HEWLETT-PACKARD TISCHRECHNER

Modell 9100B

Bedienung und Programmierung

HEWLETT hp PACKARD

GARANTIE

Die Garantie für den Hewlett-Packard Rechner 9100B erstreckt sich auf Material- und Verarbeitungsfehler. Diese Garantie wird für ein Jahr nach Auslieferung gewährt. Dabei werden fehlerhafte Teile instandgesetzt oder ausgetauscht. Weitergehende Ansprüche können nicht geltend gemacht werden, auch haften wir nicht für Folgeschäden. Wegen weiterer Fragen wollen Sie sich bitte an unser nächstes Verkaufsbüro wenden.

P. O. Box 301, Loveland, Colorado, 80537 U. S. A.

Bedienung und Programmierung

HEWLETT-PACKARD 9100B

Einführung

Verwendung dieser Bedienungsanleitung

Dies kann auf zweierlei Art erfolgen:

1. Als Lehrbuch für die Bedienung und Programmierung oder

2. als Nachschlagewerk um die Funktion der einzelnen Tasten und Schalter schneller nachsehen zu können.

Als Textbuch soll sie Leitfaden für den Gebrauch des Rechners sein, ohne Kenntnisse und Erfahrung mit Rechnern vorauszusetzen.

Zum schnellen Nachschlagen sind jede Taste und jeder Schalter auf Seite IV abgebildet und mit einer Index-Ziffer versehen, die die Seite angibt, auf der Taste oder Schalter genau in ihrer Funktion beschrieben sind. Eine Kurzbeschreibung ist in Fettdruck vorangestellt, in Beispielen wird dann die Funktion und der Zweck der Taste bzw. des Schalters erklärt.

Vom 9100A zum 9100B

Für die Benutzer des 9100B, die bisher mit dem Rechner 9100A gearbeitet haben, hier eine kurze Beschreibung der Vorteile des 9100B, die ihn vom 9100A unterscheiden.

1. Der 9100B hat eine größere Speicherkapazität, die in eine (+)–Seite und in eine (–)-Seite aufgeteilt ist (s. Seite 8 u. 57).
   Die (+)–Seite besteht aus 16 Registern mit den Adressen (+) 0 bis (+) f. Diese sind mit den Registern 0 bis f des 9100A identisch.
   Die (–)-Seite besteht ebenfalls aus 16 Registern, von (–) 0 bis (–) f.
   Das Vorzeichen der Seite ist für die Adressierung eines Registers von einer Seite zur anderen und für Verzweigungen im Programm von einer Seite zur anderen unbedingt erforderlich. Das Vorzeichen ist irrelevant für Adressierung und Verzweigungen im Programm, solange die Seite nicht wechselt, also Adresse und Testbefehl auf einer (der gleichen) Seite gespeichert sind (siehe GO-TO-Taste auf Seite 68).

2. Der 9100B hat eine echte Unterprogramm-Kapazität, die es ermöglicht, durch die „SUB/RETURN"-Taste Unterprogramme aufzurufen, und die Rücksprungadresse automatisch zu speichern (s. Seite 60 und 94).

3. Eine zusätzliche Rückruf-Taste „nach x von" ermöglicht den Rückruf von Daten aus den (+)-numerischen und den (–)-alphanumerischen Speichern (s. Seite 38).

4. In Stellung „RUN" des Programmschalters löst die „STEP-PROGRAM"-Taste keine sofortige Verzweigung beim Lesen eines IF-Testes aus, außer der IF-Test hat „Bedingung nicht erfüllt" ergeben (s. Auslesen eines Programms S. 76). In „PROGRAMM"-Stellung erscheint auf der Kathodenstrahlröhre durch Betätigung der „STEP-PROGRAMM"-Taste wie bisher im X-Register die Adresse und der Befehlscode, im Z-Register zusätzlich Adresse und Code des nächsten Programmschrittes.

5. Bedingte Verzweigungen erfolgen wie beim 9100A, jedoch kann das Seitenvorzeichen nicht auf einen IF-Test folgen, außer es ist Bestandteil einer Adresse. „CONTINUE" kann nicht als funktionsloser Befehl nach einem IF-Test programmiert werden, wenn darauf ein alphanumerischer Befehl oder Vorzeichen (+ oder –) folgt.

6. Für das Speichern oder den Rückruf von Daten ist das Seitenvorzeichen nur nötig, wenn der aufgerufene Speicher auf der (–)-Seite steht. Datenrückruf von der (+)-Seite erfolgt also immer ohne Vorzeichen, gleichgültig auf welcher Seite der Befehl steht.

7. Die ACC+ und ACC–Tasten arbeiten in der gleichen Weise wie beim 9100A und speichern den x- und y-Wert in (+)f und (+)e. CLEAR löscht (+)e und (+)f, sowie den ARC- und HYPER-Befehl.

8. END ist nicht unbedingt als letzter Programmschritt erforderlich, wenn das Programm auf Magnetkarte gespeichert werden soll (s. Seite 70).

Seitenindex der Tasten und Schalter

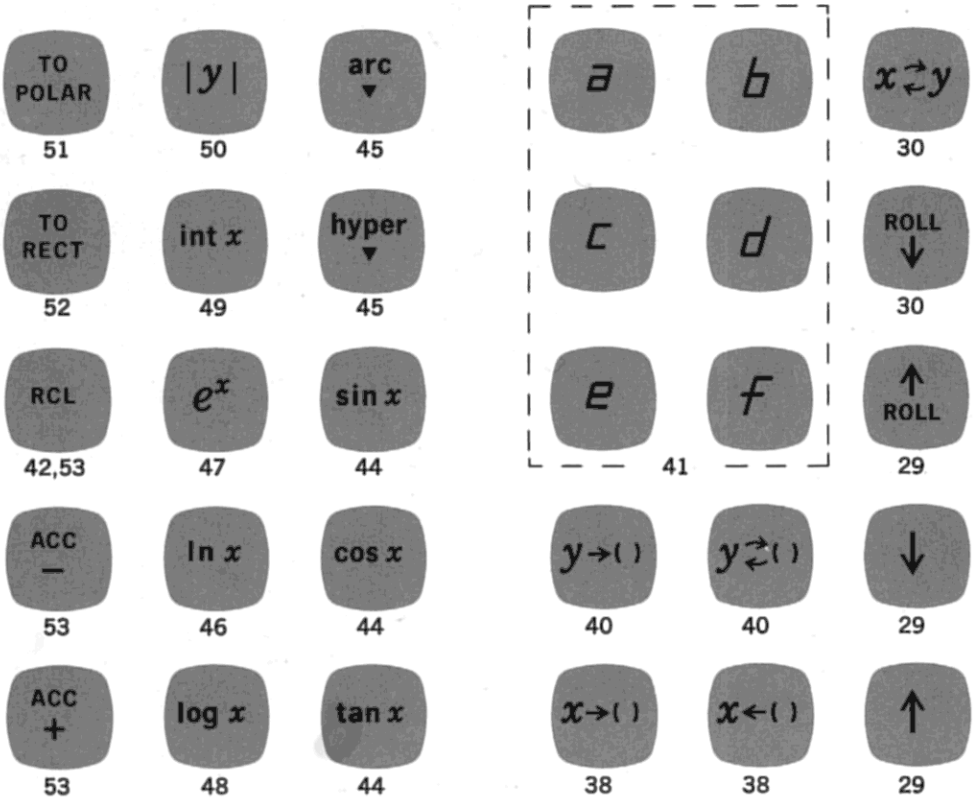
16,21

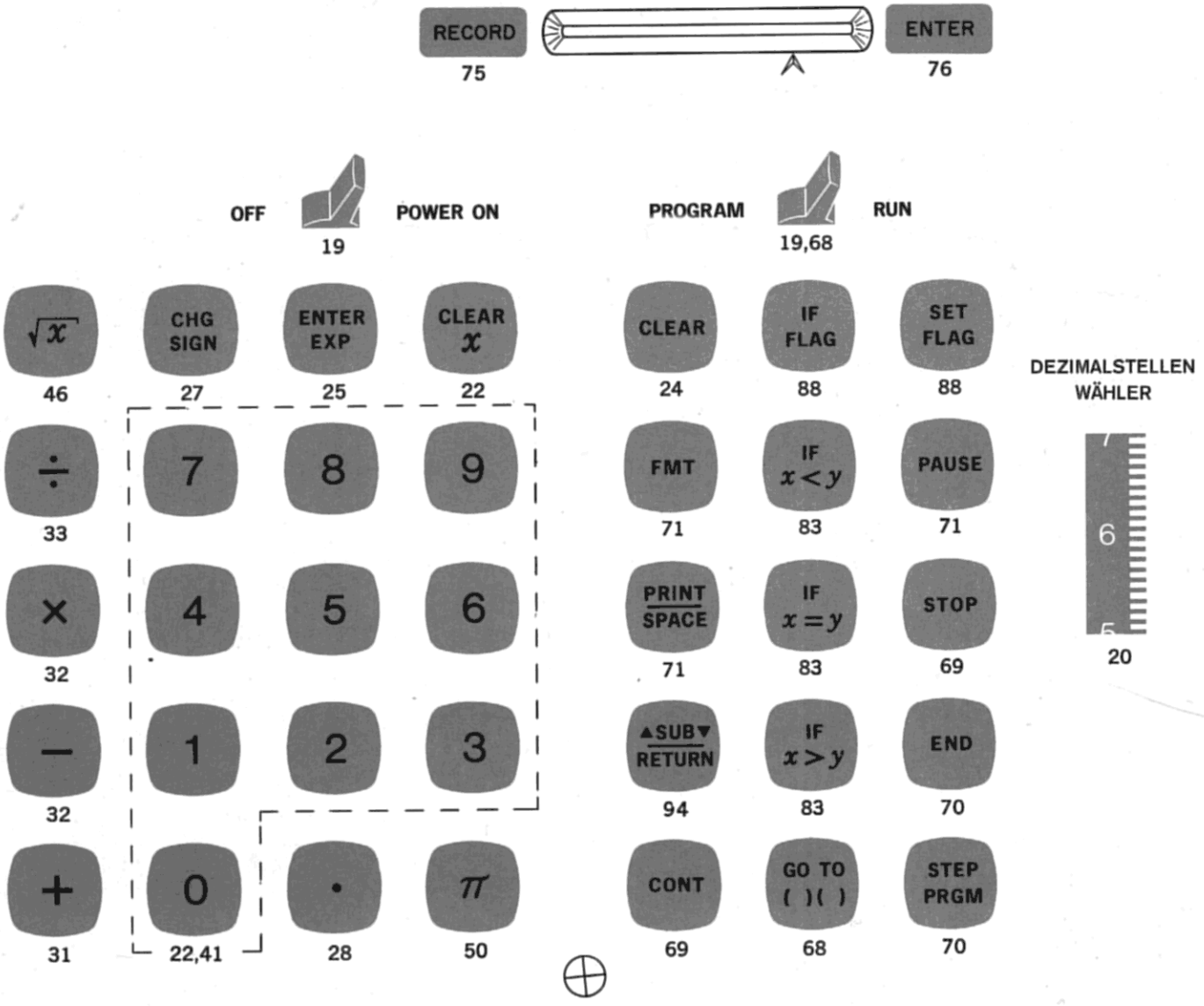
DEGREES RADIANS

19,44

FLOATING FIXED POINT

19





Die Zahlen unter den Tasten und Schaltern sind die Numerierung der Seiten, auf denen die Funktionen beschrieben sind.

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Allgemeine Informationen

Dieser Abschnitt enthält allgemeine Informationen über das Modell 9100B, das Zubehör, die Wartung, usw. Es ist die Inbetriebnahme und die Funktionsprüfung beschrieben.

Beschreibung

Der HP-9100B-Rechner ist ein wissenschaftlicher Kleinrechner. Er ist sowohl für einfache Rechnungen als auch für sehr schwierige mathematische Problemlösungen, wie sie im kommerziellen, schulischen, wissenschaftlichen und technischen Bereich anfallen, geeignet. Trigonometrische, logarithmische und mathematische Funktionen sind durch einen einzigen Tastendruck darzustellen.

Der 9100B ist programmierbar: Der computerähnliche Speicher kann Befehle und Daten für sich wiederholende Rechenoperationen speichern. Bedingte Verzweigungen (der Rechner trifft während des Programmablaufes selbst Entscheidungen) und Unterprogramme ergeben eine nahezu unbegrenzte Programmkapazität. Eine besondere „Maschinensprache" ist zur Programmierung nicht erforderlich.

Die Programme können auf den mitgelieferten Magnetkarten aufgezeichnet werden. Diese können später wieder eingelesen werden und ersparen so das zeitraubende Eintasten von Hand. Unter dem Tastenfeld befindet sich eine herausziehbare Kurzanleitung, auf der die einzelnen Tastenfunktionen erklärt sind.

ACHTUNG
DER KLEINRECHNER 9100B DARF NUR AN DAS NETZ ANGESCHLOSSEN WERDEN, NACHDEM SIE DIE HINWEISE FÜR DIE INBETRIEBNAHME AUF S. 4 GELESEN HABEN!

2 Allgemeine Informationen

Mitgeliefertes Zubehör

In Tafel 1 ist Zubehör und Ausstattung jedes 9100B Modells aufgeführt.

TAFEL 1

MITGELIEFERTES ZUBEHÖR

PART NO.	ANZAHL	BESCHREIBUNG
09100-90021	2	Bedienungs- und Programmieranleitung
09100-90022	1	Programm-Bibliothek
09100-90023	1	Programm-Formulare
09100-90024	1	Magnetkarte mit Diagnostikprogramm
4040-0350	1	Schutzhaube
5060-5919	1	Plastik-Box mit 10 Programm-Karten
8120-0078	1	Netzkabel

Ein Karton mit fünf Programm-Formular-Heften (Part-No. 09100-90020) und div. Mengen Magnetkarten sind erhältlich; Preise auf Anfrage.

Programm-Bibliothek

In der mitgelieferten Programm-Bibliothek (Part-No. 09100-90022) sind für viele Probleme Lösungen programmiert, wie sie in dem großen Bereich der kaufmännischen, wissenschaftlichen und technischen Praxis verwendet werden. Diese Programme sollen Anleitung für die Programmiertechnik und brauchbare Beispiele zugleich sein.

Zeitschrift „Keyboard"

Unsere Kunden erhalten ferner das HP-KEYBOARD, eine Hauszeitschrift, in der wir regelmäßig neue Programme ausdrucken, die wir von Kunden für Kunden erhalten.

Wichtig!

Bitte senden Sie uns die Karte (im Deckel der Programm-Bibliothek hinten) ausgefüllt ein, damit wir Sie in unsere Versandliste für das „Keyboard" aufnehmen können.

Für den Rechner gibt es die herausziehbare Kurzanleitung ohne Aufschlag auch in Landessprache:

9100B Standard: englisch
9100B Opt. 001: französisch
9100B Opt. 002: deutsch
9100B Opt. 003: italienisch
9100B Opt. 004: spanisch

Zusatzgeräte

Folgende Zusatzgeräte (Preise auf Anfrage) sind zur Erweiterung der Verwendungsmöglichkeiten erhältlich (selbstverständlich sind diese Geräte auch für 9100A zu verwenden):

DRUCKER, MODELL 9120A: Wird nur auf den Rechner oben aufgesetzt und mittels Kabel angeschlossen. Er druckt jede Kombination der drei Anzeige-Register und eine Liste des Programms mit Speicherplatz und Code-Ziffer.

X-Y SCHREIBER, MODELL 9125A: Graphische Darstellung der durch den Rechner in(de)krementierten unabhängigen Variablen und der abhängigen Variablen, sowohl manuell als auch durch Programm ansteuerbar.

MONITOR, MODELL 9150A: Großer Bildschirm 19", der den x-, y- und den z-Wert anzeigt. Besonders für Lehrzwecke und große Räume geeignet.

OPTISCHER KARTENLESER, MODELL 9160A: Mit diesem Leser können sowohl Daten als auch Programme über eine Karte (keine Lochkarte), auf der der Code mit Bleistift markiert wurde, in den Rechner eingelesen werden. Besonders für Schulen geeignet, um die Programme der Lernenden schnell zu testen.

