerät mindestens 24 Stunden lang bei ausgeschaltetem Rechner an. Wenn danach normaler Batteriebetrieb nicht mehr möglich ist, muß das Batteriepaket ausgetauscht werden. Wiederholtes restloses Entladen zerstört auf die Dauer das Batteriepaket. Ersatz- und Austausch-BP-1A-Batteriepakete erhalten Sie bei Ihrem Händler, oder direkt von Texas Instruments.

Lagerung – Wenn der Rechner mehrere Wochen lang gelagert ist oder nicht benutzt wird, muß das Batteriepaket wahrscheinlich vor netzunabhängigem Betrieb geladen werden. Das Batteriepaket ist auslaufgeschützt; deshalb ist es ungefährlich, wenn der Rechner mit eingesetztem Batteriepaket aufbewahrt wird.

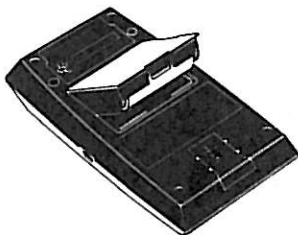


Austausch des Batteriepakketes — Das Batteriepaket läßt sich schnell und einfach ausbauen. Unterbrechen Sie immer ein laufendes Programm, trennen Sie das Netzadapter/Ladegerät ab und schalten Sie den Rechner aus, ehe Sie das Batteriepaket herausnehmen. Halten Sie den Rechner mit den Tasten nach unten. Stecken Sie eine kleine Münze in den Schlitz an der Unterseite des Rechners. Nach einer leichten Kippbewegung springt die Abdeckung auf und das Batteriepaket kann ganz herausgenommen werden.



Die außen angebrachten Metallkontakte am Batteriepaket sind die Anschlußklemmen. Achten Sie immer darauf, daß kein Metallgegenstand mit den Anschlußklemmen in Berührung kommt und die Batterien kurzschließt. Reinigen Sie die Anschlußklemmen von Zeit zu Zeit mit einem Radiergummi, um eventuell entstandene Korrosion zu beseitigen.

Beim Einsetzen des Batteriepakets müssen Sie den abgerundeten Teil des Pakets so in die Öffnung legen, daß der kleine Vorsprung an dem einen Ende unter die Kante des Rechnerbodens paßt. Das eingekerbte Ende des Pakets liegt dann dem Warnhinweis am nächsten. Mit geringem Druck rastet das Batteriepaket in die richtige Position.





ABHILFE BEI STÖRUNGEN

Wenn Schwierigkeiten auftreten, sollen Ihnen die folgenden Anweisungen bei der Analyse des Problems helfen. Sie können dann die Störung selbst beheben, ohne den Rechner an eine der Service-Abteilungen einzusenden. Sind die vorgeschlagenen Abhilfen unwirksam, wenden Sie sich schriftlich oder telefonisch an den Texas Instruments Kundendienst (siehe "Wenn Sie Fragen haben oder Unterstützung brauchen" später in diesem Anhang). Bitte geben Sie eine genaue Beschreibung der Fehlersymptome.

Tritt eine der folgenden Störungen beim Betrieb mit dem Drucker auf, nehmen Sie zuerst den Rechner vom Drucker ab und setzen das Batteriepaket wieder ein. Wenn mit dieser Maßnahme die Störung bereits beseitigt wurde, schlagen Sie in der Bedienungsanleitung für den Drucker nach.

Störung

1. In der Anzeige erscheinen unrichtige Ergebnisse, unsinnige Zahlen blinken, sie wird dunkel oder der Kartenleser läuft kontinuierlich.
2. Aus einem nicht ersichtlichen Grund bleibt die Anzeige dunkel.
3. Die Anzeige blinkt bei Tastaturoperationen.
4. Die Anzeige blinkt jedesmal, wenn ein Softwareprogramm aufgerufen wird.

Abhilfe

Wahrscheinlich ist das Batteriepaket entladen. Siehe Batterie- und Netzbetrieb zu Beginn dieses Anhangs.

Halten Sie die Taste **R/S** kurz gedrückt. Wenn jetzt wieder eine Anzeige erfolgt, verarbeitete der Rechner gerade ein langes Programm, war in einer Schleife oder erwartete das Einschleichen einer Magnetkarte.

Halten Sie **RST** kurze Zeit gedrückt. Wenn jetzt wieder eine Anzeige erfolgt, war die Verarbeitung gerade in einem Softwareprogramm, oder in einer Schleife (möglicherweise ist die Batterie fast entladen), oder es lief gerade ein langes Programm.

Das Batteriepaket kann entladen oder falsch eingesetzt sein.

Sie haben eine unzulässige Operation oder Tastenfolge eingegeben, oder die Kapazitätsgrenzen des Rechners wurden überschritten. Siehe die Auflistung dieser Bedingung in Anhang B.

Ein Softwareprogramm mit dieser Nummer existiert nicht. Siehe das entsprechende Handbuch für die Softwareprogramme.

Der Softwaremodul ist nicht richtig eingesetzt. Siehe Abschnitt III dieser Bedienungsanleitung.



Störung

5. Die Anzeige blinkt oder erzeugt falsche Resultate, wenn ein Softwareprogramm abläuft.

6. Die Anzeige blinkt oder erzeugt falsche Ergebnisse, wenn ein eigenes Programm im Programmspeicher abläuft.

Abhilfe

Es ist möglich, daß das falsche Programm aufgerufen wurde.

Falsche Programmbedienung. Siehe die entsprechenden Programm-Instruktionen im zugehörigen Handbuch.

Die Speicherbereichsverteilung ist auf zu wenige Datenregister eingestellt, um das Programm ablaufen zu lassen.

Drücken Sie **INV** **2nd** **Fix** oder schalten Sie den Rechner kurz aus und wieder ein und versuchen Sie das Programm erneut.

Drücken Sie **CLR** **2nd** **Pgm** **1** **SBR** **=** und lassen Sie das Software-Diagnoseprogramm ablaufen. Wenn die Anzeige blinkt, prüfen Sie, ob der Softwaremodul richtig eingesetzt ist (Abschnitt III) und tasten diese Folge erneut ein.

Beim Programmablauf gab es eine unzulässige Operation, Überlauf- oder Unterlaufbedingungen. Die Anhänge B, C und D erleichtern die Fehlersuche.

Eines der Softwareprogramme wurde aufgerufen. Drücken Sie **RST** und beginnen Sie erneut.

Drücken Sie **CLR** **2nd** **Pgm** **1** **SBR** **1**, um das Software-Diagnoseprogramm durchzuführen. Wenn die Anzeige blinkt, siehe die Abhilfe in Punkt 5.

Wenn das Programm von einer Magnetkarte eingelesen wurde, führen Sie den Lese/Schreibtest durch. Wenn die Testergebnisse keinen Hinweis auf eine Störung ergeben, prüfen Sie die Magnetkarte mit Ihrem Programm, ob sie beschädigt oder verschmutzt ist. Siehe Abhilfen in Punkt 7.



Störung

7. Die Anzeige blinkt nach dem Lesen oder Beschreiben einer Magnetkarte.

Abhilfe

Falsch durchgeführtes Verfahren. Siehe Abschnitt VII.

Falsch gewählte Speicherbereichsverteilung.

Ein Lesefehler ist aufgetreten. Wenn andere Karten korrekt gelesen werden, prüfen Sie die erste Karte, ob sie beschädigt oder verunreinigt ist, und reinigen oder ersetzen Sie die Karte, wenn es nötig ist. Wenn die Karte verschmutzt war, siehe auch die Anwendung der Reinigungskarten für den Magnetkopf und die Antriebsrolle in Abschnitt VII. Werden andere Karten auch nicht korrekt eingelesen, verwenden Sie die Magnetkopfreinigungskarte einmal – Siehe Abschnitt VII, Anwendung der Magnetkopfreinigungskarte und Anwendung der Reinigungskarte für die Antriebsrolle.

8. Der Rechner schaltet sich nicht in den Learn-Modus, führt keine Einzelschritte aus, listet nicht auf und zeichnet nicht auf Magnetkarte auf.

Das Programm im Programmspeicher ist geschützt. Siehe Datenschutz für Programme im Abschnitt VII.

Wenn Sie Ihren Rechner zur Reparatur einsenden, legen Sie den Rechner selbst, das Adapter/Ladegerät, den Softwaremodul sowie alle Magnetkarten bei, die beim Auftreten der Schwierigkeiten mitbetroffen waren. In Ihrem Interesse versenden Sie Ihr Gerät als registriertes und versichertes Paket (nur Bundesrepublik Deutschland) und gut verpackt – wenn möglich nicht in der Original-Verpackung.

Österreich : Als Einschreibe-Sendung.

Schweiz : Als Einschreibe-Sendung

Texas Instruments kann keine Verantwortung bei Verlust oder Beschädigung unversicherter Sendungen übernehmen. Texas Instruments kann keine Haftung für programmierte Magnetkarten übernehmen, die von Kunden unaufgefordert eingesandt werden, und empfiehlt daher den Abschluss einer Versicherung für diese Magnetkarten.



Für Reparaturen außerhalb des Gewährleistungszeitraums werden Sie mit den zur Zeit der Einsendung geltenden Servicekosten belastet. Bitte geben Sie eine genaue Beschreibung Ihrer Schwierigkeiten mit dem Rechner, und legen Sie Ihre vollständige Anschrift mit Name, Straße und Wohnort mit Postleitzahl bei. Schicken Sie die Sendung sorgfältig verpackt zum Schutz gegen rauhe Behandlung an eine der Texas Instruments Niederlassungen, die auf dem Rückumschlag der Bedienungsanleitung angegeben sind.

WENN SIE FRAGEN HABEN ODER UNTERSTÜTZUNG BRAUCHEN

Wenn Sie Fragen zur Rechnerreparatur, zum Zubehörkauf oder zu den Funktionen des Rechners haben, wenden Sie sich bitte schriftlich und nur in dringenden Fällen telefonisch an die nächstgelegene Texas Instruments Kundendienstabteilung. Hier werden auch Ihre technischen Fragen zur Programmierung, speziellen Anwendungsgebieten usw. beantwortet.