Weitere Geräte sind in der Entwicklung und in Zukunft erhältlich (auch mit 9100A verwendbar).

Wartungsvertrag

Für die Folgejahre (2., 3. usw.) können Wartungsverträge abgeschlossen werden. Bitte setzen Sie sich mit IHREM Verkaufsbüro in Verbindung.

Eingangskontrolle

Der Rechner wurde vor dem Versand mechanisch und elektrisch sorgfältig geprüft. Er darf keine äußerlichen Beschädigungen aufweisen. Bitte prüfen Sie das Gerät sofort auf äußerliche Schäden, ebenso das mitgelieferte Zubehör auf Vollständigkeit. Bei Beschädigung des Rechners setzen Sie sich bitte sofort mit dem Frachtführer in Verbindung. Bei elektrischem Defekt verfahren Sie bitte gemäß unseren Garantiebestimmungen auf der Umschlagseite. Bitte die Anleitung für die Inbetriebnahme auf Seite 4 beachten.

4 Allgemeine Information

VORSICHT
ZUR REINIGUNG DER ANZEIGESCHEIBE BITTE NUR EIN WEICHES TUCH BENUTZEN, DA DIE SCHEIBE SONST VERKRATZT WERDEN KÖNNTE!

Netzanschluß

Der Rechner kann mit 115 oder 230 V + 10%, 50 bis 60 Hz und 400 Hz betrieben werden. An der Rückseite befindet sich ein Schiebeschalter, der auf 230 V stehen muß.

Erdung

Zum Schutz des Bedienenden ist das Gehäuse und das Tastenfeld geerdet. Dazu wird das Gerät mit Schuko-Netzkabel geliefert.

ACHTUNG
SCHLIESSEN SIE DEN RECHNER NICHT AN DAS NETZ AN, OHNE SICH ZU ÜBERZEUGEN, DASS DER SCHIEBESCHALTER AUF 230 V STEHT, DA SONST DER TRANSFORMATOR BESCHÄDIGT WERDEN KANN.

Einschalten

Bitte den Rechner erst an das Netz anschließen und dann einschalten, nicht umgekehrt.

Elektrische Prüfung

In der Rückseite der Programm-Bibliothek steckt eine Magnetkarte, auf der das Test-Programm bereits aufgezeichnet ist. Diese Karte prüft den gesamten Rechner elektrisch durch. (Das Programm ist in der Programm-Bibliothek ausgedruckt). Zur Eingabe dieses Programms werden die Schalter oberhalb des Tastenfelds in folgende Stellungen gebracht:

ANMERKUNG

Erfolgt keine Anzeige nach etwa 20 Sekunden:
Flackert die Anzeige:
Arbeiten die Tasten nicht:

Drücken: STOP

Folgende Tasten sind der Reihe nach von links nach rechts zu drücken:

DRÜCKEN: GO TO ( )( ) + 0 0

oder

DRÜCKEN: END

Diagnostik-Karte mit dem A-Pfeil nach unten, bedruckte Seite nach vorn, in den Kartenleser einstecken.

Die Karte muß voll in den Kartenleser bis zum Anschlag eingesteckt werden.

DRÜCKEN: ENTER

(Diese Taste soll solange gedrückt werden, bis die Karte wieder herauskommt. Nicht richtig eingesetzte Karten bleiben stecken).

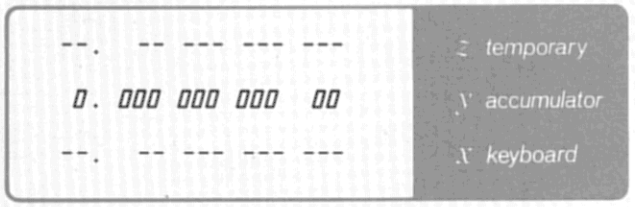
Nun Karte herumdrehen, so daß der B-Pfeil nach unten zeigt und wieder in den Kartenleser einführen.

DRÜCKEN: ENTER

Ist die Karte ganz herausgelaufen, dann

DRÜCKEN: CONT

Die Anzeige muß bei einwandfreiem Arbeiten des Rechners im Abstand von 3 Sekunden jetzt kurzzeitig aufleuchten.
Die erste Anzeige sieht so aus:



Erscheint die Anzeige zum zweiten Mal, sind die Nullen durch Einer ersetzt, dann durch Zweier, Dreier usw. bis Neun. Damit ist der Zyklus durchlaufen und das Programm beginnt von vorn bis „STOP" gedrückt wird.

Zur Funktionsprüfung ist ein Zyklus-Durchlauf ausreichend, der Rechner arbeitet nicht richtig wenn: Keine Anzeige nach 10 s erfolgt, Anzeige erfolgt, aber stehen bleibt, oder die Anzeige entspricht nicht der Abbildung.

Tritt einer dieser Fehler auf, bitte prüfen, ob der Schalter links auf RADIANS steht! Dann Eingabe der Diagnostik-Karte wiederholen.

Wenn das Gerät nicht arbeitet, muß das nächste Kundendienstbüro verständigt werden.

Einführung

Dieser Abschnitt enthält allgemeine Informationen über das Modell 9100B, die für die bestmögliche Ausnutzung der Maschine unerläßlich sind. Ausdrücke werden dort erklärt, wo sie erstmalig benutzt werden. Die Beispiele in diesem Abschnitt sollen nicht das Tastenfeld beschreiben, sondern die im Text behandelten Punkte erläutern, und sie sollen ein „Gefühl" für den Rechner vermitteln.

Merkmale

Der 9100B ist eine äußerst zweckmäßige Maschine mit einem so großen Bereich, daß er sowohl für den Gebrauch als einfache Additionsmaschine als auch für schwierigste wissenschaftliche Berechnungen geeignet ist. Besonders umfangreiche Berechnungen, allerdings mit beschränkten Datenmengen, lassen sich ohne Schwierigkeit durchführen, für die sonst ein Computer nötig wäre.

Durch den Rechenbereich (von 1×10^{-98} bis $9.999\,999\,999 \times 10^{99}$) übertrifft der 9100B viele Computer. Trotzdem ist er einfach zu bedienen.

Auslese-Speicher

Zwei Speichersysteme werden im 9100B verwendet. Das eine ist das ROM (Read- Only- Memory), in dem alle Programme eingespeichert sind, die für die vielen Funktionsberechnungen gebraucht werden. Diese werden dann durch Tastendruck abgerufen. Diese Programme sind fest verdrahtet und können nicht geändert werden. Sie sind Bestandteil der Logik der Maschine.

Der zweite Speicher besteht aus Magnetkernen, die über das Tastenfeld angesteuert werden, und die damit Daten- und Programmierspeicherung ermöglichen.

8 Besonderheit des 9100B

Dieser Speicher besteht aus 35 Registern. Neunzehn sind schematisch in Abb. 1 abgebildet. Die Register erscheinen darauf als horizontale Reihen mit den Bezeichnungen (+) 0 bis (+) f und x, y und z. Dies sind die Register der (+) Seite von (+) 0 bis (+) f, also sechzehn, da x, y und z die Anzeige- und Rechenregister darstellen. Die Register der (-) Seite (nicht abgebildet) haben die Bezeichnung (-) 0 bis (-) f. Es sind demnach 32 Register auf der (+) und (-) Seite für Datenspeicherung oder 28 für Programm und vier für Daten verfügbar, da die Register (+) e, (+) f und (-) e und (-) f nur zur Datenspeicherung zur Verfügung stehen. Der Rechner zeichnet nämlich Programme auf Magnetkarte nur bis (+) dd auf der (+) Seite und (-) dd auf der (-) Seite auf.

Jedes Register besteht aus 14 Speicherplätzen mit den Bezeichnungen 0 bis d (das Vorzeichen ist unerheblich bei der Platzbezeichnung). Jeder Speicherplatz kann einen Programmschritt oder eine Ziffer speichern. Ohne Datenspeicherung können maximal 392 (2 x 14 x 14) Programmschritte eingegeben werden, wobei (+) e, (+) f, (-) e und (-) f noch für Daten freistehen würden.

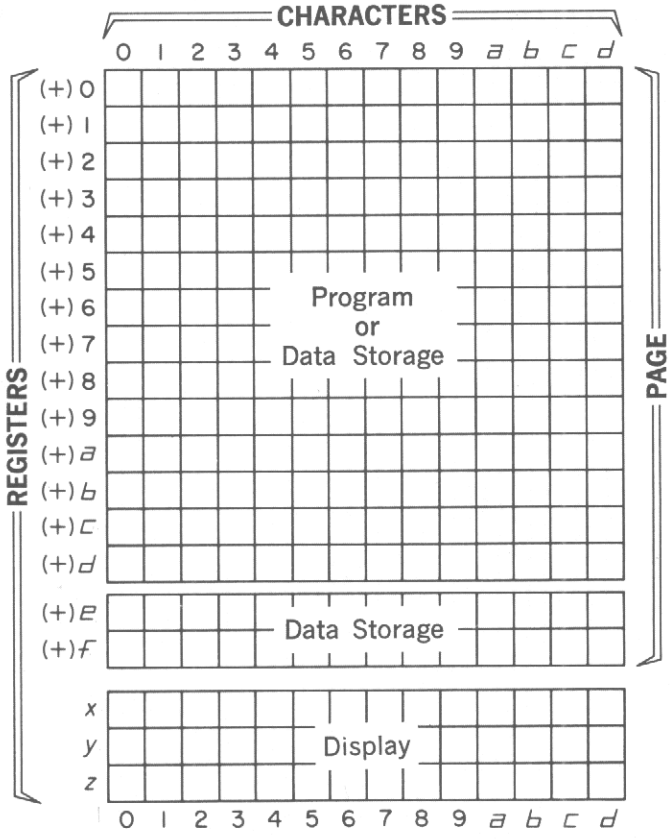


Figure 1. Magnetic-Core Memory

Daten und Programmschritte können nicht gleichzeitig in ein Register gespeichert werden, da jede Zahl ein vollständiges Register belegt (14 Speicherplätze), unabhängig von der Stellenzahl. Durch Speichern einer Zahl vermindert sich die Zahl der möglichen Programmschritte um 14.

x-KEYBOARD - dieses Register zeigt die eingegebene Zahl an, eine Ziffer auf jeden Tastendruck.

Anzeige

ANMERKUNG
Läuft im Rechner ein Programm, muß für die Eingabe von Werten der Programmablauf mit STOP angehalten werden.

BEISPIEL:
Eingabe 6,34 in das X-Register
Schalter oberhalb des Tastenfeldes in Position:

SCHALTEN  **FIXED POINT**

SCHALTEN  **RUN**

DEZIMAL-STELLEN-WÄHLER 

Drücken der Tasten in der Reihenfolge:

DRÜCKEN: CLEAR 6 . 3 4

ANZEIGE: 6.34 → x
(6,34 erscheint im X-Register)

10 Besonderheiten des 9100B

y accumulator das Ergebnis einer arithmetischen Operation zweier Zahlen, von denen eine im X-Register, die andere im Y-Register steht, erscheint im Y-Register.

BEISPIEL:

$6.34 \div 2 = 3.17$

(mit 6.34 → x)

DRÜCKEN: [↑] [2] [÷]

ANZEIGE: 3.17 → y

z temporary für ‚zeitweise' Speicherung; die Zahl wird in Z gespeichert, während im X- und Y-Register eine arithmetische Operation mit zwei anderen Zahlen durchgeführt wird. Darauf kann der Wert von Z nach Y gebracht werden, um wieder in den nächsten Rechenvorgang einbezogen zu werden. Dies erspart Programm- und Datenspeicherung.

BEISPIEL:

$\frac{20}{3+2} = 4$ (20 wird in Z gespeichert, während der Addition von 3 und 2)

DRÜCKEN: [CLEAR] [2] [0] [↑]

ANZEIGE: 20. → y

DRÜCKEN: [3] [↑]

ANZEIGE: 20. → z, 3. → y

DRÜCKEN: [2]

ANZEIGE: 2. → x

x und Y kann jetzt addiert werden ohne Z zu ändern

DRÜCKEN: [+]

ANZEIGE: 5. → y

Z wird nun nach Y gebracht und die Division durchgeführt.

DRÜCKEN: 

ANZEIGE: 20. → y, 5. → x

DRÜCKEN: 

ANZEIGE: 4.00 → y

Gleit- und Festkomma

Ein großer Vorteil des 9100B ist sein Zahlenbereich (1×10^{-98} bis $9.999\,999\,999 \times 10^{99}$), in dem er arbeitet. Die Kommastellen haben immer die gleiche Genauigkeit, gleichgültig in welcher Weise, FEST- ODER GLEITKOMMA, Dezimalstellenwähler auf 0 oder 9, der Rechner arbeitet. Die Darstellung der Anzeige hat auf den Rechenvorgang keinen Einfluß, dieser erfolgt immer in Gleitkomma.

Der Rechner speichert und arbeitet bei allen arithmetischen Operationen in Gleitkomma. Dies ist nichts anderes als eine Darstellungsart, bei der eine Zahl in zwei Teilen, nämlich der Mantisse und dem Exponenten zu 10, geschrieben wird. Die Mantisse besteht aus den Ziffern, wobei der Dezimalpunkt nach der ersten Ziffer steht. Der Exponent zu Zehn kann positiv oder negativ, aber ganzzahlig sein und bestimmt die Stelle, an der der Dezimalpunkt in Festkomma-Darstellung steht.

Nachfolgende Beispiele sollen dies deutlich machen:

		FESTKOMMA		GLEITKOMMA		
				Mantisse		Exponent
BEISPIEL:	(a)	1234.56	=	1.23456	x	10^{3}
	(b)	.00123456	=	1.23456	x	10^{-3}
	(c)	1.23456	=	1.23456	x	10^{0}

Es werden in der Mantisse 10 Ziffern und im Exponenten 2 Ziffern angezeigt, alle Zahlen werden aber mit 12 Stellen in der Mantisse verarbeitet. Diese beiden letzten Stellen werden als verdeckte Stellen bezeichnet (guard digits) und haben

den Zweck, die Genauigkeit über die zehn angezeigten Stellen hinaus zu erhöhen und in Festkommadarstellung die letzte der zehn Stellen zu runden.

Das folgende Beispiel verdeutlicht dies:

dividieren von 6.000 000 001 durch 3 und dann das Ergebnis mit 3 multiplizieren.

SCHALTEN:  **FIXED POINT**

SCHALTEN:  **RUN**

DEZIMAL-STELLEN-WÄHLER: 

DRÜCKEN: CLEAR 6 .

DRÜCKEN: 0 0 0

DRÜCKEN: 0 0 0

DRÜCKEN: 0 0 1

ANZEIGE: 6.000000001 → x

DRÜCKEN: ↑ 3 ÷

ANZEIGE: 2.000000000 → y

ohne die 11. und 12. Stelle würde die Anzeige nach der Multiplikation 6.000 000 000 und nicht 6.000 000 001, die ursprüngliche Eingabe anzeigen.

DRÜCKEN: ×

ANZEIGE: 6.000000001 → y

Die Genauigkeit bleibt erhalten.

Es ist einfach, die verdeckten Stellen sichtbar zu machen.

BEISPIEL:

Durch Drücken der π-Taste wird diese Zahl mit 12 Stellen in das X-Register eingelesen, zehn Stellen werden dargestellt. Um die beiden verdeckten Stellen anzuzeigen, wird folgendermaßen verfahren:

SCHALTEN: **FLOATING**

SCHALTEN: **RUN**

DRÜCKEN: π ↑

ANZEIGE: 3.141 592 653 00 → y

Exponent

Jetzt werden die beiden ersten Ziffern von π abgezogen:

DRÜCKEN: 3 . 1 −

ANZEIGE: 4.159 265 360 -02 → y

verdeckte Stellen

Ist eine Zahl für eine bestimmte Einstellung des Dezimalstellenwählers zu groß, so springt die Anzeige automatisch auf Gleitkomma.