Weil von vielen Seiten Vorschläge mit alten und neuen Ideen herangetragen werden, kann Texas Instruments nur die Anregungen berücksichtigen, die unverbindlich und unentgeltlich zur Verfügung gestellt werden. Texas Instruments lehnt grundsätzlich den Empfang vertraulich zu behandelnder Vorschläge ab. Wenn Sie also Texas Instruments Ihre Anregungen vermitteln oder eine selbstentwickelte Programmfolge zur Prüfung vorlegen wollen, fügen Sie bitte folgende Erklärung Ihrem Schreiben bei:

"Alle hiermit übermittelten Informationen und/oder Unterlagen werden Texas Instruments auf nichtvertraulicher und unverbindlicher Basis zur Verfügung gestellt; mit dieser Vorlage werden keine Rechtsbeziehungen zu Texas Instruments, weder ausdrücklich noch stillschweigend, weder vertraulicher noch anderer Art, begründet. Texas Instruments kann entschädigungslos frei über diese Informationen verfügen, d.h., sie insbesondere urheberrechtlich schützen, verteilen, veröffentlichen, vervielfältigen oder anderweitig verwenden, ohne daß von mir irgendwelche Ausgleichsansprüche geltend gemacht werden."



Die blinkende Anzeige weist darauf hin, daß die Anzeigekapazität des Rechners nicht beachtet oder daß eine unzulässige Operation gefordert wurde. Das Blinken wird mit **[CE]** oder **[CLR]** abgestellt. **[CLR]** löscht auch die Anzeige und unvollständige Operationen. **[CE]** stellt nur das Blinken ab, aber weitere Berechnungen mit nicht beeinflussten unvollständigen Operationen sind möglich. Die Anzeige blinkt aus folgenden Gründen:

1. Eine Recheneingabe oder ein Ergebnis (in der Anzeige oder in den Speichern) überschreiten die Kapazität des Rechners, liegen also nicht im Bereich $\pm 1 \times 10^{-99}$ bis $\pm 9.9999999 \times 10^{99}$. Der überschrittene Grenzwert blinkt und zeigt so Überlauf oder Unterlauf an.
2. Arkusfunktion mit einem unzulässigen Wert für die Funktion wie $\sin^{-1}x$ größer als 1. Der unzulässige Wert für x blinkt.
3. Wurzel oder Logarithmus einer negativen Zahl. Zur Anzeige des Vorzeichenfehlers blinkt der Logarithmus oder die Wurzel des Absolutwertes der Zahl.
4. Berechnung der Potenz (oder der Wurzel) einer negativen Zahl. Die Potenz (oder die Wurzel) des absoluten Wertes der Zahl blinkt.
5. Zwei Operationstasten wurden unmittelbar nacheinander gedrückt. Dies gilt für $+$, $-$, \times , $:$, y^x und $\sqrt[y]{}$. Die zuletzt eingegabene Zahl blinkt.
6. **[=]** oder **[)]** wurde unmittelbar nach $+$, $-$, \times , $:$, y^x oder $\sqrt[y]{}$ gedrückt. Die zuletzt eingegabene Zahl blinkt.
7. Mehr als 9 Klammern sind offen oder mehr als 8 Operationen sind unvollständig. Die 10-te Klammer oder die 9-te Operation werden nicht akzeptiert, also kann die Verarbeitung fortgesetzt werden. Die zuletzt eingegabene Zahl blinkt.
8. Division einer Zahl durch Null. In der Anzeige blinkt "9.999999 99".
9. Aufruf einer speziellen Steueroperation außerhalb des Bereichs 00 bis 39.
10. Versuch einer Aufzeichnung (**[Op]** 07) außerhalb des Bereichs 00 bis 19. Der Anzeigewert blinkt.
11. Versuch einer Speicherbereichsverteilung außerhalb der Grenzen des TI-58, TI-58C (60 Datenregister, 00 bis 59). Der programmierbare Rechner TI-59 verwendet 100 Datenregister, wenn mehr als 10 Zehnergruppen verlangt werden.
12. Die Anzeige blinkt beim Versuch, zu nicht zugeordneten Labeladressen oder außerhalb der Verteilung zu nicht-existenten Programmspeicherplätzen zu verzweigen.
13. Die Anzeige blinkt beim Versuch, ein Softwareprogramm in den Programmspeicher zu übernehmen, wenn nicht ausreichend Programmspeicherkapazität vorhanden ist.
14. Adressierung einer nicht existenten Softwareprogramm-Nummer. Der augenblickliche Anzeigewert blinkt.
15. Bei Berechnungen der linearen Regression in folgenden Fällen: Wenn die Regressionsgerade parallel zur y -Achse verläuft, beim Versuch, die Steigung, den Schnittpunkt mit der y -Achse, die Korrelation, x' oder y' zu berechnen. Verläuft die Gerade parallel zur x -Achse, blinkt die Anzeige beim Versuch, x' oder die Korrelation zu berechnen.

16. Berechnung von Steigung, Schnittpunkt, Korrelation, x' oder y' , wenn weniger als zwei Datenpunkte eingegeben wurden. Die zuletzt angezeigte Zahl blinkt.
17. Mehr als vier Operationen sind unvollständig, während lineare Regression, Trendlinienanalysen oder statistische Routinen durchgeführt werden, oder während der polar-/rechtwinkligen oder Grad-Minuten-Sekunden-Umrechnungen.
18. 0^x und \sqrt{x} erzeugen die Überlaufanzeige "9.9999999 99".
19. Es ist zu beachten, daß der Abstand r für rechtwinklig/polare Umrechnungen den Bereich 10^{-14} nicht überschreiten darf. Außerhalb dieses Bereiches liegende Werte führen zu falschen Ergebnissen.
20. Argumente, die den folgenden Definitionsbereichen nicht genügen, verursachen eine blinkende Anzeige.

Funktion	Definitionsbereich
$\sin^{-1}x, \cos^{-1}x$	$-1 \leq x \leq 1$
$\ln x \log x$	$1 \times 10^{-99} \leq x < 1 \times 10^{100}$
e^x	$-227.9559242 \leq x \leq 230.2585092$
10^x	$-99 \leq x < 100$

21. Ungerade Vielfache von $\pm 90^\circ$, $\pm \pi/2$ Radiant und ± 100 g sind nicht definierte Punkte der Tangensfunktion. Sehr kleine Vielfache führen zu einer Überlaufbedingung; Vielfache dieser Funktion können jedoch auch dann falsch berechnet sein, wenn keine Fehlerbedingung angezeigt wird. Siehe Anhang C.
22. Bei falschem Einlesen oder Beschreiben von Magnetkarten blinkt die Anzeige.

FEHLER WAHREND DES PROGRAMMABLAUFS

Wenn einer der vorausgehenden Fehler in einem Programm vorkommt, liegt die weitere Entscheidung beim Programmierer. Programmunterbrechungen sind nicht die automatische Folge einer Fehlerbedingung, ausgenommen Punkt 12 oben. Das Programm wird fortgesetzt und verwendet den Wert, der im Rechenmodus geblinkt hätte, für die weiteren Berechnungen. Wenn das Programm hält, wird der Fehler durch das blinkende Ergebnis angezeigt. Ob dieses Resultat richtig oder falsch ist, hängt von der Aufgabe selbst und von der Art der Fehlerbedingung ab. Dies ist jedoch die beste Auswahlmöglichkeit, die dem Programmierer ohne weitere Befehle zur Verfügung steht. Der Programmierer kann, wenn er es wünscht, für das Auftreten einer Fehlerbedingung eine Programmunterbrechung veranlassen. Hierzu setzt er Flag 8 oder verwendet die Fehlertests Op 18 und Op 19.

FEHLERHAFTES LESEN VON MAGNETKARTEN

Die in Ihrem Rechner verwendeten Komponenten sind mit großer Sorgfalt hergestellt und ausgesucht worden. Dennoch ist es möglich, daß das Lesen von Magnetkarten, die nicht auf Ihrem Gerät aufgezeichnet wurden, zu Problemen führen kann (blinkende Anzeige).



ANGEZEIGTE ERGEBNISSE UND GENAUIGKEIT

Rechner müssen wie andere elektronische und mechanische Geräte mit einer festen Anzahl von Regeln im Bereich vorgegebener Grenzen arbeiten.

Grundsätzlich wird die mathematische Toleranz des Rechners durch die Anzahl der Stellen gesteuert, die für die Berechnungen verwendet werden. Der Rechner arbeitet scheinbar mit 10 Stellen, wie sie in der Anzeige ausgewiesen sind, tatsächlich werden aber alle Berechnungen mit 13 Stellen ausgeführt. In Verbindung mit der eingebauten 5/4-Rundung sichern diese zusätzlichen Stellen die 10-stellige Anzeige und verbessern die Genauigkeit. Betrachten Sie das folgende Beispiel, wenn diese Schutzstellen fehlen:

$$1/3 \times 3 = .99999999 \text{ (ungenau)}$$

Das Beispiel zeigt, daß $1 \div 3$, mit 3 multipliziert, ein falsches Ergebnis erzeugt. Bei einer 13-stelligen Neuner-Reihe wird der Wert bei der Rundung auf 10 Stellen auf 1 gerundet.

Die mathematischen Funktionen höherer Ordnung verwenden iterative Berechnungen. Im allgemeinen setzt sich der kumulative Rundungsfehler nach der 10-stelligen Anzeige fort, sodaß keine Auswirkungen erkennbar sind. Die 13-stellige Darstellung einer Zahl differiert um 3 Größenordnungen gegenüber der ausgewiesenen zehnten Stelle. Auf diese Weise ist sichergestellt, daß die Ergebnisse auf 10 Stellen genau gerundet sind.