BEISPIEL:

SCHALTEN: **FIXED POINT**

SCHALTEN: **RUN**

DEZIMAL-STELLEN-WÄHLER: 7

14 Besonderheiten des 9100B

DRÜCKEN: CLEAR 1 2 3 •

DRÜCKEN: 4 5 6 7

DRÜCKEN: 8 9 8

ANZEIGE: 123.4567898 → x

Bei Dezimalstellenwähler auf 7 stehen 7 Dezimalstellen rechts vom Komma und 3 links.

DEZIMAL-STELLEN-WÄHLER: 8

ANZEIGE: 1.234 567 898 02 → x

Hierbei kann der Rechner nur 2 Stellen vor dem Komma darstellen, die Zahl ist für Festkomma zu groß und die Anzeige springt auf Gleitkomma, wobei keine Ziffern verloren werden.

Ist die Zahl zu klein für Festkommadarstellung, springt die Maschine nicht automatisch auf Gleitkomma. Auf dem Bildschirm erscheinen Nullen, aber die Zahl ist immer noch im Rechner in Gleitkomma gespeichert. Hierfür als Beispiel die Eingabe von 4×10^{-6} (0.000004) in den Rechner:

FIXED POINT

DEZIMAL-STELLEN-WÄHLER: 6

CLEAR x • 0 0 0

DRÜCKEN: 0 0 4

ANZEIGE: .000004 → x

DEZIMAL-
STELLEN-
WÄHLER: 

ANZEIGE: .00000 → x

die Zahl ‚unterläuft' und Nullen stehen im X-Register.

SCHALTEN: **FLOATING** 

ANZEIGE: 4. -06 → x

die Genauigkeit bleibt erhalten, obwohl die Anzeige nicht nach Gleitkomma wechselt. In beiden Fällen, „Überlauf" und „Unterlauf" verliert der Rechner keine Information, da nur in Gleitkomma gerechnet wird, (unabhängig von der Anzeigeart).

Bereich

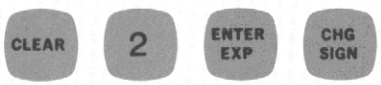
Der Rechenbereich des 9100B (9100A ebenfalls ist $9.999\,999\,999 \times 10^{99}$ bis 1×10^{-98}. Am folgenden Beispiel soll dies erklärt werden, obwohl in den Rechner $9.999\,999\,999 \times 10^{-99}$ eingegeben und gespeichert werden kann:

BEISPIEL:

$$(2 \times 10^{-50}) \times (5 \times 10^{-50}) =$$
$$10 \times 10^{-100} =$$
$$1 \times 10^{-99}$$

SCHALTEN: **FLOATING** 

SCHALTEN:  **RUN**

DRÜCKEN: 

DRÜCKEN: 

DRÜCKEN: 

DRÜCKEN: 

(das Y-Register „läuft über" und zeigt Null, in diesem Fall ist die Information verloren.)

16 Besonderheiten des 9100B

Unerlaubte Rechenoperationen

Wird eine mathematisch unerlaubte Rechenoperation, wie Division durch 0, durchgeführt, leuchtet auf der linken Seite neben der Anzeige ein rotes Licht auf. Läuft ein Programm im Rechner, in dessen Verlauf eine unerlaubte Rechenoperation durchgeführt wird, bleibt die Fehleranzeige brennen, aber das Programm läuft weiter. Die Fehleranzeige kann durch Drükken einer beliebigen Taste gelöscht werden. Verzeichnis der unerlaubten Rechenoperationen auf Seite 21.

BEISPIEL:
Division durch 0 (null)

SCHALTEN: **RUN**

DRÜCKEN: CLEAR 1 ↑ 0 ÷

Die Fehleranzeige leuchtet auf.

Die Rechnung wird aber durchgeführt, wobei das Ergebnis $9.999\,999\,999 \times 10^{99}$ (für den Rechner als unendlich interpretiert) im Y-Register erscheint.

DRÜCKEN: Eine beliebige Taste

Die Fehleranzeige erlischt.

ANMERKUNG
Es gibt dabei (scheinbare) Ausnahmen: Wenn eine Taste gedrückt wird, die wiederum eine unerlaubte Rechenoperation durchführt, verlöscht die Lampe, wird aber sofort durch die neue unerlaubte Operation wieder eingeschaltet.

Einführung

Dieser Abschnitt beschreibt die Funktionen aller Schalter und Tasten, mit Ausnahme derjenigen, die unter „Programmierung" erklärt werden. Jede Taste ist ausführlich dargestellt, indem eine Kurzbeschreibung in Fettdruck als Schnell-Information und eine ausführlichere Erläuterung mit Beispielen die Funktion erklärt. Die Abbildung des Tastenfeldes auf Seite IV enthält die Seitenangabe für die Kurzbeschreibung.

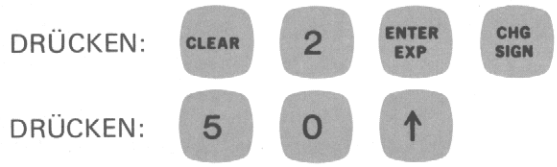
Sparen von Programmschritten

Wenn immer möglich, sollte die kürzeste Methode für die Lösung eines Problems angewandt werden, um die Zahl der Programmschritte zu vermindern. Das spart Zeit und ist besonders dann von Nutzen, wenn die Zahl der Programmschritte begrenzt ist, etwa durch notwendige Datenspeicherung. Einige Tricks nachfolgend.

Darstellung der Schrittfolge

In den Beispielen sind zwei verschiedene Arten der Darstellung verwandt, je nach Zweckmäßigkeit für das einzelne Beispiel:

ART (a)



wobei die Tasten von links nach rechts der Reihe nach gedrückt werden sollen, Zeile 1, dann Zeile 2 usw.

ART (b)

SCHRITT	TASTE
1	CLEAR
2	2
3	ENTER EXP
4	CHG SIGN

SCHRITT	TASTE
5	5
6	0
7	↑

wobei die Tasten in der Folge der Schrittnumerierung 1, 2, 3, 4, usw. zu drücken sind. Dies ist ein vereinfachter Weg des Programmschreibens.

Darstellung der Schalter

Die Schalterstellung bleibt wie bisher, zum Beispiel

SCHALTEN:  **RUN**

bedeutet „Schalten des Programm-Run-Schalters in RUN-Position."

> **ANMERKUNG**
> Ist kein Schalter abgebildet, kann dieser in jeder Stellung sein. In diesem Abschnitt bedeutet dies, daß der Programm-Run-Schalter immer auf RUN steht.

Der Dezimalstellenwähler wird wie folgt angegeben:

DREHEN: DEZIMALSTELLENWÄHLER AUF 5

Andere Darstellungen

Angaben wie „9,00 erscheint im Y-Register" ist so dargestellt:

ANZEIGE: 9.00 → y

„EINGABE" entspricht „Einlesen", wenn nicht anders angegeben.

BEISPIEL:

„EINGABE eines Programmes" bedeutet „Einlesen eines Programmes".

OFF  **POWER ON**

Schaltet die Stromversorgung des Rechners ein.
Die Bezeichnung der Register rechts von der Anzeigeröhre wird beleuchtet; damit ist der Rechner eingeschaltet.

> **ANMERKUNG**
> Bitte die Hinweise für die Stromversorgung und das Einschalten auf Seite 4 beachten.

DEGREES  **RADIANS**

Vorwahl der Dimension, Winkel- oder Bogenmaß, für das Rechnen mit trigonometrischen Funktionen. Winkel werden in der eingegebenen Dimension dargestellt. Umschalten von Winkelmaß (Bogenmaß) auf Bogenmaß (Winkelmaß) bedeutet keine automatische Umrechnung. Für Umrechnung siehe Seite 44.

PROGRAM  **RUN**

Anwahl der Arbeitsweise

PROGRAMM: Für die Eingabe eines Programms über das Tastenfeld und Auslesen eines Programms. (Im Abschnitt Programmierung erklärt.)

RUN: Für das Rechnen von Hand, Eingabe eines Programms über Magnetkarte, Rechnen per Programm und Adressierung des Programm-Zählers. (Die Programmfunktionen sind unter Programmierung erläutert.)

FLOATING  **FIXED POINT**

Anwahl der Darstellung (siehe Seite 11)

GLEITKOMMA: Zahlen bis zu zehn Stellen und zwei Stellen als Exponent zu Zehn werden angezeigt (als Multiplikator). Nullen ohne Funktion erscheinen nicht.

BEISPIEL:

ZAHL

$12{,}345.67898 = 1.234567898 \times 10^{4}$

ANZEIGE

1.234 567 898 04

FLOATING FIXED POINT

FESTKOMMA: Die Zahl wird in der üblichen Schreibweise dargestellt, ohne Exponent mit dem Komma an der richtigen Stelle. Die letzte Stelle wird automatisch gerundet.

Ist die darzustellende Zahl links vom Komma zu groß, so springt die Anzeige automatisch auf Gleitkomma-Darstellung.

Bei ‚Unterlauf' springt die Anzeige nicht auf Gleitkomma um.

DECIMAL DIGITS

BEISPIEL:

(s. Tasten- und Schalter Hinweise Seite 17)

SCHALTEN: **FIXED POINT**

SCHRITT	TASTE
1	CLEAR
2	1
3	2
4	3
5	4
6	5

SCHRITT	TASTE
7	•
8	6
9	7
10	8
11	9
12	8

DEZIMALSTELLENWÄHLER ANZEIGE:

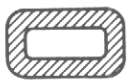
DREHEN: auf 5 — 12345.67898 → x

DREHEN: auf 3 — _12345.679 → x

Keine Nullstellen — Letzte Stelle gerundet

DREHEN: auf 6 — 1.234 567 898 04 → x

Zahl für 6 Dezimalstellen zu groß, schaltet automatisch auf Gleitkomma um.



FEHLERLAMPE, links von der Kathodenstrahlröhre, leuchtet bei unerlaubter Rechenoperation, die sowohl über das Tastenfeld von Hand als auch vom Programmablauf eingegeben werden.

Eine unerlaubte Rechenoperation hält den Programmablauf nicht an, jedoch bleibt die Fehlerlampe brennen.

Drücken einer beliebigen Taste bringt die Lampe zum Erlöschen (s. Seite 16). Unerlaubte Rechenoperationen sind:

Division durch 0
$\sqrt{x}$ wobei x <0
ln x wobei x ≤ 0; log x wobei x ≤ 0
arc sin x wobei $|x| > 1$; arc cos x wobei $|x| > 1$
arc cosh x wobei $x < 1$; arc tanh x wobei $|x| > 1$

0
BIS
9

Die Zifferntasten 0 bis 9 werden für die Eingabe von Zahlen in das x-Register verwendet. Die Ziffern werden hintereinander eingegeben, wobei die letzte eingegebene Ziffer auch als letzte angezeigt wird.
In Verbindung mit den folgenden Tasten sind diese Adressierung des Magnetkern-Speichers.

GO TO ( ) ( ), $x\rightarrow( )$, $x\leftarrow( )$, $y\rightarrow( )$ and $y\rightleftarrows( )$ (diese sind unter den Tasten erklärt).

BEISPIEL:

Eingabe 12345.06789

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

DRÜCKEN: CLEAR 1 2 3

DRÜCKEN: 4 5 . 0

DRÜCKEN: 6 7 8 9

ANZEIGE: 12345.06789 → x

CLEAR x

Löscht nur das x-Register (0. → X).
Löscht die ARC und HYPER Befehle.
Nicht nötig vor jeder Eingabe zu drücken!

CLEAR X wird gedrückt, wenn eine Zahl noch nicht zu Ende eingegeben ist, jedoch gelöscht werden soll, ohne daß die beiden anderen Anzeigeregister betroffen werden.

Die Eingabe einer Zahl gilt als beendet, wenn eine mathematische Operation erfolgt. In diesem Fall überschreibt die nächste eingegebene Zahl automatisch den alten Wert des X-Registers. CLEAR X wird nicht benötigt.

Die Eingabe einer Zahl gilt als nicht beendet, wenn die letzte gedrückte Taste eine Zahl, Komma, CHG SIGN oder ENT EXP war.

In diesem Fall überschreibt die nächste Ziffer nicht, sondern wird Bestandteil der bereits angezeigten Zahl, es muß also CLEAR X gedrückt werden.

ANMERKUNG

Viele Beispiele im Text enthalten ein CLEAR X nur, weil die vorausgegangenen Beispiele eine unbeendete Zahlenfolge im X-Register zurückgelassen haben.

Das folgende Beispiel soll drei Dinge klären:

den Gebrauch der CLEAR X-Taste,
den Unterschied zwischen einer beendeten und einer unbeendeten Zahleneingabe
einen zeitsparenden, wertvollen Trick aufzeigen.

Eine lange Zahlenreihe soll addiert werden; nach jeder Addition steht das Ergebnis im Y-Register und die letzte eingegebene Zahl in X. Die Hälfte der Zahlenreihe ist addiert als der Operator abgerufen wird. Bei seiner Rückkehr kann er sich nicht mehr erinnern, ob die Zahl im X-Register bereits hinzugezählt ist oder nicht. Das Beispiel zeigt, wie dies durch CLEAR X festgelegt werden kann, ohne die ganze Reihe neu rechnen zu müssen.

BEISPIEL:

Addition **7, 32, 4, -, -,** usw.

Unterbrechung

(angenommen die Summe vor 7 ergibt 465)

SCHRITT	TASTE
1	CLEAR
2	4
3	5
4	6
5	↑

456 → y
(Gesamtsumme)

dies ergibt den beschriebenen Sachverhalt, die nächsten zu addierenden Zahlen sind 7 und 32:

SCHRITT	TASTE
6	7
7	+
8	3
9	2

7 → x,
7 → x, 463 → y
32 → x, 463 → y

der Operator wurde unterbrochen und weiß nicht, ob + gedrückt wurde oder nicht. Um das festzustellen

SCHRITT	TASTE
10	1

321 → x, 463 → y

CLEAR x

CONTINUED

1 steht nicht an Stelle von 32, offensichtlich war also 32 nicht als beendet interpretiert, die +-Taste nicht gedrückt. Berichtigung und Fortsetzung der Zahlenreihe:

SCHRITT	TASTE	
11	CLEAR x	0. → x, 463 → y
12	3	
13	2	32 → x, 463 → y
14	+	32 → x, 495 → y
15	4	
16	+	4 → x, 499 → y
etc.	...	

Falls die +-Taste gedrückt war:

SCHRITT	TASTE	
1 – 9 wie vorher		32 → x, 463 → y
10	+	32 → x, 495 → y

war die +-Taste gedrückt?

SCHRITT	TASTE	
11	1	1 → x, 495 → y

Die 1 steht an Stelle der 32, die Eingabe von 32 war beendet, dies besagt, daß die +-Taste gedrückt war. Berichtigung und Fortsetzung der Zahlenreihe:

SCHRITT	TASTE	
12	CLEAR x	0 → x, 495 → y
13	4	
etc.	...	

CLEAR

CLEAR

CLEAR löscht nicht den Inhalt der internen Kernspeicher, löscht nur die Anzeigeregister (0 ➤ X, Y und Z).

Löscht den Inhalt des (+)e und (+)f-Registers im Kernspeicher (dies ist bei der Verwendung der Acc+und Acc - Tasten erklärt).
Löscht die SET-FLAG-Bedingung (unter Programmierung erklärt).
Löscht die ARC und HYPER Bedingungen.

Es ist nicht nötig, immer die CLEAR-Taste zu drücken, bevor eine neue Rechenoperation begonnen wird: Ausnahme bei unbeendeter Eingabe (siehe CLEAR X) und Benutzung der ACC+ und ACC−-Tasten. Neue Werte ersetzen die alten.

CLEAR ist jedoch ein nützlicher Programmbefehl, wenn in der Anzeige und im (+)e und (+)f-Register Werte nicht mehr benötigt werden, oder zu falschen Ergebnissen führen.

ENTER EXPONENT löscht den Exponenten im X-Register und bewirkt, daß die nächst eingegebenen Werte und CHG SIGN nur als Exponent gelesen werden. Die Eingabe der Exponenten erfolgt hintereinander, jede neu eingegebene Zahl erscheint an der letzten Stelle des Exponenten.

Durch Drücken der Taste Dezimalpunkt nach ENT EXP wird der ENT-EXP-Befehl gelöscht (s. Beschreibung der Dezimalpunkt-Taste). Drücken von ENT EXP nach einer Tastenfeldoperation außer Zahleneingabe liest in das X-Register eine 1 ein.

BEISPIEL:

SCHALTEN: **FLOATING**

DREHEN: DEZIMALSTELLENWÄHLER auf 3

(a) Eingabe 1×10^0 (1.0)

DRÜCKEN: CLEAR x ENTER EXP

ANZEIGE: 1.000 000 000 00 → x

(es ist nicht nötig, 1 oder Komma zu drücken)

Wiederholung mit: **FIXED POINT**

ANZEIGE: 1.000 → x

(b) Eingabe 1×10^3 (1000)

SCHALTEN: **FLOATING**

DRÜCKEN: CLEAR x ENTER EXP 3

ANZEIGE: 1.000 000 000 03 → x

CONTINUED

(die 1 und das Komma wurden nicht benutzt)

Wiederholen mit: **FIXED POINT**

ANZEIGE: 1000.000 → x

(c) Eingabe 1.2678 x 10^2 (126.78)

SCHALTEN: **FLOATING**

DRÜCKEN: CLEAR x | 1 | 2 | 6

DRÜCKEN: . | 7 | 8

ANZEIGE: 1.2678 02 → x

alternativ,

DRÜCKEN: CLEAR x | 1 | 2 | 6

DRÜCKEN: 7 | 8 | ENTER EXP | 2

ANZEIGE: 1.2678 02 → x

Zahlen kleiner als 1×10^{-9} müssen mittels ENT EXP eingegeben werden.

BEISPIEL:

Eingabe 2×10^{-10} (.000 000 000 2)

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 9

(a) falsche Eingabe:

DRÜCKEN: CLEAR x | . | 0 | 0 | 0

DRÜCKEN: 0 | 0 | 0

DRÜCKEN: 0 0 0 2

ANZEIGE: 2. → x

SCHALTEN: **FLOATING**

ANZEIGE: 2. -00 → x

Die gewünschte Zahl wurde nicht eingelesen.

(b) richtige Eingabe:

DRÜCKEN: CLEAR x 2

DRÜCKEN: ENTER EXP CHG SIGN 1 0

ANZEIGE: 2. -10 → x

CHG SIGN

CHANGE SIGN ändert das Vorzeichen des Wertes im X-Register. War ENT EXP gedrückt, wird das Vorzeichen des Exponenten geändert.

CHG SIGN

BEISPIEL:

Eingabe -4.9×10^{-3} (–.0049)

SCHALTEN: **FLOATING**

DREHEN: DEZIMALSTELLENWÄHLER auf 4

DRÜCKEN: CLEAR x 4 9 CHG SIGN

DRÜCKEN: ENTER EXP CHG SIGN 3

ANZEIGE: -4.9 -03 → x

Wiederholung mit: **FIXED POINT**

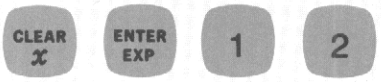
ANZEIGE: -.0049 → x

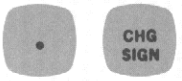
Eingabe des Kommas.
Es ist nicht nötig, diese Taste bei Ganzzahlen oder ENT EXP zu benutzen. Löscht den ENT EXP Befehl, so daß eine folgende Zahlentaste oder CHG SIGN die Mantisse und nicht den Exponenten verändert.

BEISPIEL:

Eingabe -1×10^{12}

SCHALTEN: 

DRÜCKEN: 

DRÜCKEN: 

ANZEIGE: -1.000 000 000 12 → x

normalerweise wird 1×10^{12} eingegeben:

DRÜCKEN: 

Die Möglichkeit, mittels des Kommas den ENT EXP Befehl zu löschen, kann zur Berichtigung eines Fehlers bei der Eingabe ohne CLEAR X verwendet werden.

BEISPIEL:

-4.5×10^{-2} ist verlangt, aber
$4. \times 10^{-2}$ ist falsch eingegeben.

SCHALTEN: **FLOATING** 

DRÜCKEN: 

ANZEIGE: 4. -02 → x

zur Berichtigung ohne CLEAR X:

DRÜCKEN: 

ANZEIGE: -4.5 -02 → x

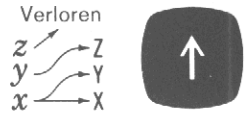
SCHALTEN:  **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 3

Wiederholung des obigen Beispiels.

ANZEIGE: -.045 → x

Die fünf Kontrolltasten (↑, ↓, ROLL↑, ROLL↓, $x \rightleftarrows y$) sind für die Positionierung der Inhalte der Anzeige-Register zu verwenden.



↑ **Doppelt den Wert des X-Registers in das Y-Register. Schiebt den Wert des Y-Registers in das Z-Register. Der Inhalt des Z-Registers ist verloren.**

BEISPIEL:

SCHALTEN: **FIXED POINT**

DRÜCKEN: 4 ↑ 3 ↑

ANZEIGE:
4. → z
3. → y
3. → x

↓ **Doppelt den Inhalt des Z-Registers in das Y-Register. Schiebt den Inhalt des Y-Registers in das X-Register. Der Inhalt des X-Registers ist verloren.**

BEISPIEL:

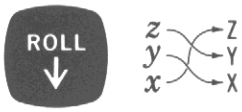
SCHALTEN: **FIXED POINT**

DRÜCKEN: 6 ↑ 5 ↑ ↓

ANZEIGE:
6. → z
6. → y
5. → x

↑ ROLL **Zyklisch vertauschen: Rollt die Registerinhalte in Pfeilrichtung ohne Werte zu verlieren.**
Schiebt den Inhalt des X-Registers in das Y-Register.
Schiebt den Inhalt des Y-Registers in das Z-Register.
Schiebt den Inhalt des Z-Registers in das X-Register.

Trick: Roll ↓ hat die gleiche Wirkung wie zweimaliges Roll↑.



Rollt die Registerinhalte in Pfeilrichtung ohne Werte zu verlieren.
Schiebt den Inhalt des Z-Registers in das Y-Register.
Schiebt den Inhalt des Y-Registers in das X-Register.
Schiebt den Inhalt des X-Registers in das Z-Register.

Trick: Roll ↑ hat die gleiche Wirkung wie 2 maliges Roll ↓

BEISPIEL:

SCHALTEN: **FIXED POINT**

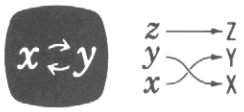
DRÜCKEN: CLEAR 9 ↑ROLL π

DRÜCKEN: ↑ROLL ↑ROLL

ANZEIGE: 3.142 → z
0. → y
9. → x

Durch Roll ↓ können Programmschritte gespart werden.

DRÜCKEN: CLEAR π ROLL↓ 9



Austausch der Inhalte des X- und Y-Registers. Der Inhalt des Z-Registers bleibt unverändert.

BEISPIEL:

$$\frac{20}{3+2} = 4$$

Dieses Beispiel wurde auf Seite 10 erläutert, um das vorübergehende Speichern im Z-Register zu zeigen. Abzüglich des CLEAR-Befehls (der nicht immer benötigt wird) waren neun Programmschritte erforderlich. Mit **x⇄y** werden nur acht Programmschritte benötigt.

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 2

DRÜCKEN: 3 ↑ 2 +

DRÜCKEN: 2 0 x⇄y ÷

ANZEIGE: 4.00 → y

Die vier arithmetischen Tasten +, –, x und ÷ arbeiten mit den Zahlen im X- und Y-Register, wobei das Ergebnis in Y erscheint. Der Wert des X-Registers bleibt unverändert. Der Wert des Z-Registers bleibt ebenfalls unverändert. Rechenoperationen mit diesen vier Tasten sind anschließend an die Erklärung der Divisionstaste dargestellt, ebenso programmsparende Tricks.

Addiert den Wert des X-Registers zu dem Wert des Y-Registers. Die Summe erscheint in Y; die Werte des X- und Z-Registers bleiben unverändert.

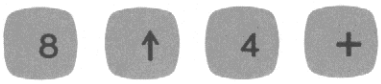
BEISPIEL:

SCHALTEN:  **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

(a) $8 + 4 = 12$

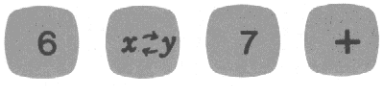
(ist der Inhalt des X-Registers unbeendet, CLEAR X drücken.)

DRÜCKEN: 

ANZEIGE: Ursprünglicher Inhalt von Y → z
12. → y
4. → x

(b) $6 + 7 + 5 = 18$

(der Inhalt von Z bleibt während der Rechnung unverändert.)

DRÜCKEN: 

DRÜCKEN: 

ANZEIGE: Ursprünglicher Inhalt von Z → z
18. → y
5. → x

Zieht den Wert des X-Registers vom Wert des Y-Registers ab. Die Differenz erscheint im Y-Register. Die Werte des X- und Y-Registers bleiben unverändert.

BEISPIEL:

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

(a) $17 - 9 = 8$

DRÜCKEN: 1 7 ↑ 9 −

ANZEIGE: Ursprünglicher Inhalt von Y → z
8. → y
9. → x

(b) $47 - 19 - 8 + 5 = 25$

(der Inhalt von Z bleibt während der Rechnung unverändert)

DRÜCKEN: 4 7 $x \rightleftarrows y$ 1 9

DRÜCKEN: − 8 − 5 +

ANZEIGE: Ursprünglicher Inhalt von Z → z
25. → y
5. → x

Multipliziert den Wert des Y-Registers mit dem Wert des X-Registers. Das Produkt erscheint im Y-Register. Der Wert des Z-Registers bleibt unverändert.

BEISPIEL:

SCHALTEN: **FIXED POINT**

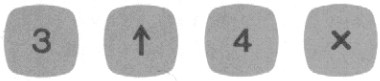
DREHEN: DEZIMALSTELLENWÄHLER auf 5

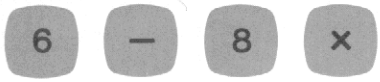
(a) $9 \times 7 = 63$

DRÜCKEN: 9 ↑ 7 ×

ANZEIGE: Ursprünglicher Wert von Y → z
63. → y
7. → x

(b) [(3 X 4) − 6] X 8 = 48

DRÜCKEN: 

DRÜCKEN: 

ANZEIGE: Ursprünglicher Inhalt von Y → *z*
48. → *y*
8. → *x*

(c) $5^2 = 25$

DRÜCKEN: 

ANZEIGE: *25.* → *y*

(d) $5^4 = 625$

(siehe auch ln x und e^x-Taste)

DRÜCKEN: 

ANZEIGE: *625.* → *y*

Dividiert den Wert des Y-Registers durch den Wert des X-Registers! Der Quotient erscheint im Y-Register. Die Werte des X- und Z-Registers bleiben unverändert!

BEISPIEL:

36 ÷ 9 = 4

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

DRÜCKEN: 

ANZEIGE: Ursprünglicher Inhalt von Y → *z*
4.00000 → *y*
9. → *x*

Das folgende Beispiel beinhaltet den Gebrauch aller vier arithmetischen Tasten; die Rechenoperationen wurden auf zwei Arten durchgeführt, um zu zeigen, wie Programmschritte eingespart werden können.

BEISPIEL:

$$\frac{(3 \times 4) + (8 - 9)}{(8 \times 2) - 6} = 1.1$$

Methode I:
(a) Beginnend mit dem Zähler, werden die Klammern ausgerechnet, dann addiert und das Ergebnis gespeichert;
(b) ebenso wird der Nenner ausgerechnet;
(c) Rückruf des Zählers und Division.

SCHALTEN: **FIXED POINT**

DREHEN: STELLENWÄHLER auf 5

Schritt	Taste	Anzeigeregister X	Y	Z	Bemerkung
1	CLEAR	0	0	0	- -
2	3	3	0	0	- -
3	↑	3	3	0	- -
4	4	4	3	0	- -
5	×	4	12	0	(3x4) → Y
6	ROLL ↑	0	4	12	Speichern in Z
7	8	8	4	12	- -
8	$x \rightleftarrows y$	4	8	12	- -
9	9	9	8	12	- -
10	−	9	−1	12	(8 − 9) → Y
11	↓	−1	12	12	Rückruf (3 x 4)
12	+	−1	11	12	(3x4)+(8−9) → Y
13	ROLL ↑	12	−1	11	Speichern in Z
14	8	8	−1	11	- -
15	$x \rightleftarrows y$	−1	8	11	- -
16	2	2	8	11	- -
17	×	2	16	11	(8 x 2) → Y
18	6	6	16	11	- -
19	−	6	10	11	(8x2) −6 → Y
20	ROLL ↓	10	11	6	Rückruf (3 x 4)
21	÷	10	1.1	6	$\frac{(3x4) + (8-9)}{(8x2) - 6} = 1.1 \rightarrow Y$

Dieses Programm kann ohne Änderung des Rechenablaufs um einige Programmschritte gekürzt werden:

1. Die CLEAR-Taste ist in den meisten Fällen ein unnötiger Programmschritt;
2. Schritt 6 ist nicht nötig, wenn Schritt 8 in ↑ geändert wird:

Schritt	Taste	X	Y	Z
5	×	4	12	0
6				
7	8	8	12	0
8	↑	8	8	12
9	9	9	8	12

Der Inhalt der Register ist bei Schritt 9 der gleiche und ein Schritt wurde gespart. Ähnlich kann Schritt 13 ausgelassen und Schritt 15 in ↑ geändert werden.

Methode II:

Durch Einzel-Addition von (+) 8 und Einzelsubtraktion von (-9) im Zähler können mehr Schritte gespart werden. Diese Lösung reduziert die Zahl der Programmschritte von 21 auf 16 (17 wenn CLEAR benutzt wird).

$$\frac{(3 \times 4) + (8 - 9)}{(8 \times 2) - 6} = 1.1$$

Schritt	Taste	Anzeigeregister X	Y	Z	Bemerkung
1	3	3	-	-	- -
2	↑	3	3	-	- -
3	4	4	3	-	- -
4	×	4	12	-	(3 x 4) → Y
5	8	8	12	-	- -
6	+	8	20	-	(3 x 4)+8 → Y
7	9	9	20	-	- -
8	−	9	11	-	(3x4)+(8–9) → Y
9	8	8	11	-	- -
10	↑	8	8	11	Speichern in Z
11	2	2	8	11	- -
12	×	2	16	11	(8x2) → Y
13	6	6	16	11	- -

Schritt	Taste	Anzeige-register X	Y	Z	Bemerkung
14	—	6	10	11	(8x2)–6 →Y
15	↓	10	11	11	Rückruf (3x4)+ (8–9)
16	÷	10	1.1	11	$\frac{(3x4)+(8-9)}{(8x2)-6}$ =1.1 →Y

Das folgende Beispiel stellt einen aufaddierenden Programmablauf dar, bei dem die arithmetischen Tasten verwendet werden, dieses Programm ist jedoch insoweit bedeutsam, als es durch geringe Veränderung zur Lösung anderer Ausdrücke dienen kann (wie Summe von Produkten im nachfolgenden Beispiel):

Schritt	Taste	Bemerkung
1	Eingabe n_1	jede Zahl für n
2	↑	- -
3	Eingabe n_2	- -
4	×	$(n_1\ n_2)$ →Y
5	Eingabe n_3	- -
6	↑	$(n_1\ n_2)$ in Z gespeichert
7	Eingabe n_4	- -
8	×	$(n_3\ n_4)$ →Y
9	↓	Rückruf $(n_1\ n_2)$
10	+	$(n_1\ n_2) + (n_3\ n_4)$ →Y
11	Eingabe n_5	- -
12	↑	$(n_1\ n_2) + (n_3\ n_4)$ in Z gesp.
13	Eingabe n_6	- -
14	×	$(n_5\ n_6)$ →Y
15	↓	Rückruf $(n_1\ n_2) + (n_3\ n_4)$
16	+	$(n_1\ n_2) + (n_3\ n_4) + (n_5\ n_6)$ →Y
17	n_7	- -
- - - - - usw. - - - - -		

Anmerkung: Nach Eingabe der n-Werte in Schritt 1 und 3 wiederholt sich die Schrittfolge, Schritt 5 bis 10, Schritt 11 bis 16 usw. Dieses „Programm" kann ebenso für die Lösung anderer Ausdrücke benutzt werden. So entsteht durch Austausch der Mal- und Plus-Taste das Produkt der Summen $(n_1 + n_2)(n_3 + n_4)(n_5 + n_6) \cdots$ etc.