Normalerweise kann man diese Schutzstellen völlig außer acht lassen. Bei bestimmten Berechnungen können sie jedoch als unerwartetes Ergebnis in Erscheinung treten. Die mathematischen Grenzen einer endlichen Operation, Wortlänge, Abbrech- und Rundungsfehler ermöglichen nicht in jedem Fall eine völlige Genauigkeit der Schutzstellen. Deshalb kann der Rechner bei der Subtraktion zweier mathematisch gleicher Funktionen ein von Null abweichendes Ergebnis anzeigen.

Beispiel: $\sin 45^\circ - \cos 45^\circ \neq 0$

Winkelmodus: Altgrad

Taste	Anzeige
45 2nd sin =	.7071067812
45 2nd cos =	.7071067812
=	7.-13

Die identischen Anzeigegergebnisse von $\sin 45^\circ$ und $\cos 45^\circ$ zeigen, daß die Funktionen auf mindestens 10 Stellen genau sind. Das Endergebnis weist auf einen Unterschied in der 13-ten Stelle hin. Der wesentliche Aspekt hierbei ist, daß Ergebnisse, die um einen Faktor von 10^{-11} bis 10^{-13} kleiner als Eingabe oder Zwischenergebnisse sind, faktisch mit Null gleichgesetzt werden.

Diese Tatsache gewinnt vor allem dann Bedeutung, wenn Sie Ihre eigenen Programme schreiben. Wenn Sie ein Rechenergebnis testen, ob es mit einem anderen Wert gleich ist, zum Beispiel **SC=1**, müssen Sie Vorsichtsmaßnahmen treffen, um falsche Berechnungen wegen der Unterschiede in den Schutzstellen zu vermeiden. "Die Folge **EE INV EE** entfernt die Schutzstellen eines Ergebnisses und nur der gerundete Anzeigewert wird weiterverwendet."



Im Standardformat der Anzeige sind die Ergebnisse für alle Berechnungen genau, sofern nicht die in Anhang B aufgelisteten Einschränkungen übertreten wurden. Hierzu gelten folgende Ausnahmen:

TRIGONOMETRISCHE FUNKTIONEN — Alle im Standardformat angezeigten Stellen sind auf ± 1 in der 10-ten Stelle für den folgenden Bereich genau: $\pm 36\,000$ Altgrad, $\pm 200\,\pi$ Radiant und $\pm 40\,000$ Gon. Wenn das Argument den Bereich $\pm 3.6 \times 10^{14}$ Altgrad ($\pm 6.2799993 \times 10^{12}$ Radiant oder $\pm 4.0 \times 10^{14}$ Gon) erreicht oder überschreitet, wird die Periodizität dieser Funktionen nicht mehr erkannt. Im allgemeinen nimmt die Genauigkeit außerhalb des angegebenen genauen Bereichs um eine Stelle je Dekade ab. Eine Ausnahme bildet der Tangens eines ungeraden Vielfachen von $\pm 90^\circ$, $\pm \pi/2$ Radiant oder ± 100 Gon, der zu einer Überlaufbedingung führt, weil die Funktion an diesen Punkten nicht definiert ist.

POTENZEN UND WURZELN — Ein Genauigkeitsverlust für Potenzen und Wurzeln kann in Berechnungen dann entstehen, wenn die Basis y sehr nahe an 1 geht und die Potenz x sehr groß wird. Zum Beispiel ist $.99999944^{-160000}$ in 8 Stellen genau, während $.99999944^{-400}$ in allen 10 angezeigten Stellen genau ist.



STÖRUNGSSUCHE IN PROGRAMMEN

Selbst der umsichtigste Programmierer kommt manchmal in Situationen, wo das Programm nicht richtig funktioniert. Der Rechner verfügt über eine Gruppe von Tasten, die die Korrektur leicht machen. Das Problem liegt bei der Feststellung des Fehlers. Natürlich sollten Sie zuerst prüfen, ob das Programm richtig eingegeben wurde. Wenn Sie hier keine Widersprüche finden, kann Ihnen dieser Anhang sicher eine Hilfe sein. Zunächst wird eine Liste der häufigsten Programmierfehler erstellt. Sobald Sie mit diesen Vorschlägen vertraut sind, haben Sie die Möglichkeit, Ihren Fehler zu erkennen. Wenn nicht, fahren Sie mit der Programm-Diagnose, dem nächsten Teil dieses Abschnitts, fort.

Hier ist besonders der Drucker PC-100A, PC-100B oder PC-100C zur Auflistung Ihres Programms oder zur Protokollierung Ihrer Berechnungen von Vorteil.

GRUNDSÄTZLICHE ÜBERLEGUNGEN

Algebraisches Operationssystem

Die meisten Probleme können Sie so in den Rechner eintasten, wie sie auch geschrieben werden, das bedeutet aber nicht, daß sie in dieser Reihenfolge interpretiert werden. Nach der algebraischen Hierarchie des Rechners wird ein Ausdruck wie $2 + 3 \times 6$ als $(3 \times 6) + 2 = 20$ und nicht als $(2 + 3) \times 6 = 30$ interpretiert.

Außerdem ist zu bedenken, daß die Funktionen mit einer Variablen nach dem Wert folgen müssen, für den die Funktion errechnet wird. Sin π , zum Beispiel, wird mit der Folge **2nd** **π** **2nd** **sin** bestimmt.

Der Gleichheitsbefehl — **=**

Der Gleichheitsbefehl schließt alle unvollständigen Operationen ab. Verwenden Sie ihn daher mit Überlegung, besonders in Unterprogrammen.

Auch das Fehlen des Gleichheitsbefehls kann Störungen verursachen. Betrachten Sie den Ausdruck $(2 \times 3)^2$. Mit der Folge **2** **X** **3** **\square^2** ermittelt der Rechner den Wert für 2×3^2 . Die richtige Tastenfolge muß hier entweder **2** **X** **3** **=** **\square^2** oder **(** **2** **X** **3** **)** **\square^2** lauten.

Unvollständige Operationen

Der Rechner kann intern neun Klammerebenen und acht unvollständige Operationen halten. Einige der Tastenfunktionen benötigen jedoch bereits 4 unvollständige Operationen. Dazu gehören die polar/rechtwinkligen sowie die Altgrad-Umrechnungen und die statistischen Funktionen. Eine Eingabe, die diese Grenzen überschreitet, wird einfach ignoriert, und als Hinweis darauf blinkt die Anlage.

Mehrfach Labels

Jedes Label kann man nur einmal in einem Programm verwenden. Das Labelsuchsystem beginnt immer bei Speicherplatz 000, und nicht an der Stelle, wo das Label aufgerufen wurde. Aus diesem Grund erfolgt die Verzweigung immer zum ersten Label und bei einer zweiten Anwendung könnte dieses Label nie aufgefunden werden.



Unterprogramm

Sechs Unterprogrammebenen sind wahrscheinlich mehr als Sie jemals benötigen. Wird ein Unterprogramm außerhalb der sechsten Ebene aufgerufen, wird keine neue Rücksprungadresse mehr im Unterprogrammrücksprung-Register gespeichert. Wenn **INV** **SBR** in der siebenten Unterprogrammebene erreicht wird, geht der Verarbeitungsfluß zur letzten oder sechsten Adresse, die im Rücksprungregister gespeichert ist, und erreicht natürlich nicht die erwartete Stelle. Diese Bedingung verursacht jedoch keine Fehleranzeige. Sie müssen also streng darauf achten, daß keine falsche Verzweigung erfolgt.

Bedenken Sie, daß eine Programmadress-Taste ein Unterprogrammaufruf ist, wenn nicht ein Verzweigungsbefehl wie **GTO** oder **2nd** **scall** vorausgeht.

Auch polar/rechtwinklige und Altgrad-Umrechnungen sowie die statistischen Funktionen belegen eine Unterprogrammebene.

Der Reset-Befehl — **RST**

Dieser Befehl ist sehr zweckmäßig, aber berücksichtigen Sie alle seine Funktionen, um unbeabsichtigte Wirkungen zu vermeiden. **RST** führt vier Funktionen aus: Rückstellung des Programmzeigers auf Speicherplatz 000, Rücksetzen aller Programmflags, Löschen des Unterprogrammrücksprung-Registers und Unterbrechung der Softwareprogramme, Rückstellung des Programmzeigers auf den Programmspeicher.

Statistische Funktionen

Bei der Verwendung der vorprogrammierten statistischen Funktionen werden die Daten in die Register R_1 bis R_6 summiert. Deshalb dürfen in einem Programm mit den statistischen Funktionen diese Datenregister nicht anderweitig belegt werden. Vor Beginn der Dateneingabe muß man sie löschen. Auch sollten Sie veranlassen, daß der Wert im T-Register erhalten bleibt, wenn er später im Programm benötigt wird. Wie oben erwähnt, erfordern diese Funktionen bis zu vier unvollständige Operationen und eine Unterprogrammebene.

Polar/Rechtwinklige Umrechnungen

Primär muß man hier an die Wahl der richtigen Winkleinheit denken. Auch in diesen Umrechnungen wird eine Unterprogrammebene belegt und bis zu 4 Operationen sind unvollständig.

Wahl des Winkelmodus

Beim Einschalten ist Ihr Rechner auf Altgrad eingestellt. Wenn Winkel in Radiant oder in Gon ausgedrückt werden sollen, müssen Sie mit den entsprechenden Tasten die Anweisungen eingeben. Der Rechner bleibt dann im gewählten Winkelmodus, bis eine andere Einstellung vorgenommen wird.

Funktionen, die nur mit dem Anzeigewert arbeiten

EE und **2nd** **DMS** arbeiten nur mit dem Inhalt der Anzeige, nicht des Anzeigeregisters. Das heißt, daß alle Schutzstellen und alle Stellen, die durch eine Festkomma-Einstellung unterdrückt werden, bei Anwendung dieser Befehle verloren gehen.