Wird die Mal-Taste durch die Divisions-Taste ersetzt, so ergibt dies die Summe der Quotienten:

$$\frac{n_1}{n_2} + \frac{n_3}{n_4} + \frac{n_5}{n_6} \cdots \text{etc.}$$

Ebenso können mehr komplexe Ausdrücke wie

$$\frac{n_1 + n_2}{n_3} + \frac{n_4 + n_5}{n_6} + \text{etc.}$$

in dieser allgemeinen Programmform gelöst werden, allerdings werden dazu zusätzliche Programmschritte benötigt.

Speicher- und Rückruf-Tasten

Die Speicher- und Rückruf-Tasten $x\rightarrow( )$, $y\rightarrow( )$, $x\leftarrow( )$, $y\rightleftarrows( )$, werden in Verbindung mit den numerischen Tasten (0 bis 9), den alphanumerischen Tasten (a bis f) und den Vorzeichen-Tasten (±) zur Adressierung der Speicher-Register benutzt (s. Magnetkernspeicher, Seite 8).

Zum Speichern oder Rückruf ist der (+)-Befehl nicht erforderlich, wenn das angerufene Register auf der (+)-Seite ist, jedoch ist der (–)-Befehl obligatorisch, wenn das Register auf der (–)-Seite steht.

RCL ist ein Spezialfall, der auch unter Vektor-Tasten (Seite 51) und ACC + und ACC– beschrieben ist.

Es ist nicht nötig, ein Register vor dem Einspeichern eines neuen Wertes zu löschen, da der neue Wert den alten ersetzt.

Speichert den Inhalt des X-Registers in den Speicher, der durch die nächste Taste adressiert wird (für die + -Seite), bzw. durch die zwei nächsten Tasten (für die (–)-Seite). Der Wert des X-Registers bleibt unverändert.

(+) Seite: Drücken $x\rightarrow( )$, gefolgt von: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, oder *f*;

(–) Seite: Drücken $x\rightarrow( )$, (–), gefolgt von: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, oder *f*.

BEISPIEL:
(a) Speichern von π in das (+) c-Register
(b) Speichern von π in das (–) 2-Register

DRÜCKEN: π

(a) DRÜCKEN: $x\rightarrow( )$ C

(b) DRÜCKEN: $x\rightarrow( )$ – 2

Rückruf des Inhalts eines Registers nach X, die Adressierung des Registers erfolgt durch Drücken der nächsten Taste(n). Der Wert des angerufenen Registers bleibt unverändert.

(+) Seite: Drücken $x\leftarrow( )$ gefolgt von: 0, 1, 2, 3, 4, 5, 6, 7, 8 oder 9;

(–) Seite: Drücken $x\leftarrow( )$, (–), gefolgt von: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, oder *f*.

Es ist nicht nötig, die $x\leftrightarrow()$ Tasten zu drücken, wenn aus einem (+)-alphanumerischen Register zurückgerufen wird.

ANMERKUNG
Die Befehle $x\leftrightarrow()$ und HYPER haben den gleichen Oktalcode (s. Seite 58). Der Gebrauch beider Tasten ist austauschbar.

BEISPIEL:
Rückruf aus dem (+) c- und (–) 2-Register (im vorstehenden Beispiel gespeichert).

DRÜCKEN: CLEAR

(CLEAR ist nicht notwendig, es ist in diesem Beispiel nur zur Veranschaulichung des Rückrufs von π benutzt.)

Rückruf von π aus (+) c:

DRÜCKEN: C

ANZEIGE: π erscheint im X-Register

Rückruf von π aus (–) 2:

DRÜCKEN: CLEAR

(falsch) – 2

ANZEIGE: 2. → x

(richtig)
DRÜCKEN: $x\leftrightarrow()$ – 2

ANZEIGE: π erscheint im X-Register

Speichert den Inhalt des Y-Registers in den Speicher, der durch die nächste(n) Taste(n) adressiert wird. Der Inhalt von Y bleibt unverändert.

(+) Seite: Drücken $y\rightarrow( )$ gefolgt von: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, oder *f*;

(–) Seite: Drücken $y\rightarrow( )$, (–), gefolgt von: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, oder *f*.

Die $y\rightarrow( )$ Taste speichert den Inhalt von Y in der gleichen Weise wie $x\rightarrow( )$ den Inhalt von X, deshalb hier ohne Beispiel.

Sowohl für Speichern als auch Rückruf. Tauscht den Inhalt des Y-Registers gegen den Inhalt des durch die nächste(n) Taste(n) adressierten Registers.

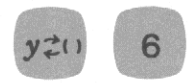
(+) Seite: Drücken $y\rightleftarrows( )$, gefolgt von: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, oder *f*;

(–) Seite: Drücken $y\rightleftarrows( )$, (–), gefolgt von: 0,1,2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, oder *f*.

BEISPIEL:

Eingabe von π in das Y-Register und Austausch mit dem Inhalt des (+) 6-Registers.

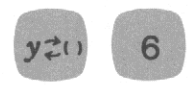
DRÜCKEN: 

DRÜCKEN: 

ANZEIGE: Ursprünglicher Inhalt von (+)6 $\rightarrow y$

(Enthält Y in der Anzeige einen unverständlichen Ausdruck, so stand im (+) 6-Register ein Programmschritt).

Rückruf von π:

DRÜCKEN: 

ANZEIGE: $\pi \rightarrow y$

Das (+) 6-Register enthält nun wieder seinen ursprünglichen Inhalt.

Die $y\rightleftarrows( )$-Taste wird besonders zum Verschieben von Daten von einem zum anderen Platz im Kernspeicher verwendet.

DRÜCKEN: y⇄( ) f

DRÜCKEN: y⇄( ) − b

DRÜCKEN: y⇄( ) 9

Der Inhalt von Y → (+)f,
der Inhalt von (+)f → (−)b,
der Inhalt von (−)b → (+)9,
der Inhalt von (+)9 → Y.

Die alphanumerischen Tasten a bis f werden sowohl zur Speicherung als auch zum Rückruf benutzt.

SPEICHERN: **Siehe x→( ), y→( ), y⇄( ), ACC+ und ACC− Tasten.**

RÜCKRUF: **Rückruf durch alleiniges Drücken der betreffenden Taste erfolgt in das X-Register (der Inhalt des angerufenen Registers bleibt unverändert).**

(+) Seite: Drücken von a, b, c, d, e, oder f;

(−) Seite: Drücken von x←( ), (−), gefolgt von a, b, c, d, e, oder f.

a
BIS
f

ANMERKUNG
Der CLEAR-Befehl löscht nur die (+)e und (+)f Register.

Siehe auch RCL in diesem und GO TO ( ) ( ) im Programmierabschnitt.

Die numerischen Tasten (0 bis 9 für die (+) Seite und (−) 0 bis (−) 9 für die (−) Seite) können nur für Speicherung oder Rückruf nach einer der folgenden Tasten verwendet werden:

x→( ), x←( ), y→( ), y⇄( ).

(Siehe auch GO TO ( ) ( ) im Programmierabschnitt.)

Im Gegensatz zu den alphanumerischen Tasten wird bei der Verwendung der numerischen Tasten allein oder in Verbindung mit der Minus-Taste, gefolgt von einer numerischen Taste, das Speicherregister nicht zurückgerufen, sondern der Inhalt des X-Registers (X und Y-Register, wenn minus benutzt wurde) geändert.

0
BIS
9

Ruft gleichzeitig den Inhalt des (+)e Registers nach Y und den Inhalt des (+)f Registers nach X zurück. Die Inhalte der Register bleiben unverändert. Das Vorzeichen der Speicher muß nicht nach RCL gedrückt werden.

RCL ist keine allgemeine Rückruf-Taste. Sie ist aber besonders in Verbindung mit den VEKTOR-Tasten (TO POLAR, TO RECT, ACC + und ACC, auf Seite 53 erklärt), wertvoll.

Nachfolgend ein allgemeines Beispiel für den Gebrauch des Kernspeichers.

BEISPIEL:

Multiplikation mit einer Konstanten;

Eine Zahlenreihe (n_1, n_2, n_3 · · · etc.) soll mit einer Konstanten k multipliziert werden. $k = 1.684$, $n_1 = 3$, $n_2 = 11.2$, $n_3 =$ · · · · etc.

Die Konstante wird zuerst gespeichert, wodurch es unnötig wird, diese bei Bedarf neue eingeben zu müssen.

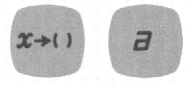
SCHALTEN:  **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

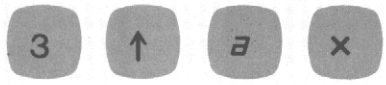
DRÜCKEN: 

Eingabe und Speichern von k in das (+)a Register (vorzugsweise in ein alphabetisches Register, da der Rückruf aus einem numerischen Register mindestens zwei Programmschritte erfordert):

DRÜCKEN: 

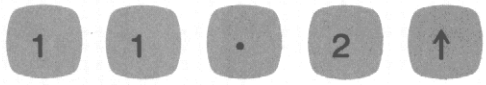
DRÜCKEN: 

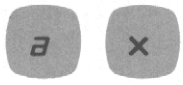
Eingabe n_1, Rückruf k, dann Multiplikation:

DRÜCKEN: 

ANZEIGE: (kn $_1$) 5.052 → y

Eingabe n_2, Rückruf k, dann Multiplikation:

DRÜCKEN: 

DRÜCKEN: 

ANZEIGE: (kn $_2$) 18.8608 → y

Eingabe n_3 . . . usw.

DEGREES ◢ **RADIANS**

Dieser Schalter wählt die Einheit, Winkel- oder Bogenmaß, in der die Rechenoperation mit den trigonometrischen Funktionen durchgeführt und angezeigt werden soll. Die Werte müssen in der gewählten Einheit eingegeben werden.

Im Winkelmaß kann von 0° bis 360° (oder größer) gerechnet werden, jedoch können die Umkehrfunktionen nur vom Grundwert der Funktion ausgehen.

θ=sin ARC x; $-90° \leq \theta \leq +90°$; $(-\pi/2 \leq \theta \leq +\pi/2)$
θ=cos ARC x; $0° \leq \theta \leq +180°$; $(0 \leq \theta \leq +\pi)$
θ=tan ARC x; $-90° \leq \theta \leq +90°$; $(-\pi/2 \leq \theta \leq +\pi/2)$

Zum Beispiel: $\cos 150° = \cos 210° = \cos 510° = (\text{etc.}) = -.866$
aber $\arccos -.866 = 150°$

Zur Umrechnung von WINKEL (BOGEN) in BOGEN (WINKEL):

1. SCHALTEN: WINKEL (BOGEN)
2. EINGABE: WINKEL (BOGEN)
3. DRÜCKEN: (alternativ) sin x, cos x, tan x;

Das Ergebnis in BOGENMASS (WINKEL) erscheint im X-Register, aber der Grundwert muß bei dieser Methode berücksichtigt werden.

Im folgenden Beispiel braucht der Grundwert nicht berücksichtigt zu werden:

Zur Umrechnung von WINKEL (BOGEN) in Bogen (WINKEL):

Umrechnung in BOGEN $\frac{\pi}{180°}$ mal Winkel

Umrechnung in WINKEL $\frac{180°}{\pi}$ mal Bogen.

sin x

Ersetzt den Wert des Y-Registers durch den Sinus dieses X-Wertes (siehe auch ARC und HYPER Tasten).

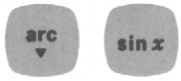
cos x

Ersetzt den Wert des X-Registers durch den Cosinus dieses X-Wertes (siehe oben).

tan x

Ersetzt den Wert des X-Registers durch den Tangens dieses X-Wertes (siehe oben).

Als Vortaste für die hyperbolische und trigonometrische Umkehrfunktion. Das Ergebnis erscheint im X-Register:



ergibt ARC sin x,

ergibt ARC sinh x.

Als Vortaste zur Errechnung der trigonometrischen Funktion; ARC und HYPER müssen als Vortaste der Funktionen gedrückt werden. Das Ergebnis erscheint im X-Register.

ANMERKUNG

Die $x \leftrightarrow ()$ und HYPER haben den gleichen Oktal-Code (67) (siehe Seite 58); die Tasten sind austauschbar.

BEISPIEL:

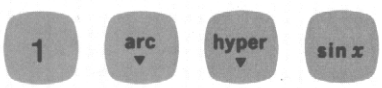
ARC sinh 1 = .88137

SCHALTEN: **DEGREES**

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

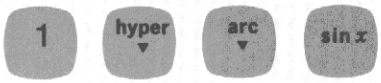
(richtige Eingabefolge)

DRÜCKEN: 

ANZEIGE: (ARC sinh 1) .88137 → x

CONTINUED

(falsche Eingabefolge)

DRÜCKEN: 

ANZEIGE: (sin ARC 1) 90.00000 → x

wird ARC nach HYPER gedrückt, wird der HYPER-Befehl gelöscht.

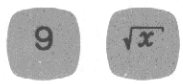
Ersetzt den Inhalt des X-Registers durch die Quadratwurzel des Wertes im X-Register.

BEISPIEL:

$\sqrt{9} = 3$

SCHALTEN:  **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 2

DRÜCKEN: 

ANZEIGE: 3.00 → x

ANMERKUNG
Zur Berechnung von $\sqrt{x}$ wenn $x = a^2 + b^2$ siehe Taste TO POLAR.

Ersetzt den Inhalt des X-Registers durch den natürlichen Logarithmus des Wertes im X-Register (Logarithmus zur Basis e).

Als Beispiel siehe

Ersetzt den Inhalt des X-Registers durch e, das mit dem X-Wert potenziert wird (den Antilogarithmus zur Basis e des Inhalts von X).

BEISPIEL:

Eingabe e in das X-Register.

SCHALTEN: **FLOATING**

DRÜCKEN: 1 e^x

ANZEIGE: 2.718 281 828 00 → x

(Anmerkung: Die nicht angezeigten Stellen enthalten . . 48)

BEISPIEL:

$19^{1.6} = 1.111\ 746\ 475 \times 10^2$

SCHALTEN: **FLOATING**

DRÜCKEN: 1 9 ln x ↑

DRÜCKEN: 1 . 6 ×

DRÜCKEN: ↓ e^x

ANZEIGE: 1.111 746 475 02 → x

BEISPIEL:

SCHALTEN: **FLOATING**

DRÜCKEN: 2 3 ln x ↑

DRÜCKEN: 5 ÷ ↓ e^x

ANZEIGE: 1.872 171 230 00 → x

log x

log x **Ersetzt den Wert des X-Registers durch den Logarithmus zur Basis 10 des Wertes im X-Register. Für die Umrechnung des Antilogarithmus zur Basis 10 wird folgende Formel benutzt:**

$x = 10^{y}$ (wobei $y = \log_{10} x$)

10^{y} kann unter Benutzung des natürlichen Logarithmus errechnet werden,

$\ln 10^{y} = y \cdot \ln 10$

die Umkehrfunktion ist ($\ln^{-1} x = e^{x}$),

BEISPIEL:

Berechnung des Antilog $_{10}$ von .60206

($\log^{-1}_{10}$ .60206 = 4)

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

Der natürliche Logarithmus von 10:

DRÜCKEN: 1 0 ln x ↑

ANZEIGE: ln 10 → y

Multiplikation ln 10 mit y:

DRÜCKEN: . 6 0 2

DRÜCKEN: 0 6 × ↓

ANZEIGE: y · ln 10 → x

Antilog ln (y x ln 10):

DRÜCKEN: e^x

ANZEIGE: ($\log^{-1}_{10}$ .60206 =) 4.00000 → x

int x

Schneidet den dezimalen Teil des Wertes des X-Registers ohne Rundung und Beeinflussen des Vorzeichens weg.

BEISPIEL:

Ganzzahl $-5.9 = -5$

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 4

DRÜCKEN: CHG SIGN 5 . 9

ANZEIGE: $-5.9 \rightarrow x$

DRÜCKEN: int x

ANZEIGE: $-5. \rightarrow x$

BEISPIEL:

Umrechnung, von 5.72° in Grad und Minuten, (°) bzw. (') bzw. [5°43.2']

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

Eingabe 5.72

DRÜCKEN: 5 . 7 2 ↑

trennt den ganzzahligen Teil (Grad) vom dezimalen Teil (Minuten):

DRÜCKEN: int x —

Speichern der Gradzahl in das Z-Register:

DRÜCKEN: ROLL ↓

zur Umrechnung des dezimalen Teils in Minuten durch Multiplikation mit 60:

DRÜCKEN: x⇄y 6 0 ×

ANZEIGE: (°) $5. \rightarrow z$

(') $43.2 \rightarrow y$

Absoluter Wert von Y. Macht den Wert des Y-Registers positiv ohne Veränderung des Vorzeichens des Exponenten.

BEISPIEL:

SCHALTEN: **FLOATING**

DRÜCKEN: CLEAR x | CHG SIGN | 5

DRÜCKEN: ENTER EXP | CHG SIGN | ↑

ANZEIGE: -5. -00 → y

DRÜCKEN: |y|

ANZEIGE: 5. -00 → y

Dieser Befehl wird (meistens als Programmschritt) zum Testen einer Zahl innerhalb vorgegebener (+ oder –) Limits verwendet. Das Testprogramm im Teil „Verschiedenes" in der Programm-Bibliothek enthält dafür einige Beispiele.

Löscht das X-Register und liest den Wert π ein. Bis zu einschließlich zehn Ziffern werden angezeigt, zwei weitere Ziffern enthalten die nicht angezeigten Stellen (. . 60) (siehe Seite 11).

Die Vektortasten (TO POLAR, TO RECT, ACC+, ACC- und RCL) gestatten die vollständige Rechnung der komplexen und vektoriellen Arithmetik.

Umsetzung von rechtwinkligen Koordinaten in polare Koordinaten berechnet den Winkel, θ im Bereich:

$-180° < \theta \leq +180°$
$-\pi$ radians $< \theta \leq +\pi$ radians

TO POLAR

Umsetzung von rechtwinkligen Koordinaten, die aus einer x-Komponente im X-Register und einer y-Komponente im Y-Register bestehen, in Polar-Koordinaten:

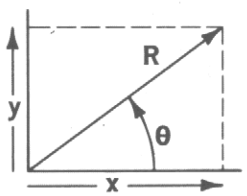
TO POLAR

Winkel $(\theta) =$ **ARC** Tan y/x $\rightarrow$ Y

Radius $(R) = \sqrt{x^2 + y^2}$ $\rightarrow$ X

BEISPIEL:

(a): $x = 4$, $y = 3$, Umsetzung in polare Form.



SCHALTEN: **DEGREES**

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 3

DRÜCKEN: 3 ↑ 4 TO POLAR

ANZEIGE: (θ, in Grad) 36.870 → y
(R) 5.000 → x

SCHALTEN: **RADIANS**

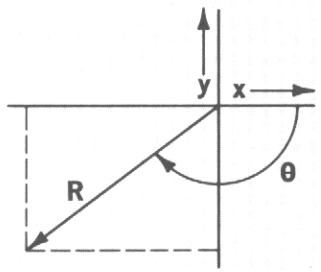
DRÜCKEN: Das vorstehende Beispiel wiederholen

ANZEIGE: (θ, in Bogenmaß) .644 → y
(R) 5.000 → x

TO POLAR

BEISPIEL:

(b): x = –4, y = –3, Umsetzung in polare Form.



SCHALTEN:

SCHALTEN:

DREHEN: DEZIMALSTELLENWÄHLER auf 3

DRÜCKEN: 3 CHG SIGN ↑ 4 CHG SIGN

DRÜCKEN: TO POLAR

ANZEIGE: (θ, in Winkelmaß) -143.130 → y
(R) 5.000 → x

TO RECT

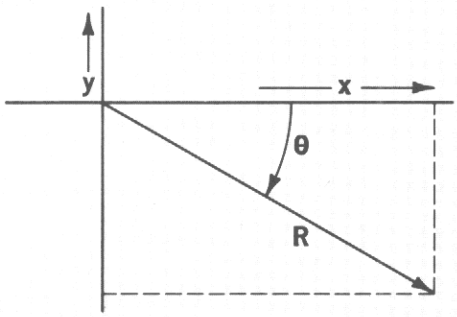
Umsetzung von Polarkoordinaten, die aus einem Radius (r) im X-Register und einem Winkel (θ) im Y-Register bestehen, in rechtwinklige Koordinaten:

x Komponente = R sin θ →Y

y Komponente = R Cos θ →X

BEISPIEL:

R = 8. θ = –30° (330°), Umsetzung in rechtwinklige Form.



SCHALTEN: **DEGREES**

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 3

DRÜCKEN: 3 0 CHG SIGN ↑ 8

DRÜCKEN: TO RECT

alternativ

DRÜCKEN: 3 3 0 ↑ 8

DRÜCKEN: TO RECT

ANZEIGE: (y) -4.000 → y

(x) 6.928 → x

Die ACC+, ACC– und RCL-Tasten sind ausschließlich Speicher und Rückruf-Tasten für den alleinigen Gebrauch in Verbindung mit den (+)e und (+)f Registern. Diese Tasten erlauben Vektor-Addition und Vektor-Subtraktion durch einen einzigen Tastendruck.

Bei Verwendung dieser Tasten ist zu beachten, daß die CLEAR-Taste den Inhalt der (+)e und (+)f Register löscht.

ACC +

ACC –

RCL

ANMERKUNG

Die + und – Vorzeichen der ACC + ACC-Tasten haben nichts mit den Seitenvorzeichen des Kernspeichers gemeinsam. Beide Tasten arbeiten nur in Verbindung mit den (+) e und (+) f Registern.

ACC+: Addiert den Inhalt des X-Registers zu dem ursprünglichen Inhalt des (+)f Registers und gleichzeitig den Inhalt des Y-Registers zu dem ursprünglichen Inhalt des (+) e-Registers. Die Summen werden in (+)e, bzw. (+)f gespeichert. Die Inhalte der X- und Y-Register bleiben unverändert.

$(+)e + y \rightarrow (+)e$

$(+)f + x \rightarrow (+)f$

ACC–: Subtrahiert den Inhalt des X-Registers von dem ursprünglichen Inhalt des (+)f Registers und gleichzeitig den Inhalt des Y-Registers von dem ursprünglichen Inhalt des (+)e Registers. Die Differenzen werden in (+)f, bzw. (+)e gespeichert. Die Inhalte der X und Y-Register bleiben unverändert.

$(+)e - y \rightarrow (+)e$

$(+)f - x \rightarrow (+)f$

RCL: Rückruf der Inhalte von (+)f in das X-Register und (+)e in das Y-Register. Die Inhalte der (+)f und (+)e Register bleiben unverändert.

$(+)e \rightarrow Y$

$(+)f \rightarrow X$

BEISPIEL:

(a): Vektoraddition

$(2x+3y)+(4x+5y)-(3x-6y) = 3x+14y$

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 3

Schritt	Taste	Anmerkung
1	CLEAR	um (+)e und (+)f zu löschen y wird vor x eingegeben
2	3	
3	↑	..
4	2	..
5	ACC +	$0+3=3 \rightarrow (+)e$ $0+2=2 \rightarrow (+)f$
6	5	..
7	↑	..
8	4	..
9	ACC +	$3+5=8 \rightarrow (+)e$ $2+4=6 \rightarrow (+)f$
10	6	..
11	CHG SIGN	..
12	↑	..
13	3	..
14	ACC −	$8-(-6)=14 \rightarrow (+)e$ $6-3=3 \rightarrow (+)f$
15	RCL	$(+)e \rightarrow Y$ $(+)f \rightarrow X$

ANZEIGE: (y) 14. → y

(x) 3. → x

ANMERKUNG
Im folgenden Beispiel soll die mögliche Kombination der Vektortasten und der logarithmischen Tasten dargestellt werden. Dadurch ist eine gleichzeitige Addition und Multiplikation von Zahlen möglich.

BEISPIEL:

(a): Multiplikation komplexer Zahlen

$(j=i=\sqrt{-1})$

$(3+j4)\ (-2+j3)=-18+j1$

Diese Gleichung wird im Rechner so gelöst, daß die gleiche Methode leicht für die Berechnung mit vielen komplexen Ausdrücken abgewandelt werden kann:

1. Umwandlung der Klammerausdrücke in die polare Form $(R\ \angle\Theta)$,
2. Multiplikation der r-Größen und Addition der Winkel und
3. Rückwandlung in rechtwinklige Form.

SCHALTEN: **DEGREES**

SCHALTEN: **FIXED POINT**

DREHEN: DEZIMALSTELLENWÄHLER auf 3

Löschen des (+) e und (+) f Registers:

DRÜCKEN: CLEAR

eingeben (3+j4) und Umwandlung in Polarkoordinaten $(R_1\ \angle\Theta_1)$:

DRÜCKEN: 4 ↑ 3 TO POLAR

den ln von r_1 errechnen und ln r_1 in (+) f und $\angle\Theta_1$ in (+) **e** speichern:

DRÜCKEN: ln x ACC +

eingeben (−2 + j3) und Umwandlung zu Polarkoordinaten $(R_2\ \angle\Theta_2)$:

DRÜCKEN: 3 ↑ 2 CHG SIGN TO POLAR

Den ln von R_2 errechnen und ln R_2 in (+) f und $\angle\Theta_2$ (+) e speichern:

DRÜCKEN: [ln x] [ACC +]

Rückruf aus (+) e nach Y und (+) f nach X, dann Anti-ln (e-Funktion) von (lnR1 + lnR2)

DRÜCKEN: [RCL] [e^x]

Umwandlung des Ergebnisses (R in X und $\angle\Theta$ in Y) in rechtwinklige Koordinaten:

DRÜCKEN: [TO RECT]

ANZEIGE: (j) /. → y Y (imaginärer)
-/8. → x X (reeller Teil)

ANMERKUNG

Das vorstehende Beispiel kann in eine Division komplexer Zahlen wie $\frac{3+j4}{-2+j3}$ geändert werden, indem an Stelle von ACC + beim zweiten Mal ACC- gedrückt wird.

Einführung in die Programmierung

Dieser Abschnitt beschreibt die Programmtasten und die Art der Programmierung des 9100B. Der auf Seite 8 bereits kurz beschriebene Magnetkernspeicher ist zur besseren Ausnutzung durch den Programmierer ausführlicher behandelt.

Die Programmierung des 9100B ist einfach, da die Tasten die Programmbefehle beinhalten und keine besondere Programmsprache zu erlernen ist. Es ist jedoch unerläßlich, daß der Programmierende genau die Funktion der einzelnen Tasten kennt und dem Rechner die Befehle in der logischen Reihenfolge gegeben werden. Die Berichtigung von Fehlern im Programmablauf ist einfach.

Die untenstehende Abb. 2 gibt die dreidimensionale Darstellung einer Seite des Magnetkernspeichers wieder. Dieser besteht jedoch aus zwei Seiten, der (+) Seite und der (-) Seite.

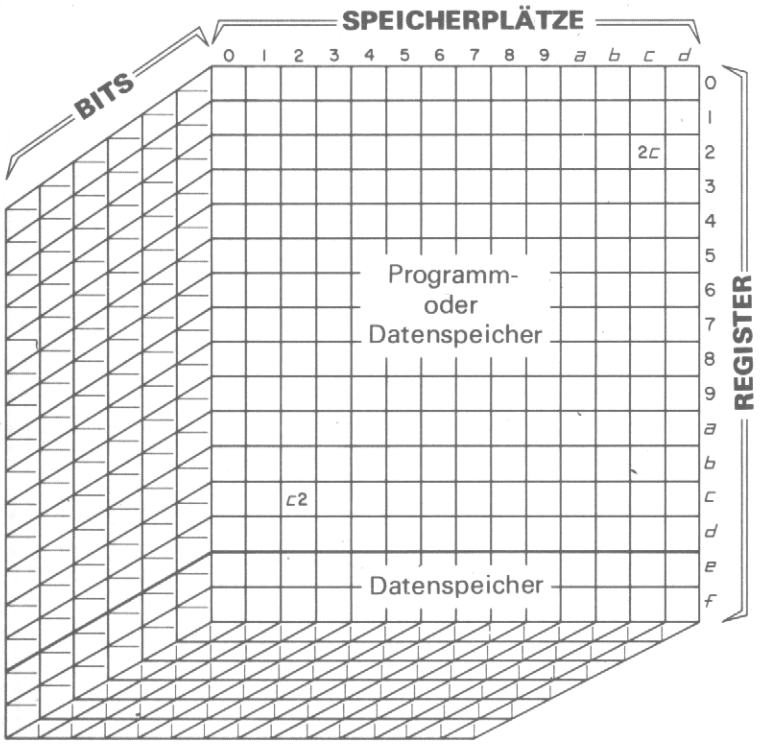


Abb. 2 Programmierbarer Speicher

Register

Wie vorher beschrieben, besteht eine Seite aus sechzehn Registern:

(±) 0 bis (±) d - diese werden für Programm- oder Datenspeicherung benutzt.

(±) e bis (±) f - diese werden nur für Datenspeicherung benutzt.

Speicherplätze

Die achtundzwanzig Programmregister, (±) 0 bis (±) d, enthalten je vierzehn Speicherplätze. Jeder Speicherplatz kann entweder einen Programmschritt (einen Tastendruck) - das ergibt die Maximal-Kapazität von 392 Programmschritten, - oder eine Ziffer einer Zahl enthalten. Das Speichern einer Zahl in einem Register kann maximal zwölf Ziffern für die Mantisse und zwei Ziffern für den Exponenten belegen, so daß sich für jede in einem Register (nicht als Schrittfolge im Programm) gespeicherte Zahl die Zahl der Programmschritte um 14 vermindert. Daten und Programmschritte können nicht gemeinsam in einem Register gespeichert werden (außer als Schrittfolge).

Speicher-Adresse

Jeder Speicherplatz hat eine dreiteilige Adresse; diese sind (immer in dieser Folge):

1) das Seitenvorzeichen (+ oder -),
2) die Registerbezeichnung (0 bis d) und
3) die Speicherplatzbezeichnung (0 bis d)

Der dritte Speicherplatz im Register c auf der (-) Seite wird mit ' (-) c2' adressiert;

„(-) 2c" ist die Adresse des Speicherplatzes c im dritten Register der (-) Seite;

„(+) 2c" ist die Adresse des Speicherplatzes c im dritten Register der (+) Seite.

Bits

Jeder Speicherplatz ist sechs Stellen tief, die als Bits bezeichnet werden. Diese sechs Bits enthalten den Code eines Programmschrittes oder der Ziffer einer Zahl.

Der Code jeder Taste ist im Anhang am Ende der Bedienungsanleitung und in der herausziehbaren Karte an der Vorderseite des Rechners angegeben. Der Code ist oktal (basierend auf 8 an Stelle von 10, daher gibt es keine 8er und 9er). Jede Taste wird durch eine zweiziffrige Zahl dargestellt und im Programm benutzt. STEP PRGM (Oktal-Code 51) ist kein Programmschritt. Jeder zweiziffrige Befehls-Code kann in den sechs Bits jedes Speicherplatzes gespeichert werden (das ist kein Widerspruch zum Vorhergesagten, daß nur eine Ziffer einer Zahl pro Speicherplatz gespeichert werden kann, da jede Ziffer durch einen zweistelligen Code ausgedrückt werden muß).

Programm-Adressierung

Es ist nicht nötig, den Oktal-Code zur Programmierung zu verwenden, aber als Ausgabe oder Berichtigung des Programms. Die Bezeichnung des Kernspeichers in zwei Dimensionen, Register und Speicherplätze, ist einfacher.