T-Register Vergleiche

Bei der Entscheidung, ob eine Verzweigung zu einem neuen Programmspeicherplatz erfolgt oder nicht, wird mit **2nd** **xx=1** oder **2nd** **xx≥1** der Inhalt des Anzeigeregisters mit dem Inhalt des T-Registers verglichen. Versuchen Sie die nachstehende Folge als Beispiel zu der Art des Problems, das bei Anwendung dieser Befehle auftreten kann.

```

2nd Deg
45 2nd sin
xx=1
45 2nd cos
2nd xx=1 114
  
```

Wenn Sie jetzt den Learn-Modus eingeben, stellen Sie fest, daß diese Verzweigung nicht erfolgte, obwohl $\sin 45^\circ$ und $\cos 45^\circ$ mathematisch gleich sind. Die Ursache liegt in der Rundung der Schutzstellen, wenn der Rechner diese Werte ermittelt (siehe Anhang C). Subtrahieren Sie $\cos 45^\circ$, um diesen Nachweis auch selbst zu erbringen. Sie erhalten ein von Null abweichendes Ergebnis als Hinweis, daß diese Werte in den Schutzstellen differieren. Im allgemeinen bleiben diese Differenzen unbemerkt, aber sie müssen bei der Arbeit mit bedingten Verzweigungen berücksichtigt werden. Die Folge **EE** **INV** **EE** trennt die Schutzstellen vom Ergebnis, und für die weitere Anwendung bleibt nur der gerundete Anzeigewert.

Vorsicht auch bei den Funktionen **EE** und **2nd** **DMS**, wo die Schutzstellen gelöscht oder geändert werden.

Redigieren

Redigieren Sie Ihre Programme sorgfältig. Selbst einfache Änderungen, die an einer Stelle nahezu bedeutungslos sind, können an anderer Stelle Störungen verursachen. Ziehen Sie alle möglichen Auswirkungen einer Änderung in Betracht. Einige Punkte, die besondere Aufmerksamkeit erfordern, sind die kombinierten Adressen, gleichlautende Labels, kombinierte Befehle (wie **INV** **SBR**), sowie Adressen, die als Tastenkodes interpretiert werden können. Ein geschütztes Programm können Sie nicht redigieren. Bedenken Sie, daß sich beim Einfügen und Löschen von Befehlen Teile des Programms gleichbleibend nach oben und unten verschieben. Verzweigungsbeefehle mit absoluter Adressierung müssen also entsprechend korrigiert werden.

Speicherbereichsverteilung

Achten Sie darauf, daß die Datenregister und der Programmspeicher innerhalb Ihrer Verteilung liegen. Ein Datenregister kann 8 Programmbefehle aufnehmen. Jedes Ziffern paar in einem Datenregister ist ein potentieller Befehl. Vorsicht deshalb bei Einstellung einer neuen Verteilung, daß der Inhalt eines Datenregisters nicht zu 8 Programmbefehlen wird, oder umgekehrt.



PROGRAMM-DIAGNOSE

Unter Beachtung der oben angegebenen Punkte hier einige Vorschläge zur Programmdiagnose. Diese Anregungen sollen Ihnen die Auswertung von Programmen ermöglichen, die nicht richtig funktionieren.

Das Programm wird nicht abgeschlossen

Wenn das Programm nicht zum erwarteten Zeitpunkt beendet wird, läuft das Programm in einer endlosen Schleife. Das beste Verfahren ist die genaue Analyse des Programms in Einzelschritten und die Überprüfung jedes Befehls, wobei den Verzweigungsbefehlen besondere Beachtung geschenkt werden muß. Obwohl die Ursache wahrscheinlich bei einer bedingten Verzweigung liegt, sollten Sie zuerst die unbedingten Sprünge testen, weil sich hier der Fehler leichter feststellen läßt. Achten Sie vor allem auf Befehlsfolgen wie `[2nd] [1b] [D] [GTO] [D]`. Um die Schleife zu verlassen, ist hier eine bedingte Verzweigung erforderlich. Prüfen Sie als nächstes die bedingten Verzweigungen, besonders, wenn sie als Schleifenanfang oder als Schleifenende programmiert sind.

Der Befehl `[2nd] [05]` darf das Programm nicht in eine endlose Schleife schicken, es sei denn, daß das Programm eine Aktion durchführt, um den Inhalt des zu vermindernenden Datenregisters zu ändern, oder daß der Registerwert $\geq 10^{10}$ ist. Stellen Sie sicher, daß dieses Datenregister auf Null gebracht werden kann. Wenn in dem Datenregister, das vermindert werden muß, eine sehr große Zahl gespeichert ist, kann das Programm bis zum Abschluß außergewöhnlich lange laufen und sich nur scheinbar in der endlosen Schleife befinden. Wenn jedoch das Programm die Anzahl der benötigten Schleifen bestimmen soll, wollen Sie diese Berechnung vielleicht überprüfen.

Bedingte Verzweigungen mit T-Registervergleichen sollen ebenfalls sorgfältig nachgeprüft werden. Wenn Sie eine Verzweigung dieses Typs zur Schleifensteuerung verwenden, erwarten Sie im allgemeinen, daß Ihre Berechnungen innerhalb vorgegebener Grenzen konvergieren. Wenn die Berechnungen nicht konvergieren, wird die Schleife folglich nicht beendet. Prüfen Sie Ihre Gleichungen sowie die Befehle, mit denen Sie die Gleichungen programmiert haben. Vergewissern Sie sich auch, ob der richtige Testwert im T-Register gespeichert ist. Wenn das Programm diese Zahl bestimmen soll, prüfen Sie auch diese Berechnungen. Ein weiteres Problem kann sich aus der Tatsache ergeben, daß diese Befehle den vollen T-Registerinhalt mit dem vollen Anzeigeregisterinhalt vergleichen, ehe die Entscheidung getroffen wird, ob die Verzweigung zu einem neuen Speicherplatz erfolgt oder nicht. (Siehe auch T-Register Vergleiche oben und BEDINGTE VERZWEIGUNGEN im Abschnitt V).

Wenn Sie ein Softwareprogramm als Unterprogramm verwenden, können Sie `[RST]` drücken. Wird das Programm dann unterbrochen, lief das Softwareprogramm in einer endlosen Schleife. In diesem Fall schlagen Sie bei den Programm-Instruktionen des betreffenden Softwareprogramms nach. `[RST]` ist eine Notmaßnahme und soll nur das letzte Mittel sein, weil man die Stopstelle nicht vorhersagen und den Anzeigewert nicht identifizieren kann.

Wenn Sie nach dieser Auswertung keinen Fehler finden können, siehe "Anwendung des Rechners in der Fehlerdiagnose" in diesem Abschnitt.



Widerspruchsfreie Daten liefern widersprüchliche Ergebnisse

Fehler dieser Art werden im allgemeinen durch bedingte Verzweigungen verursacht, die bei falscher Anwendung einmal korrekte und beim nächsten Mal falsche Ergebnisse liefern können. Betrachten Sie zur Veranschaulichung folgendes Beispiel:

```

2nd  Lbl
A
INV
2nd  <=>
B
2nd  St/Flg
3
2nd  Lbl
B
.
.
.
2nd  If/Flg
3
C
.
.
.
2nd  Lbl
C
.
.
.

```

Hier will der Programmierer einen Teil des Programms überspringen, wenn seine ursprünglichen Daten kleiner als Null waren. Angenommen, die folgende Datengruppe wird in das Programm eingegeben: 12, -16, 12. Das Programm liefert bei den ersten beiden Eingaben korrekte Ergebnisse. Wenn jedoch 12 zum zweiten Mal eingegeben wird, ist das Ergebnis falsch. Der Grund ist, daß bei Eingabe von -16 Flag 3 gesetzt wurde. Weil das Programm keinen Befehl enthält, der Flag 3 zurücksetzt, wenn positive Eingaben gemacht werden, wird 12 beim zweiten Mal als negative Eingabe behandelt. In diesem Fall kann man die Störung beseitigen, wenn man nach 2nd Lbl E die Folge INV 2nd St/Flg 3 programmiert.

Ähnliche Probleme können bei jeder Verzweigungsoperation auftreten. Leider ist es nicht möglich, für jeden Fall ein Beispiel anzugeben. Generell gilt, daß man als eine erste Maßnahme bei der Fehlerdiagnose eines Problems dieser Art ein Muster in den Ergebnissen sucht. Im obigen Beispiel lieferten nur positive Eingaben falsche Ergebnisse, und nur nach einer negativen Eingabe.



Natürlich werden nicht alle Fehler dieser Art durch Verzweigungsoperationen verursacht. Betrachten Sie eine Situation, in der bei Eingabe der gleichen Daten mehrere Male nacheinander gleichbleibend unterschiedliche Ergebnisse erzeugt werden (z.B. Ansteigen der Werte nach einem erkennbaren Schema).

2nd Lbl

A

.

.

.

SBR

SUM

.

.

.

2nd Lbl

SUM

SUM

1

2

.

.

.

INV SBR

Hier ist das Problem auf die unachtsame Anwendung eines Datenregisters zurückzuführen. Wenn R_{12} nie gelöscht wird, und vorausgesetzt, daß die Anfangsoperation für dieses Datenregister ein **STO**-Befehl ist, steigen die Werte der Ergebnisse konstant an.

Wie bereits erwähnt, ist der Schlüssel zu derartigen Problemen das Erkennen eines Musters in den Resultaten. Wenn Sie ein solches Muster nicht ohne großen Aufwand finden können, siehe Anwendung des Rechners in der Fehlerdiagnose.



In den meisten Fällen lassen sich solche Situationen vermeiden, wenn man jedem Programm eine vorbereitende Tastenfolge angliedert. Ein typisches vorbereitendes Verfahren ist nachstehend angegeben. Es wird einfach über die Taste **E** aufgerufen.