Zu Beginn des Programmierens wird zuerst die Start-Adresse zugeteilt (Da der END-Befehl den Befehl GO TO (0) (0) beinhaltet, wie auf Seite 70 beschrieben, ist die Start-Adresse gewöhnlich (+)(0)(0). Bei der Eingabe jedes Programmschrittes in den Rechner läuft der Programmzähler automatisch mit der Speicherung weiter, beginnend bei der Start-Adresse. Die Programmzähler und -Speicher arbeiten bei Startadresse (+) 00 beginnend wie folgt: Ist die Adresse (+) dd belegt und damit Seite (+), schaltet der Programmzähler automatisch zur (-) Seite: Ist die (-) dd-Adresse belegt, springt der Programmzähler automatisch zur Adresse (+) 00 zurück, so daß der darin gespeicherte Schritt durch den neu eingegebenen Programmschritt überschrieben wird.

(+)00, (+)01, · · · · · (+)09, (+)0*a*, (+)0*b*, (+)0*c*,
(+)0*d*, (+)10, · · · · · (+)19, (+)1*a*, (+)1*b*, (+)1*c*,
(+)1*d*, (+)20, · · · · · bis · · · · · (+)*dc*, (+)*dd*.

· · · (+)*dc*, (+)*dd*, (−)00, (−)01, · · · · · bis · · · ·
(−)*dc*, (−)*dd*.

Programm-Zähler

Am besten wird der „Programm-Zähler" als elektronischer Schaltkreis bezeichnet, der den Rechner durch die Schrittfolge im Kernspeicher führt. Der Programmierer kann den Programmzähler jedoch zu Beginn an jede beliebige Adresse setzen, als Adressieren des Programmzählers bezeichnet. (Siehe GO TO Taste auf Seite 68). Der Programmierer kann den Zähler ebenso eine Verzweigung im Programm ausführen lassen (d. h. GO. TO einer anderen Adresse und Weiterlauf des Programms). Eine Verzweigung ist auf zwei Arten möglich: ‚Bedingt' und ‚Unbedingt'.

Verzweigung

Bei einer bedingten Verzweigung entscheidet der Rechner selbst ob zu verzweigen ist oder nicht, zum Beispiel kann er abfragen, ob eine Zahl größer ist als eine andere. Ist die Antwort ‚NEIN', so springt der Rechner zu einer anderen Adresse und rechnet dort weiter; ist die Antwort ‚JA', so liest er den nächstfolgenden Programmschritt. Bedingte Verzweigung ist auf Seite 83 ausführlich beschrieben. Die unbedingte Verzweigung läßt dem Rechner keine Wahl oder Entscheidung, er muß zum adressierten Programmschritt springen (siehe GO TO Taste auf Seite 68).

Unterprogramme

Der 9100B kann Unterprogramme fahren; dies ist eine Folge von Programmschritten, die beliebig oft verwendet werden können, aber nur einmal gespeichert werden. Das Hauptprogramm kann an jeder Stelle das Unterprogramm abrufen (d. h. dorthin verzweigen) und kehrt nach dessen Beendigung zu dem, dem Aufruf des Unterprogramms folgenden Programmschritt, automatisch zurück. Damit werden also Programmschritte gespart (Unterprogramme sind auf Seite 94 beschrieben).

Anzeige

Die Anzeige erlischt während des Programmablaufs. Hält das Programm an, so erscheint die Anzeige wieder.

Viele Programme sind so kurz, daß die Anzeige nur blinkt und sofort wieder erscheint. Bei langen Programmen, z. B. dem Diagnostik-Programm, kann die Anzeige für einige Sekunden - oder länger - verlöschen. Man kann auch ein Programm so schreiben, daß der Rechner in einer Schleife rechnet und ohne STOP-Befehl keine Anzeige mehr erfolgen kann. Das Tastenfeld ist währenddessen blockiert, bis durch den STOP-Befehl das Programm angehalten wird.

Ein Fehler im Programm kann ein falsches Ergebnis liefern oder zu einer Schleife führen, so daß keine Anzeige erfolgen kann. Der STOP-Befehl hält dann den Rechenablauf an, und die Anzeige erscheint wieder.

Schreiben eines Programms

Was ist ein Programm?

Ein Programm ist nichts anderes als die logische Folge von Befehlen, die dem Rechner die Durchführung einer bestimmten Rechnung übertragen. In den vorherigen Abschnitten dieser Anleitung wurde der ‚Operator' programmiert, indem im Ablauf des Programms eine bestimmte Reihenfolge von Befehlen eingehalten und gedrückt werden mußte. Andernfalls erschien in der Anzeige ein unrichtiges Ergebnis. Ähnlich soll dem Rechner jetzt eine Folge der richtigen Befehle eingegeben werden.

So soll zum Beispiel versucht werden:

> Eingabe von drei Zahlen (A, B und C) und zwar so, daß A im Z-Register, B im Y-Register und C im X-Register erscheinen. Etwa A = 6, B = 5 und C = 3.

Es soll angezeigt werden:

(A) 6 → z

(B) 5 → y

(C) 3 → x

Jetzt sollen die Tasten so gedrückt werden, daß die Gleichung $\frac{A \times B}{C}$ gelöst wird.

$\left[\frac{6 \times 5}{3}\right]$.

Die einzelnen Programmschritte sollen in der Reihenfolge ihrer Durchführung aufgeschrieben werden, ebenso die Anzeige in den drei Registern (der Platz unten reicht dafür aus, der erste Schritt soll in 00 stehen; der letzte Schritt soll END sein).

Schritt	Taste	Anzeige		
		X	Y	Z
Eingabe der Zahlen		C(=3)	B(=5)	A(=6)
(+)00				
01				
02				
03				
04				
05				
06				
07				
08				
etc.				

Was ist ein Programm?

Wir haben jetzt ein brauchbares Programm vorliegen, das wie folgt eingegeben wird.

SCHALTEN:  **RUN**

DRÜCKEN: END

SCHALTEN: **PROGRAM** 

DRÜCKEN: Die Tasten in der Folge des Programms

SCHALTEN:  **RUN**

DRÜCKEN: END

Zum Ablauf des Programms:
A, B und C in Z, Y und X eingeben

DRÜCKEN: CONT

Hier zwei Programme (von mehreren Möglichkeiten) zur Lösung von $\frac{A \times B}{C}$

1)

Schritt	Taste	Anzeige X	Anzeige Y	Anzeige Z
Eingabe der Zahlen		C	B	A
00	ROLL ↓	B	A	C
01	×	B	A x B	C
02	↓	A x B	C	C
03	$x \rightleftarrows y$	C	A x B	C
04	÷	C	$\frac{A \times B}{C}$	C
05	END	C	$\frac{A \times B}{C}$	C

2)

Schritt	Taste	Anzeige X	Anzeige Y	Anzeige Z
Eingabe der Zahlen		C	B	A
00	÷	C	B/C	A
01	↓	B/C	A	A
02	×	B/C	$\frac{A \times B}{C}$	A
03	END	B/C	$\frac{A \times B}{C}$	A

Schreiben eines Programms

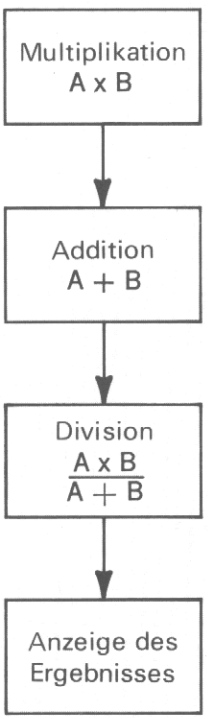
Programmschreiben kann in drei Hauptteile gegliedert werden:

a) Definition des Problems,
b) der Lösungsweg und
c) die Niederschrift der Schrittfolge für den Rechner.

Beispiel für Programmschreiben:

a) Definition des Problems: Es soll ein Programm für die Gleichung $\frac{A \times B}{A+B}$ geschrieben werden, wobei A und B beliebige Werte sein können. Nach Beendigung des Programms sollen das Ergebnis und die Werte für A und B angezeigt werden.

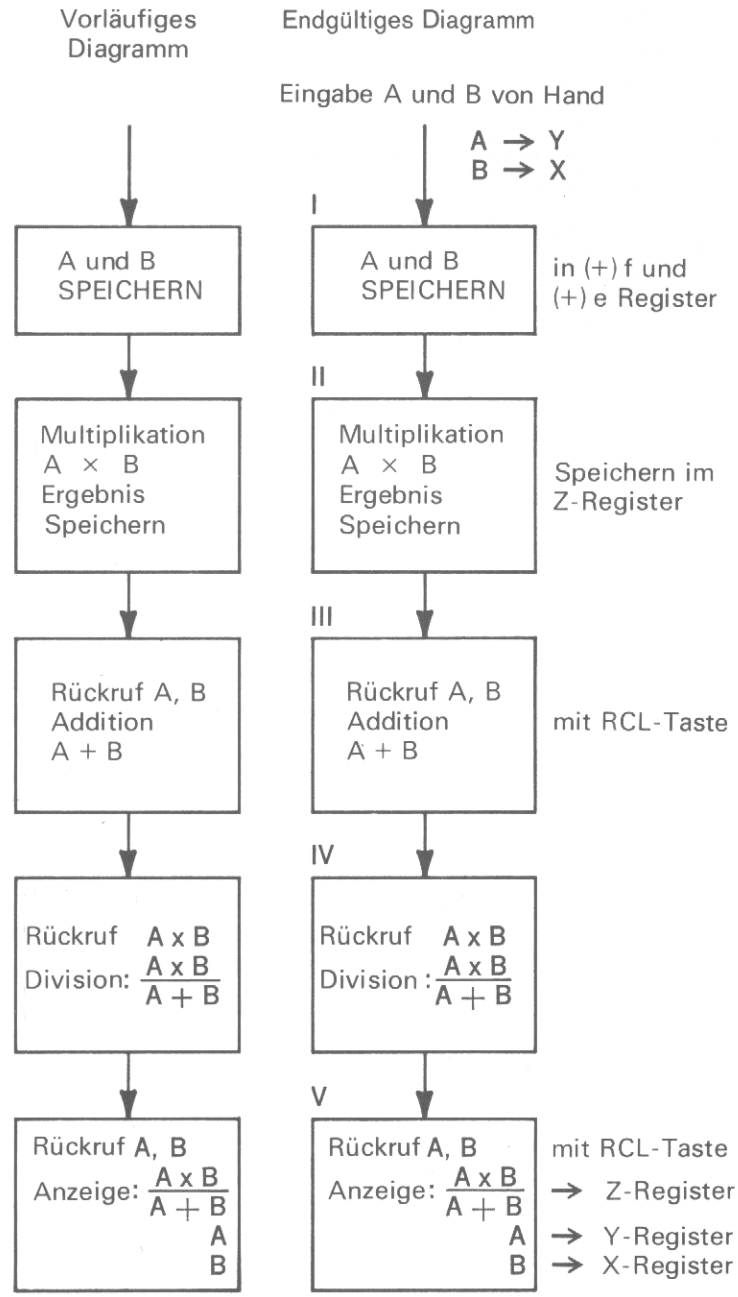
b) Lösungsweg: Dies erfolgt am besten durch ein Fluß-Diagramm, das so einfach wie möglich sein sollte.



Nun wird das Fluß-Diagramm detailliert und besondere Bemerkungen, wie Speicherbelegung usw. hinzugefügt. Es kann bei einem komplexen Gleichungssystem nötig sein, mehrere Diagramme zu zeichnen.

Schreiben eines Programms

Schreiben eines Programms



c) Die Schrittfolge für den Rechner: Diese soll genauso wie bei manueller Rechnung erfolgen. Für jeden Schritt soll auch die Anzeige und die Belegung der Speicher dargestellt werden (die Verwendung der mitgelieferten Programmformulare ist zu empfehlen).

(Beginnend mit I im entgültigen Fluß-Diagramm) A und B werden gespeichert:

Schritt (Adresse)	Taste	Anzeige			Speicher	
		X	Y	Z	(+)f	(+)e
(+)00	x→( )	B	A	–	–	–
01	f	B	A	–	B	–
02	y→( )	B	A	–	B	–
03	e	B	A	–	B	A

(II im Fluß-Diagramm) Multiplikation A x B und Speichern des Ergebnisses:

Schritt (Adresse)	Taste	Anzeige			Speicher	
		X	Y	Z	(+)f	(+)e
04	×	B	A x B	–	B	A
05	↑	B	B	A x B	B	A

(III im Fluß-Diagramm) Rückruf A und B deren Addition:

Schritt (Adresse)	Taste	Anzeige			Speicher	
		X	Y	Z	(+)f	(+)e
06	RCL	B	A	A x B	B	A
07	+	B	A + B	A x B	B	A

(IV im Fluß-Diagramm) Rückruf (A x B) und Division:

Schritt (Adresse)	Taste	Anzeige			Speicher	
		X	Y	Z	(+)f	(+)e
08	↓	A + B	A x B	A x B	B	A
09	÷	A + B	$\frac{A \times B}{A + B}$	A x B	B	A

Schreiben eines Programms

(V im Fluß-Diagramm) Anzeige des Ergebnisses und A und B:

Schritt (Adresse)	Taste	Anzeige X	Anzeige Y	Anzeige Z	Speicher (+)f	Speicher (+)e
0a	↑	A + B	A + B	$\frac{A \times B}{A + B}$	B	A
0b	RCL	B	A	$\frac{A \times B}{A + B}$	B	A
*0c	END	B	A	$\frac{A \times B}{A + B}$	B	A

ANMERKUNG

END (oder STOP) muß eingegeben werden, sonst würde der Rechner mit dem nächsten folgenden Schritt fortfahren. Der Programmzähler liest schrittweise ohne Berücksichtigung des Inhalts weiter. END wird deshalb als letzter Programmschritt vorgezogen, da es den Programmzähler automatisch nach (+)00 zurücksetzt und damit neue Werte für A und B eingegeben werden können.

Nachfolgend ein vollständiges Programm, das später in diesem Abschnitt zur Illustration von Programm-Operationen verwendet wird (Seite 72). Die Code-Ziffern wurden hinzugefügt.

Schritt (Adresse)	Taste	CODE	Anzeige X	Anzeige Y	Anzeige Z	Speicher (+)f	Speicher (+)e
(+)00	x→( )	23	B	A	–	–	–
01	f	15	B	A	–	B	–
02	y→( )	40	B	A	–	B	–
03	e	12	B	A	–	B	A
04	×	36	B	A x B	–	B	A
05	↑	27	B	B	A x B	B	A
06	RCL	61	B	A	A x B	B	A
07	+	33	B	A + B	A x B	B	A
08	↓	25	A + B	A x B	A x B	B	A
09	÷	35	A + B	$\frac{A \times B}{A + B}$	A x B	B	A
0a	↑	27	A + B	A + B	$\frac{A \times B}{A + B}$	B	A
0b	RCL	61	B	A	$\frac{A \times B}{A + B}$	B	A
0c	END	46	B	A	$\frac{A \times B}{A + B}$	B	A

Nach Beendigung eines Programms ist es oft zweckmäßig, das Programm mit den Zahlenwerten von Hand schrittweise durchzugehen, um festzustellen, ob die Anzeige tatsächlich die Werte enthält.

BEISPIEL:

Ausführung obigen Programms von Hand

SCHALTEN:  **FIXED POINT**

SCHALTEN:  **RUN**

DREHEN: DEZIMALSTELLENWÄHLER auf 5

DRÜCKEN: beliebige Zahl für A

DRÜCKEN: 

DRÜCKEN: beliebige Zahl für B

DRÜCKEN: Die Tasten in der Schrittfolge des Programms.

Vorstehendes Programm könnte so aussehen:

Schritt (Adresse)	Taste	Code	Anzeige			Speicher	
			X	Y	Z	(+)f	(+)g
00	CLEAR	20	0	0	0	0	0
01	STOP	41	0	0	0	0	0
A und B von Hand eingeben		. . .	B	A	0	0	0
02	ACC +	60	B	A	0	B	A

Das Ergebnis in der Anzeige und die Speicher-Register sind bei Schritt 02 gleich Schritt 03 im Original-Programm; ein Schritt wurde gespart. In dieser alternativen Form ist es jedoch notwendig, die CONT–Taste zweimal für den Programmablauf zu drücken; zum ersten Mal für den Programm-Start, zum zweiten Mal nach Eingabe der Daten. Im Original-Programm mußte CONT nur einmal, und zwar nach der Eingabe der Werte gedrückt werden (siehe Seite 69 CONT-Taste).