Tastenfolge	Bemerkungen
R/S	Beendet die Folge (Speicherplatz 000)
2nd Lbl E	
2nd CM5	Löschen der Datenregister
CLR	Löschen der Anzeige
2nd CP	Löschen des T-Registers
INV 2nd fix	Aufheben der Festkomma-Einstellung
RST	Rücksetzen aller Flags
	Löschen des Unterprogrammrücksprungregisters
	Einstellen des Programmzeigers auf 000

Gleichbleibend falsche Ergebnisse

Es ist möglich, daß die Lösungsfolge eines Programms fehlerhaft geschrieben wurde, wenn gleichbleibend die selben falschen Ergebnisse erzeugt werden, unabhängig, welche Daten Sie eingeben. Wenn Sie beim manuellen Durcharbeiten Ihrer Gleichungen keinen Fehler finden, und feststellen, daß sie für alle Fälle gültig sind, beziehen Sie sich auf das folgende Beispiel

Anwendung des Rechners in der Fehlerdiagnose

Sobald Sie bestimmt haben, welche Werte zu berechnen und anzuzeigen sind, und wo sie in den verschiedenen Verarbeitungsstufen des Programms gespeichert werden müssen, kann man den Rechner als das wirksamste Mittel einsetzen, um fehlerhaft laufende Programme zu überprüfen.

Ein wichtiger Hinweis: wenn Sie an einem Prüfpunkt den Inhalt eines Datenregisters zum Testen aufrufen, müssen Sie vor der Fortsetzung des Programmablaufs sicherstellen, daß der Inhalt des Anzeigerregisters wiedereingesetzt wurde. Andernfalls können nicht existierende Programmfehler scheinbar auftreten. Es empfiehlt sich, die Daten zur Überprüfung mit **2nd** **ExC** in die Anzeige aufzurufen und sie dann mit derselben Folge wieder zu vertauschen. Damit kommt der zuletzt berechnete Wert wieder in die Anzeige.

Es gibt mehrere Befehle zur Programmanalyse, wenn das Programm abläuft. Das Einfügen von **R/S**-Befehlen an verschiedenen Schlüsselstellen im Programm ist eine schnelle Methode, um herauszufinden, wo ein Fehler erstmals auftritt. Wenn das Programm unterbrochen wird, prüfen Sie nicht nur den Inhalt der Anzeige, sondern auch den der Daten- und Testregister.

Beim Einfügen der **R/S**-Befehle ist es am einfachsten, mit dem höchstnumerierten Programmspeicherplatz zu beginnen und nach rückwärts zu arbeiten. Die Eingabe der **R/S** Befehle in umgekehrter Reihenfolge verschiebt keine Speicherplätze, und Sie erhalten dennoch Zugang zu diesen Befehlen.



Sobald Sie Gegensätze auffinden, lassen Sie das Programm erneut ablaufen und unterbrechen Sie es einen **[R/S]**-Befehl vor dem Stop, wo der Fehler festgestellt wurde. Setzen Sie jetzt die Durchführung des Programms mit der Taste **[SST]** in Einzelschritten fort, bis Sie die genaue Position des Fehlers gefunden haben. Nach der Identifizierung des Problems und nach der Fehlerkorrektur löschen Sie die **[R/S]**-Befehle der Reihe nach. Wiederholen Sie diesen Vorgang, bis alle Fehler festgestellt und korrigiert sind.

Anstelle von **[SST]** kann man die Taste **[GTO]** verwenden und gedrückt halten. Damit wird zwischen der Ausführung jedes Schrittes eine kurze Pause eingelegt, und ein kurzer, aber automatischer Blick auf den Fortlauf des Programms ist möglich. Bei dieser Methode müssen Sie zuerst das Programm starten. Um sicherzustellen, daß Sie die Ergebnisse aller Programmbefehle von Anfang an beobachten können, befolgen Sie nachstehende 3 Punkte:

1. Taste **[R/S]** drücken und niederhalten.
2. Während die Taste **[R/S]** gedrückt bleibt, Taste **[GTO]** drücken und niederhalten.
3. Taste **[R/S]** wieder loslassen.

Am einfachsten ist die Diagnose, wenn das Problem zu einer Fehlerbedingung führt. In diesem Fall setzen Sie einfach Flag 8 und lassen das Problem erneut ablaufen. Wenn jetzt der Fehler erreicht wird, unterbricht das Programm. Über die Taste **[LRN]** wird der Programmspeicherplatz aufgezeigt, wo der Fehler auftrat. (Tatsächlich wird die Verarbeitung unterbrochen, wenn der Programmzeiger auf den ersten Speicherplatz nach dem Fehler eingestellt ist.) Sie können dann die Art des Fehlers bestimmen und die nötige Korrektur vornehmen.

Achtung: Die Durchführung eines Softwareprogramms in Einzelschritten ist nicht möglich. Wenn Sie den Programmspeicher, der ein Softwareprogramm aufruft, in Einzelschritten durchgehen, wird die Anzeige während des Ablaufs des Softwareprogramms völlig dunkel. Nach Abschluß dieser Routine können Sie mit **[SST]** oder **[GTO]** die Durchprüfung des Hauptprogramms fortsetzen. (Wenn Sie **[GTO]** verwenden, dürfen Sie diese Taste beim Ablauf des Softwareprogramms nicht loslassen, damit kein Teil des Hauptprogramms der Überprüfung entgeht.)

Anwendung des Druckers in der Fehlerdiagnose

Der Drucker ist ein weiteres wertvolles Hilfsmittel für die Fehlerdiagnose in einem Programm. Wenn Ihr Rechner auf den Drucker aufgesetzt ist, sind folgende Maßnahmen möglich.

- (1) Drücken Sie **[RST] [2nd] [List]**, um eine vollständige Auflistung Ihrer Programmbefehle zu erhalten, Speicherplatznummern, Befehlskodes und Befehlsmnemonic. Damit erübrigt sich die Durchprüfung eines Programms in Einzelschritten im Learn-Modus wie das Umsetzen der Befehlskodes in Programmbefehle, um nachzuweisen, ob das Programm richtig eingegeben wurde.
- (2) Führen Sie das Programm durch und schalten Sie dabei den Drucker in den Protokollmodus (Parallelbetrieb). Sie können den Ablauf der Berechnungen Schritt für Schritt verfolgen und genau festlegen, wo das Programm von der beabsichtigten Lösung abweicht.
- (3) Drücken Sie **X X [INV] [2nd] [List]**, und Sie erhalten eine Liste aller Datenregisterinhalte ab **R_{xx}**. Unterbricht man das Programm an verschiedenen Schlüsselpositionen und führt dann diese Operation durch, können Sie leicht nachprüfen, ob die Datenregister zur rechten Zeit mit den richtigen Größen belegt sind.
- (4) Drücken Sie **[RST] [2nd] [Op] 08** und Sie erhalten einen Ausdruck aller Labels und ihrer absoluten Adressen. Wenn Sie diese Eigenschaft nutzen, müssen Sie die Position Ihrer Labels nicht von der gesamten Programmauflistung herausuchen.



SACHREGISTER

- A**
Absolute Adressierung IV-44, 86, V-57
Absolutwert-Taste V-20
Adressierung
 Absolute- IV-44, V-57
 Indirekte- IV-84, V-68
 des Programms IV-44, V-57
 der Speicher II-6
 Kurzform- IV-15, 47, V-22, 58
Additionstaste II-2, V-10
Addition zum Speicher II-7, V-24
Algebraisches Operationssystem (AOS) II-3, V-11
Algebraische Hierarchie II-3, V-11
Allgemeine Labels IV-43, V-56
Alphanumerische Operationen VI-7
Algebraische Funktionen II-10, V-15
Altgrad-Taste II-12, V-16
Analyse Trendlinie V-39
Antilogarithmen UU-11
Anzeige
 Charakteristika II-8, V-1, 5, 44
 Blinken der- V-9, B-1
 Kontrolle der- II-8, V-8
 des Programms V-17
 -Register II-8, V-5
 Unterlauf der- B-1
 Überlauf der- B-1
Arithmetik, Speicher II-7, V-24
Arithmetische Funktionen II-2, V-10
Arkuskosinus II-12, V-18
Arkussinus II-12, V-18
Arkustangens II-12, V-18
Aufladen des Batteriepakets A-1
Austausch-Taste II-6, V-26
Auf- oder Abrunden C-1
Aufzeichnen von Magnetkarten VII-2
Aufruftaste II-6, V-23
Aufgezeichnete Programme I-3, III-1
Auflistung
 der Inhalte der Datenregister VI-4
 der Labels VI-11
 des Programms VI-4
- B**
Batterieaustausch A-2
Batteriebetrieb A-1
Back-Step-Taste (Rückschritttaste) IV-21, V-48
Bearbeitungsgebühren-Programm IV-93
Befehlskodes IV-93
Bedingte Sprünge II-57, V-62
Blorhythmen IV-53
Blindoperationen V-15
Blinkende Anzeige V-9, B-1
- D**
Datenspeicherregister
 Adressierung der- II-6, V-22
 Anzahl II-6, V-22, V-42
 Auflisten der Inhalte der- VI-4
 Indirekte Adressierung der- IV-84, V-68
 Inkrement/Dekrement der- V-29
Dekadischer Antilogarithmus II-11, V-16
Dekadische Logarithmus-Taste II-11, V-16
Delete-Taste IV-21, V-51, 52
Dezimalgrad in Minuten und Sekunden II-13, V-30
Diagnose der Software A-4
Diagnose des Rechners A-3
Direkte Registrierarithmetik II-7, V-24
Divisionstaste II-2, V-10
Divisionstaste, Speicher II-7, V-24
DMS-Taste II-13
Doppelfunktions-Taste II-5, V-3
Drucker
 Auflistung mit dem- VI-4
 Merkmale des VI-1
 Operationen mit dem- VI-1
 Pflege des- V-12
Druckregister VI-8
DSZ-Anweisungen IV-57, 71, V-63
DSZ-Taste IV-71, V-63
- E**
e^x-Taste II-11, V-16
Einfüge (INS)-Taste IV-21, V-52
Eingabe von Programmen IV-16
Eingabe von 10er Exponenten II-8, V-5
EIN/AUS (ON/OFF) I-2
Einzelschritt-Taste IV-21, V-48
Entscheidungsbefehle IV-57
Entladene Batterie A-1
Erstfunktions-Tasten II-5, IV-17, V-3
Exponenten V-5
- F**
Fakultät-Programm VI-72, V-65
Fehlerbedingungen V-67, B-1, D-1
Festkomma-Taste II-9, V-8
Flags
 Indirekte Adressierung IV-87
 Sonderfunktionen IV-65, V-67
Fließkomma II-8, V-2
Flußdiagramme IV-4
Format der Anzeige II-8, V-5
Fragezeichen (Drucker) VII-2, 5
Funktionen, Algebraische II-10, V-15



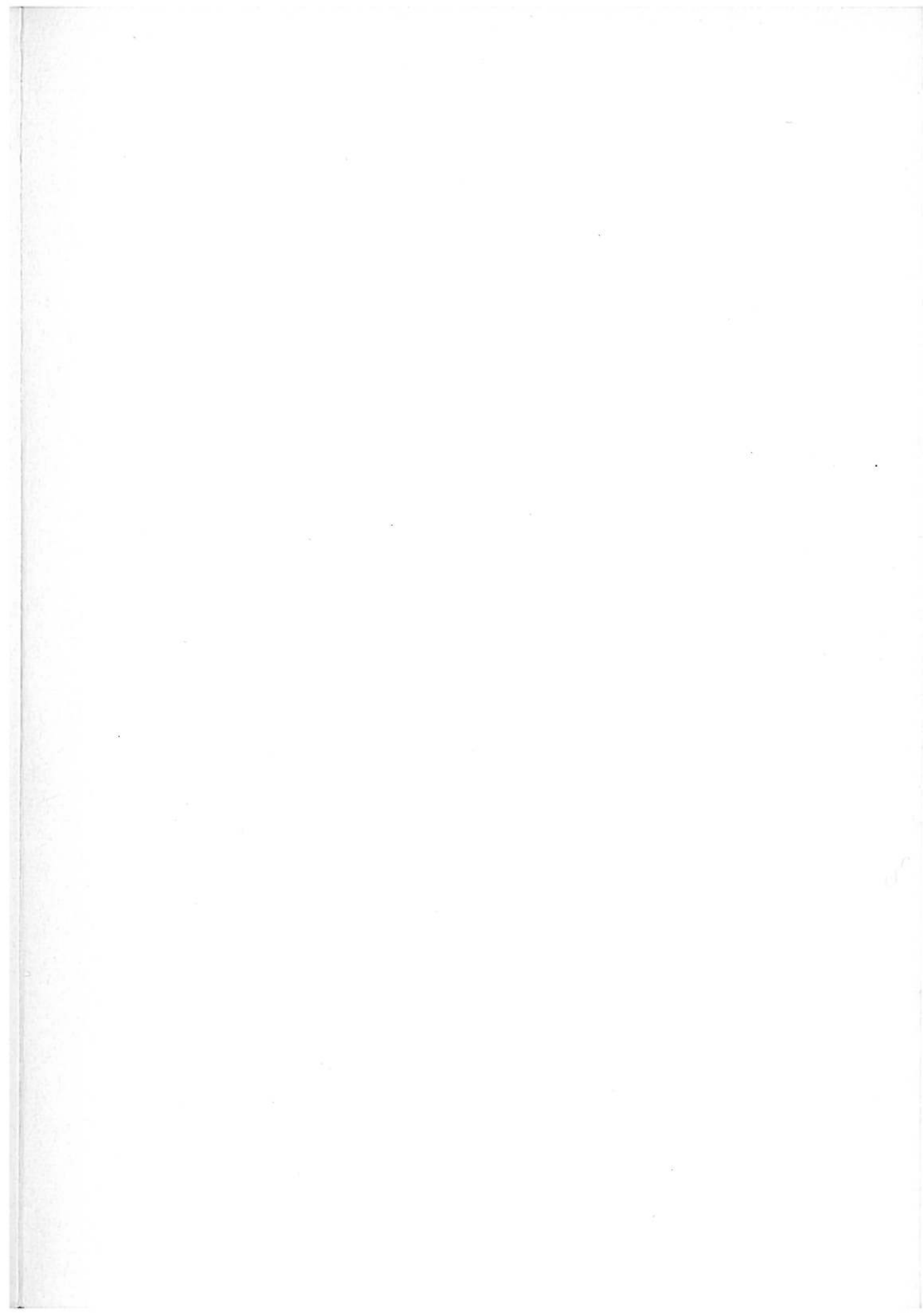
G	
Ganzzahl-Taste (INT)	II-8, V-10
Ganzzahl, Löschen der	V-20
Genauigkeit	C-1
Gewichtung, Statistik	V-19
Gleichheitstaste	II-2, V-10
Go to Taste	IV-44, V-56
Grad in Radiant Umrechnung	V-19
Grundrechenarten	II-2
H	
Hauptprogramm	IV-47
HI-LO Spiel	I-3
Hierarchie Algebraische	II-3, V-11
I	
If Flag-Taste	IV-61, V-65
If $X = t$ -Taste	IV-57, 60, V-62
If $X \geq t$ -Taste	IV-57, 60, V-62
Indirekt-Taste	IV-84, V-68
Indirekte Adressierung	
von Programmen	IV-84, V-68
von Datenspeichern	IV-84, V-68
Inkrement/Dekrement von Daten- speichern	V-29
Instruktionen, Programm	I-3
Inverse Trigonometrische Funktionen ..	II-12, V-18
K	
Karten, Magnet	VII-1
Datenschutz	VII-4
Einlesen, Laden von-	VII-5
Beschreiben - Aufnehmen von - ..	VII-2
Fehler, Lesen/Schreiben	VII-2
Markieren, Kennzeichnen	VII-2, 8
Pflege	VII-7
Reinigung	VII-9
Schutz	VII-4
Karte, Reinigung des Magnetkopfes ..	VII-8
Karte, Reinigung der Transportrolle ..	VII-8
Klammertasten	II-4, V-12
Kodes	
Befehls-Tasten-	IV-17, V-44, 48
Kombinierte-	IV-17, V-51
Steueroperations- (Op)	V-27
Kodebrecher-Spiel	IV-101
Komma-Taste	II-2, V-2
Kontrollsymbole, Drucker	VI-5
Kopfreinigen, Drucker	VII-9
Kopfreinigungskarte-Anwendung	VII-9
Korrektur von Programmen	IV-21, V-48, 51
Kosinus-Taste	II-12, V-17
Kosekans	V-18
K	
Kotangens	V-18
Kugelkoordinaten-Programm	IV-38
Kurswert von Festverzinslichen	
Wertpapieren	IV-75
Kurzformadressierung	IV-15, 44, V-22, 58
L	
Labelauflistung	IV-11
Label, Allgemeine	IV-43, V-56
Programmadress-	IV-11, V-55
Label-Taste	IV-11, V-55
Learn-Taste/Learn-Modus	I-4, VI-8, V-43, 44
Lesen (Laden) von Magnetkarten	VII-5
List-Taste (Auflisten)	IV-4
Lineare Regression	II-17, V-36
Logarithmus	
Dekadischer (Zehner)-	II-11, V-16
Natürlicher-	II-11, V-16
Löschen der Fehlerbedingungen	B-3
Löschstasten	V-3
Allgemeine-	II-2, V-3
Datenspeicher-	II-6, V-23
Eingabe-	II-2, V-3, 15
Programmspeicher	IV-16, V-3, 41, 43
M	
Magnetkarten (Siehe Karten, Magnet)	
Mantisse	II-8, V-5
Metrische Umrechnungen (Programm) ..	IV-65
Mittelwert	V-33
Minuszeichen	II-8, V-1
Module Software	III-1
Multiplikationstaste	II-2, V-10
Multiplikation mit Datenspeichern	II-7, V-24
Multiplikation von zwei Klammer- ausdrücken	II-4, V-13
N	
Natürlicher Logarithmus-Taste	II-11, V-16
Natürlicher Antilogarithmus	II-11, V-16
Negative Zahlen	V-1, 2
Neugrad Taste	II-12, V-16
Netzteil/Ladegerät	A-1
Netzbetrieb	A-1
Notation	
technische-	IV-99, V-51
Null-Operationstaste	IV-99, V-51
O	
Operation, Typen von	I-2
Optimierung von Programmen	IV-89

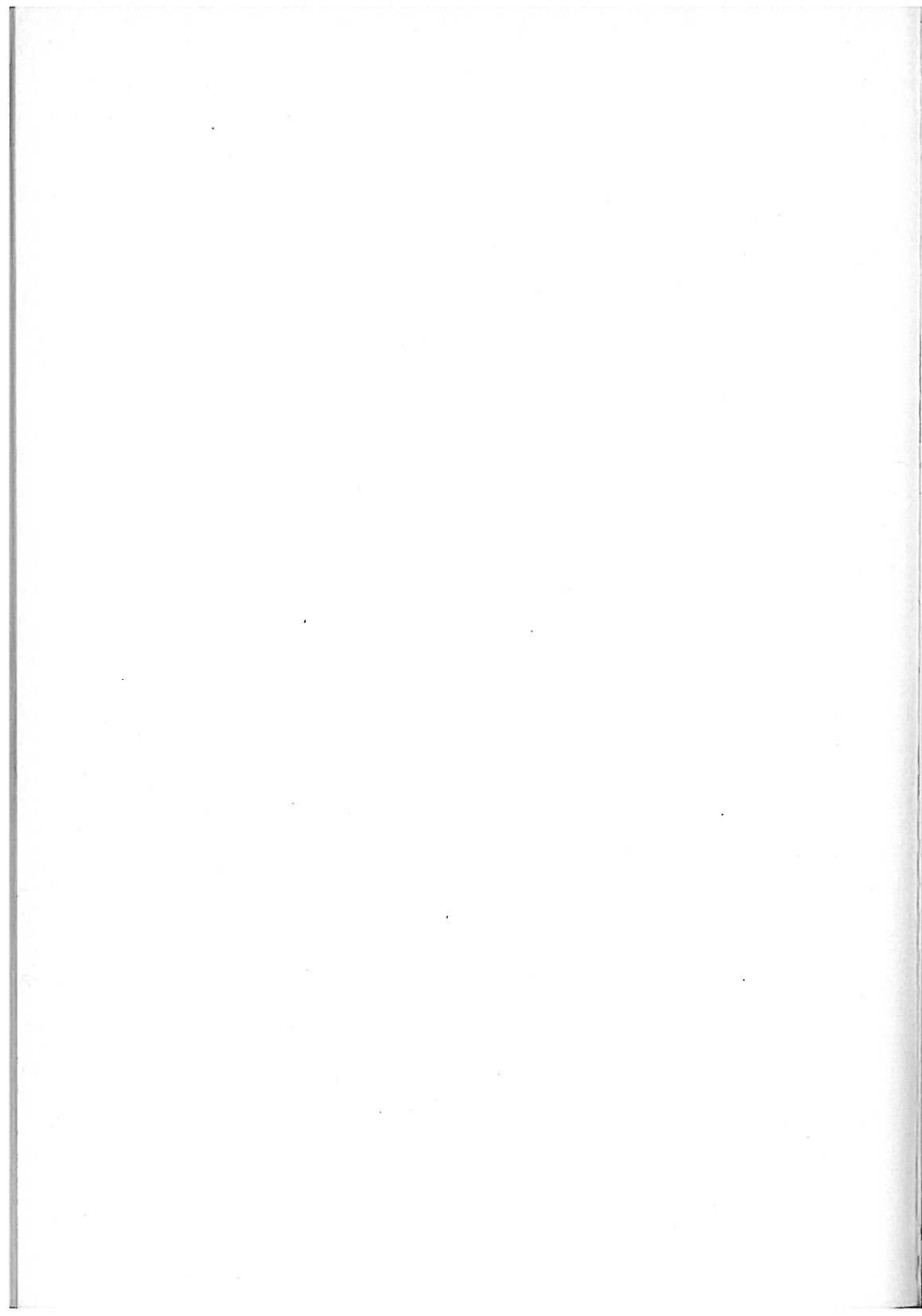


P		
Papiervorschub-Taste	VI-3	
Papier Einlegen (Siehe Anleitung PC-100A/B/C)		
Pausentaste	V-44	
Pi (π) Taste	II-2, V-2	
Plotten (Drucker)	VI-10	
Plus/Minus-Taste	II-2, V-2	
Polar-Rechtwinklige Umrechnung	II-14, V-30	
Potenzen	II-10, V-21	
Preisrechnungsprogramm	IV-32	
Produkt, Speicher	II-7, V-24	
Programme, Software	III-1	
Programm (e) (n)	IV-1	
Ablauf der	V-46	
Anzeigen der	IV-17, V-44	
Auflisten der	VI-4	
Aufzeichnen der	I-4	
Datenschutz der	VII-4	
Eingabe der	IV-16, V-45	
Fehlerbedingungen in	B-2, D-1	
Fehlersuche in	D-1	
Flags in	IV-61	
Flußdiagramme für	IV-4	
Haupt	IV-47	
Löschen von	IV-17, V-44	
Modul	II-1	
Optimieren von	IV-89	
Parallelbetrieb in	D-8	
Persönliche	IV-27	
Plätze in	IV-17, V-44	
Redigieren	IV-21, V-48, 51	
software	I-3, III-1	
softwaremodul	I-3, III-1	
Speicher für das	V-42	
Taste	I-3, III-3	
übernehmen	III-4, V-28	
Unterprogramme	IV-46	
Verzweigungen im	IV-44	
Zeiger	V-47	
Programmadress-Labels	IV-11, V-55	
Programmadressstasten	IV-11, V-55	
Programmierung	IV-1	
elementare	IV-2	
fortgeschrittene	IV-1	
Grundfunktionen	IV-1, V-41	
Schritte der	IV-10	
Sprache	IV-10	
Techniken	IV-10	
Programm-Instruktionen	I-3	
Programm-Löschstaste	V-3, 43	
Programmspeicher	V-41	
Programmspeicherplatz	V-44	
Programmzeiger	V-44	
Q		
Quadratische Gleichung (Programm)	IV-79	
Quadrat-Taste	II-10, V-10	
Quadratwurzel-		
Beispiele	V-59	
Taste	II-10, V-20	
R		
Radiant/Altgrad-Umrechnung	V-19	
Radiant/Neugrad-Umrechnung	V-19	
Radiant-Taste	II-12, 16	
Rechtwinklig/polare Umrechnung	II-15, V-30	
Redigieren von Programmen	IV-21, V-48, 51	
Register		
Anzeige	II-8, V-5	
Arithmetik	II-7, V-24	
Daten(speicher)	II-6, V-22	
Drucker	IV-8	
Rechner	I-1, V-22	
T-Register	II-14, IV-57, V-62	
Unterprogrammrücksprung	IV-46, V-58	
Regression, lineare	V-36	
Reinigungskarte für das Transportsystem	VII-8	
Reset-Taste	V-44	
Reziprokwert-Taste	II-10, V-15	
Rücksprungregister	IV-46, V-58	
Rücksprung, Unterprogramm	IV-46, V-58	
Rundungsstellen	V-5, C-1	
Run/Stop-Taste	IV-4, V-43	
S		
Schleifen	IV-68, V-64	
bedingte	IV-70	
DSZ	IV-71	
unbedingte	IV-68	
Service-Informationen	A-1	
Set-Flag-Taste	IV-61, V-65	
Signum-Funktion	V-28	
Sinus-Taste	II-12, V-17	
Software	III-1	
Softwareprogramme	I-3, III-1	
Solid-State-Software	I-3, III-1	
Speicherarithmetik	II-7, V-24	
Speicher, Daten (Siehe Datenspeicherregister)		
Speichertasten	II-6, V-23	
Spezielle Steueroperationen	V-27, IV-7	
Spezifikation des Rechners	I-1	
Standardabweichung	V-33	
Standardformat der Anzeige	II-8, V-1	
Standard-Softwareprogramme	I-3, III-3	
Statistik	V-32	
Steueroperationen, spezielle (Op.)	V-27	
Störungssuche		



S	
Operationen	A-3
Programme	D-1
Stunden-Minuten-Sekunden-Umrechnung	II-13, V-30
Subtraktionstaste	II-2, V-10
Subtraktion vom Speicher	II-7, V-24
Summentaste (Speichaddition)	II-7, V-24
Symbole, Drucker	VI-5, 7
T	
Tabelle der Tastenkodes	V-48, 49 50
Tangentstaste	II-12, V-17
Tastaturberechnungen	I-4, II-2
Tastenindex	Umschlag
Tastenkod-Auflage	V-49
Tastenkodes	IV-17, V-44, 48
Tasten, Programmdresstasten	IV-11
Technik der Programmierung	IV-10
Technische Notation	II-9, V-8
Temperatur-Umrechnungen	I-4
Testoperationen	V-29, 62
Trace-Taste, Drucker	VI-5
T-Register	II-15, IV-57
-Vergleiche	V-62
Trendlinienanalyse	V-39
Trigonometrische Funktionen	II-12, V-17
-Definitionsbereiche	C-2
U	
Überlauf der Anzeige	B-1
Übernahme von Softwareprogrammen	III-4, V-28
Umkehrfunktions-Taste	II-5, V-3, 18
Umrechnungen	
Altgrad/Radiant/Neugrad	V-19
Grad-Minuten-Sekunden/	
Dezimalgrad	II-13, V-30
polar-rechtwinklig	II-14
Unbedingte Verzweigung	IV-44, V-56
Unterlauf der Anzeige	B-1
Unterprogramme	
Aufruf	IV-48
Rücksprungregister	IV-46, V-58
SBR-Taste	IV-46, V-58
Softwareprogramme als	IV-52, V-60
wichtige Faktoren bei	IV-49
Unterstützung	A-4
Unvollständige Operationen	II-3, V-11
V	
Variable, Programm	IV-2
Varianz	V-33
Versandanweisungen	A-4
Verteilung (Programm-Datenspeicher) ..	V-22, 29, 42
V	
Verzweigen	V-62
Verzweigungsbefehle im Programm.	II-43, V-56
Vorzeichenwechsel-Taste	II-2, 8, V-2
W	
Wahlweiser Ausdruck	VI-2
Wartung und Service	A-1
Winkelberechnung	II-12, V-16
Winkelmodus-Umrechnung	V-19
Write-Taste	VII-2, 5
Wurzeln	II-10, V-21
-Grenzen	C-2
X	
X1-Fakultät, Programm	IV-22
X-t-Austauschtaste	II-14, IV-57, V-30,
X=t-Taste	IV-57, 60, V-62
X ≥ t-Taste	IV-57, 60, V-62
X ² -Taste	II-10, V-20
\sqrt{x} -Taste	II-10, V-21
Yx-Taste	II-10, V-21
Z	
Zeiger, Programm	IV-7, V-44, 47
Zeitdifferenzprogramm	IV-18
Zifferntasten	II-2, V-2
Zinseszinsprogramm	IV-27
Zweifunktionstasten	II-5, IV-17, V-3



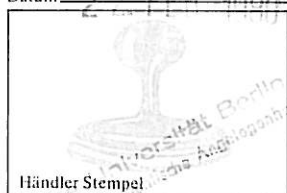


Bitte die anhängende Karte sorgfältig ausfüllen und sofort an die für Sie zuständige Texas Instruments Niederlassung absenden (Adresse siehe Rückseite Bedienungsanleitung).

TEXAS INSTRUMENTS

NACHWEIS ÜBER DAS KAUFdatum

Datum 22 FEB 1980



Modell TI-58 Serien Nr. _____

Modell TI-58C 2304655 Serien Nr. _____

Modell TI-59 Serien Nr. _____

1 ☐ Herr 2 ☐ Frau, Fräulein 3 ☐ Firma

Familienname _____ Vorname _____

Firma _____ Strasse _____

Ort _____ Postleitzahl _____ Land _____

Bitte beschreiben Sie den Fehler _____



TEXAS INSTRUMENTS

GEWÄHRLEISTUNGSKARTE

Modell TI-58 Serien Nr. _____

Modell TI-58C 2304655 Serien Nr. _____

Modell TI-59 Serien Nr. _____

Datum: 22 FEB 1980

1 ☐ Herr 2 ☐ Frau, Fräulein 3 ☐ Firma

Familienname: _____ Vorname: _____

Firma: _____ Strasse: _____

Ort: _____ Postleitzahl: _____ Land: _____

War der Rechner ein Geschenk? 1 ☐ Ja
2 ☐ Nein

Gebrauch: 1 ☐ Privat
3 ☐ Beides 2 ☐ Geschäftlich

Wo gekauft?

- A ☐ Kaufhaus
B ☐ Grosseinkauf
C ☐ Bürofachhändler
D ☐ Rechner-
Spezialgeschäft
E ☐ Versandhandel

- F ☐ Elektrogeschäft
G ☐ Radiogeschäft
H ☐ Fotogeschäft
I ☐ Schreibwarengeschäft
J ☐ Andere (spezifizieren)

Ihr Beruf

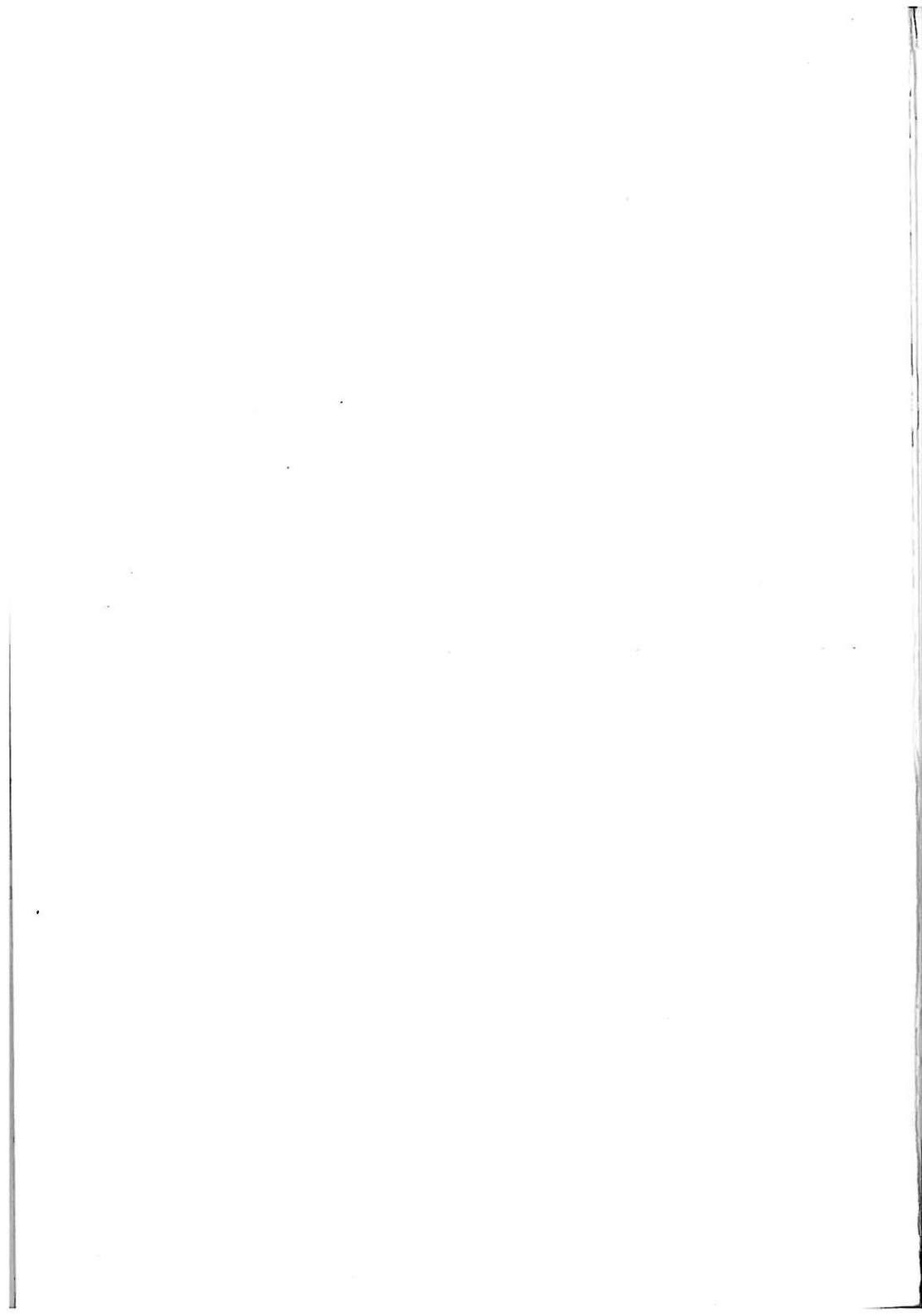
- A ☐ Ingenieur
B ☐ Wissenschaftler
C ☐ Geschäftsmann
D ☐ Buchhalter
E ☐ Schüler
F ☐ Student
G ☐ Lehrer, Dozent
H ☐ Ladeninhaber

- I ☐ Arzt - Rechtsanwalt
J ☐ Techniker
K ☐ Architekt -
Landvermesser
L ☐ Finanzfachmann
M ☐ Andere (spezifizieren)

Nebenbemerkungen: _____

Ihr Alter 1 ☐ -14 2 ☐ 15-18 3 ☐ 19-28 4 ☐ 29-48 5 ☐ 49+





EIN JAHR GEWÄHRLEISTUNG

Texas Instruments gewährleistet dem Endverbrucher (Erstkäufer), daß die elektronischen Taschenrechner TI-58/58C/59 (einschließlich Aufladegerät) von Texas Instruments bei sachgemäßer Wartung und sachgemäßem Gebrauch für die Dauer von einem (1) Jahr ab Kaufdatum frei sind von Herstellungs- und Materialfehlern.

Der Gewährleistungsanspruch besteht nur, wenn :

1. Der Rechner nicht durch Unfall, unsachgemäße Behandlung, Nachlässigkeit, unsachgemäße Wartung oder andere Ursachen, die nicht auf Material- oder Herstellungsfehler zurückzuführen sind, beschädigt wurde.
2. Der Nachweis über das Kaufdatum vom Endverbraucher erbracht ist. **FEHLT DIESER NACHWEIS, WIRD DER ELEKTRONISCHE RECHNER ZU DEN ZUR ZEIT DER REPARATUR GÜLTIGEN SERVICE-PREISEN REPARIERT.**

Während der Gewährleistungszeit wird der mangelhafte Rechner nach Wahl von Texas Instruments kostenlos repariert oder durch einen einwandfreien nachgebesserten Austauschrechner ("RECONDITIONED") entsprechender Qualität und Güte ersetzt, sofern der Rechner portofrei und versichert mit Kaufpreisdatumnachweis an Texas Instruments geschickt wird. Bei berechtigten Gewährleistungsansprüchen kann Erstattung der Versandkosten verlangt werden. (Nur Bundesrepublik Deutschland).

Im Falle der Ersatzlieferung unterliegt der nachgebesserte Austauschrechner bis zum Ablauf der ursprünglichen Gewährleistungsfrist, mindestens jedoch für 90 Tage, den vorstehenden Gewährleistungsbedingungen.

Weitere Ansprüche, insbesondere Ansprüche auf Ersatz von Schaden, die nicht an dem Rechner selbst entstanden sind, sind ausgeschlossen.

WICHTIG ! Bei Rücksendung zwecks Reparatur bitte Versand- und Dienstvorschriften in diesem Buch sorgfältig beachten !

SERVICE CENTERS

BELGIË-BELGIQUE

Mercure Centre
Rue de la Fusée/Raketstraat 100
(Parallèle av. Leopold III)
1130 Brussel/Bruxelles
Tel. (2) 7208000

CANADA

41 Shelley Road
Richmond Hill, Ontario
Tel. (416) 8897373

DANMARK

Marielundvej 46E
2730 Herlev
Tel. (02) 917400

DEUTSCHLAND

Kepserstrasse 33
8050 - Freising
Tel. (08161) 801

ENGLAND

Manton Lane
Bedford, MK41 7PU
Tel. (0234) 67466

ESPAÑA

Carretera Antigua a Barcelona
K. M. 23. 100
Apartado de Correos 98
Torrejon de Ardoz - Madrid
Tel. 6755300 - 6755350

FRANCE

B. P. 28
06021 - Nice Cedex
Tel. (93) 200101

ITALIA

Cassella Postale 1
02015 - Cittaducale
Tel. (746) 69034/35/36

NEDERLAND

Postbus 43
Kolkthofsingel 8
7600 AA Almelo
Tel. (05490) 63967

NORGE

Kr. Augusts gt. 13
Oslo 1
Tel. (02) 206040

ÖSTERREICH

Rennweg 17
1030 - Wien
Tel. (0222) 724186

PORTUGAL

Dept ECD
R. Eng. Frederico Ulrich 2650
Moreira da Maia
4470 - Maia
Tel. (02) 9481003

SCHWEIZ-SUISSE

Aargauerstrasse 250
CH 8048 - Zürich
Tel. (01) 643455/56

SUOMI FINLAND

Elimäenkatu 14-16
P.L. 53
00511 Helsinki 51
Tel. (80) 7013133

SVERIGE

Norra Hamnvägen 3
Fack
10054 Stockholm 39
Tel. (08) 235480

TEXAS INSTRUMENTS