Programm-Tasten

Wahl der Betriebsart

PROGRAM  **RUN**

PROGRAM: Bei Eingabe eines Programms über das Tastenfeld oder beim Auslesen mittels Anzeige.

RUN: Bei Rechenoperationen von Hand, Aufzeichnung eines Programms auf Magnetkarte, Einlesen eines Programms mittels Magnetkarte, bei automatischem Programmablauf und Adressierung des Programm-Zählers.

Zur Adressierung des Programm-Zählers und als unbedingte Verzweigung zu der Stelle, die durch die, dem GO TO-Befehl unmittelbar folgenden, beiden Schritte, bezeichnet wird. GO TO, gefolgt von dem SUB/RETURN-Befehl mit anschließender Adresse, ist eine unbedingte Verzweigung zum Unterprogramm an der bezeichneten Adresse (siehe Seite 94).

Die Verwendung von GO TO über das Tastenfeld oder per Programm ist seitenabhängig (+oder –). Das Seitenvorzeichen ist nur bei Verzweigung von einer (+) Seite zur (–) Seite oder umgekehrt erforderlich.

Von ( ) Seite	zu ( ) Seite	nötiger Befehl	Schrittzahl
(+)	(+)	GO TO, 4, 3	3
(–)	(–)	GO TO, 4, 3	3
(+)	(–)	GO TO, –, 4, 3	4
(–)	(+)	GO TO, +, 4, 3	4

ANMERKUNG

Obwohl eine Verzweigung seitenabhängig ist, erfolgt die Adressierung eines Registers zur Speicherung oder zum Rückruf immer zur (+) Seite, es sei denn, das anzurufende Register befindet sich auf der (-) Seite. Dann ist das Seitenvorzeichen vor die Adresse zu setzen.

Für die Adressierung des Kernspeichers wird für jede Taste ein Programmschritt benötigt.

Richtig

SCHRITT	TASTE
(+)29	GO TO ( )( )
2a	—
2b	5
2c	C

Falsch

SCHRITT	TASTE
(+)29	GO TO ( )( )
2a	−5
2b	C
2c	

Folgt auf GO TO eine Adresse, aber kein Seitenvorzeichen, so wird die Verzweigung immer zu der angegebenen Adresse der Seite ausgeführt, auf der der GO TO-Befehl steht.

SCHRITT	TASTE
(+)dc	GO TO ( )( )
(+)dd	5
(−)00	8
(−)01	

das Programm springt zu (+) 58.

CONT

CONTINUE wird zum Start des Programms gedrückt. Das Programm beginnt bei der Adresse, an der es gerade steht. Es ist ein Programmschritt ohne mathematische Funktion (siehe auch Seite 79, Programmberichtigung). Auch kann CONT zum Auffüllen des Kernspeichers benutzt werden, um Raum für spätere Rechenoperationen freizuhalten.

Hat das Programm auf STOP-Befehl angehalten, so muß das Programm mit CONT wieder gestartet werden (beim nächsten Programmschritt).

Hat das Programm auf END-Befehl angehalten, wird der Programmzähler durch Drücken von CONT nach (+) 00 gesetzt und das Programm dort gestartet.

STOP

Hält das Programm an und läßt den Rechner anzeigen (siehe Seite 60). Im Programm wird der Stop-Befehl zur Dateneingabe von Hand benutzt.

Hat ein Programm angehalten, kann jede Operation über das Tastenfeld durchgeführt werden, ohne daß das gespeicherte Programm davon berührt wird. Werden jedoch während der Schrittfolge Daten in einen mit Programmschritten belegten Speicher eingegeben, so werden die Programmschritte in diesem Speicher gelöscht.

ENDE des Programmablaufs und Anzeige. Setzt den Programmzähler nach (+) 00, so daß CONT den Programmablauf bei (+) 00 neu startet.

Als Programmschritt ist END gleichbedeutend mit:

STOP, GO TO, +, 0, 0.

Als Taste ist END gleichbedeutend mit:

GO TO, +, 0, 0.

Bei der Eingabe eines Programms über Magnetkarte, an dessen Ende der END-Befehl steht, beendet der Kartenleser das Einlesen und setzt den Programmzähler nach (+) 00. Das bewirkt, daß alle Befehle nach dem END-Befehl auf der gleichen Seite nicht wieder eingelesen werden können. Aus diesem Grunde wird END durch einen beliebigen Programmschritt ersetzt, wenn es nicht der letzte Programmschritt ist, oder wenn auf END noch Daten folgen, bevor das Programm auf Magnetkarte aufgezeichnet werden soll. Ist das Programm über Magnetkarte wieder eingelesen, kann der END-Befehl eingefügt werden (siehe Berichtigung eines Programms auf Seite 79).

Bei diesem Befehl läuft das Programm in Einzelschritten durch. Dies ist die einzige Taste, die nicht als Programmschritt in den Rechner eingegeben werden kann.

In PROGRAM erscheint auf Drücken von STEP PRGM die nächste Adresse mit der Code-Ziffer des Befehls in dieser Adresse im Z-Register. Die letzte Adresse mit der Code-Ziffer des Befehls in dieser Adresse erscheint im X-Register.

ANMERKUNG

Der ursprüngliche Inhalt des Z-Registers erscheint bei Schalten des PROGRAM-RUN-Schalters auf RUN wieder. Der Inhalt des X-Registers ist verloren.

In RUN läuft das Programm durch Drücken von STEP PRGM schrittweise durch, wobei die Inhalte von X, Y und Z angezeigt werden. Daten müssen zum jeweiligen Programmschritt zur Kontrolle des Programms und dessen Ablauf eingegeben werden.

Der Gebrauch von STEP PRGM ist unter AUSLESEN EINES PROGRAMMS und BERICHTIGUNG EINES PROGRAMMS ausführlich beschrieben (Seite 76 folgende). Verschiedene Programmbefehle arbeiten nicht wie erwartet mit STEP PRGM, diese sind gleichfalls unter AUSLESEN EINES PROGRAMMS erklärt.

Bewirkt kurzzeitige Anzeige (etwa 1 / 8 Sekunde), bevor der nächste Schritt ausgeführt wird. Für längere Anzeige kann PAUSE nochmals eingegeben werden. Wird die PAUSE-Taste gedrückt gehalten, liest der Rechner den nächsten PAUSE-Befehl während des Programmablaufs als Stop-Befehl (durch CONT läuft das Programm dann weiter).

Der PAUSE-Befehl ist besonders bei langen Schleifendurchläufen wertvoll. Damit können Teilergebnisse der Rechnung während des Programmablaufs beobachtet werden. Er erlaubt das Anhalten des Programms an einem bestimmten Punkt ohne den STOP-Befehl, bei dem nach jedem Schleifendurchlauf das Programm mit CONT neu gestartet werden müßte. Ohne den PAUSE-Befehl bleibt die Anzeige für die interne Rechnung frei. Der Bedienende hat keine Kontrollmöglichkeiten, ob das Programm richtig abläuft. Ohne PAUSE wäre ein gewisser Sicherheitsfaktor nicht einzubauen.

PRINT wird als Befehl für periphere Geräte benutzt. Ausführliche Beschreibung in der Bedienungsanleitung des betreffenden Gerätes. Ohne Drucker ist dieser Befehl wie STOP zu verwenden, zur Fortführung des Programms muß CONT gedrückt werden.

Der FORMAT-Befehl, in Verbindung mit dem nächsten Befehl, steuert ebenfalls periphere Geräte. Er gestattet, den Rechner als Steuereinheit kleiner Systeme zu verwenden. Ohne Zusatzgeräte liest der Rechner im Programm FMT und den nächsten Schritt als STOP. CONT bewirkt den Weiterlauf des Programms, beginnend beim zweiten Schritt nach FMT.

Die Anzeige verlöscht auf Drücken von FMT, durch Drükken einer beliebigen Taste kehrt die Anzeige zurück.

Ausführliche Beschreibung in der Bedienungsanleitung des betreffenden Gerätes.

Programm-Durchführung

Jede der untenstehenden Operationen wird beschrieben, wobei das Beispiel (auf Seite 66) verwendet wird.

Eingabe eines Programms über das Tastenfeld.

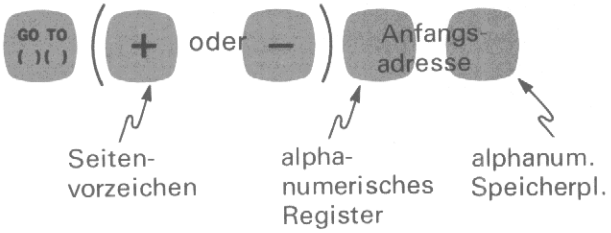
Aufzeichnung eines Programms (oder Daten) auf eine Magnetkarte.

Eingabe eines Programms mittels Magnetkarte.

Auslesen eines Programms.

Berichtigung eines Programms.

In diesem Abschnitt werden die Tasten-Symbole verwendet, um die Adressierung darzustellen.



Es wird entweder die (+) Taste oder die (–) Taste verwendet, abhängig davon, welche Seite adressiert werden soll.

Eingabe eines Programms über das Tastenfeld

1. Adressierung des Programmzählers:

   SCHALTEN: **RUN**

   DRÜCKEN: GO TO ( )( ) (+ oder –) Anfangs-Adresse

> **ANMERKUNG**
> Ist die Startadresse (+)00, so kann an Stelle von GO TO, + , 00 der Programmzähler mit END adressiert werden.

2. Eingabe des Programms:

   SCHALTEN: **PROGRAM**

   DRÜCKEN: Tasten in der Reihenfolge.

Bei der Eingabe jeden Schrittes zeigt das X-Register die Adresse und den Oktal-Code des Befehls an. Das Z-Register enthält den nächsten Programmschritt mit Adresse und Code-Ziffer. Der ursprüngliche Inhalt von Y bleibt unverändert.

BEISPIEL:

ANZEIGE:

0.1	17	→ z
	Inhalt Y	→ y
0.0	23	→ x

Adresse (+)00 enthält den Oktal-Code 23, die Code-Ziffer für die x→( )-Taste; die Adresse (+)01 enthält den Oktal-Code 17, die Code-Ziffer für die d-Taste.

3. Der Programmablauf:

SCHALTEN: **RUN**

DRÜCKEN: GO TO ( )( ) (+ oder −) Anfangs-Adresse

DRÜCKEN: END wenn die Start-Adresse (+)00

a. Sobald CONT gedrückt wird, beginnt der Rechner mit dem Programmbefehl in der Start-Adresse.
b. Dateneingabe und CONT sind zur Durchführung des Programms obligatorisch.

BEISPIEL:

Eingabe und Ablauf des Programms für $\frac{A \times B}{A + B}$ (Seite 66)

Eingabe des Programms:

SCHALTEN: **RUN**

Eingabe eines Programms über das Tastenfeld

DRÜCKEN: GO TO ( )( ) + 0 0

oder

DRÜCKEN: END

SCHALTEN: **PROGRAM**

DRÜCKEN: die Tasten in der Programmfolge

Es ist zu beachten, daß der eingegebene Schritt in X angezeigt wird. Im Z-Register kann jeder beliebige Befehl (etwa aus einem vorherigen Programm) stehen.

Adressierung des Programmzählers:

SCHALTEN: **RUN**

DRÜCKEN: GO TO ( )( ) 0 0

oder

DRÜCKEN: END

Eingabe der Daten (A und B):

DRÜCKEN: Zahlentasten für A

DRÜCKEN: ↑

DRÜCKEN: Zahlentasten für B

Programmablauf:

DRÜCKEN: CONT

ANZEIGE: $\frac{A \times B}{A + B} \rightarrow z$

$A \rightarrow y$

$B \rightarrow x$

Zur Wiederholung des Programms sind neue Werte einzugeben und CONT zu drücken. In diesem Programm muß der Programmzähler nicht vor jedem Ablauf neu adressiert werden, der END-Befehl setzt den Programmzähler automatisch auf (+) 00 zurück.

Die Magnetkarte hat zwei Seiten, A und B. Der Inhalt der einen Seite des Kernspeichers kann auf beiden Seiten einer Karte aufgezeichnet werden (mit Ausnahme von (±)e und (±)f und der Anzeigeregister X, Y und Z). Zur Aufzeichnung (oder Einlesen) des Teils A wird die Magnetkarte so in den Kartenleser eingeschoben, daß der Pfeil A nach unten zeigt und die bedruckte Seite vorne ist. Zur Aufzeichnung des Teils B wird die Karte entsprechend mit dem Pfeil B nach unten eingesteckt.

RECORD

ANMERKUNG

Sollen Daten nach Programmschritten aufgezeichnet werden, darf der END-Befehl nicht auf der gleichen Seite verwendet werden, da dieser Befehl die Aufzeichnung im Kernspeicher beendet und somit keine Daten eingelesen werden können.

Aufzeichnen eines Programms:

SCHALTEN: **RUN**

DRÜCKEN: GO TO ( )( ) (+ oder −) Startadresse

EINSETZEN: Magnetkarte in den Kartenleser (Pfeil A nach unten, wenn beide Seiten der Karte benutzt werden sollen).

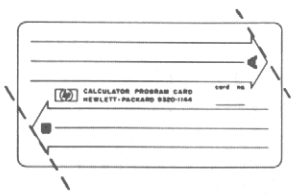
DRÜCKEN: RECORD (die Taste solange drücken, bis die Karte teilweise herausgelaufen ist). Werden beide Seiten verwendet, wird die Karte mit Pfeil B nach unten eingesetzt und nochmals RECORD gedrückt.

ANMERKUNG

Der Inhalt des X-Registers wird beim Aufzeichnen eines Programms zerstört.

Ein aufgezeichnetes Programm kann mit einem neuen Programm auf obige Weise überschrieben werden.

Zur Konservierung eines Programms wird die Ecke bei A oder B entlang der Pfeilspitze abgeschnitten.



Einlesen eines Programms von Magnetkarte

ENTER

SCHALTEN:  **RUN**

DRÜCKEN: GO TO ( )( ) (+ oder −) Start-adresse

EINSETZEN: Magnetkarte in den Kartenleser mit (Pfeil A oder B nach unten, mit A zuerst, wenn beide Seiten eingelesen werden sollen).

DRÜCKEN: ENTER

ANMERKUNG
Die Inhalte der Register X, Y und Z werden nicht verändert.

Ist die (+) Seite des Kernspeichers belegt, so wird der Programmzähler auf (–) 00 gesetzt, wenn kein END-Befehl auf der (+) Seite eingegeben wurde. END setzt den Programmzähler nach (+) 00.

Eingabe des B-Teils der Karte auf der (–) Seite:

(wenn nötig)

DRÜCKEN: GO TO ( )( ) − Start-adresse

Einsetzen der Karte mit Pfeil B nach unten und ENTER drücken.

Das Programm ist eingelesen und kann laufen.

Prüfen eines Programms

Ist ein Programm eingegeben, so kann es durch Betätigen der Step-Programm-Taste in „Run oder Programm" schrittweise geprüft werden.

1. In Program: Das Programm wird zur Überprüfung in der Reihenfolge der Eingabe mit Programmplatz und Codeziffer aufgerufen.

SCHALTEN:  **RUN**

DRÜCKEN: GO TO ( )( ) (+ oder −) Start-adresse

SCHALTEN: **PROGRAM** 

DRÜCKEN: STEP PRGM

Durch jeden Tastendruck STEP PROGRAM wird der Programmzähler um einen Schritt weitergeschaltet. Das X-Register zeigt den letzten Programmschritt mit Adresse und Oktal-Code an. Im Z-Register wird der nächste Programmschritt mit Adresse und Oktal-Code dieses Befehls angezeigt.

ANMERKUNG
Wird in PROGRAM eine andere Taste als STEP PRGM gedrückt, so wird dieser Befehl in das Programm eingelesen.

Beispielsweise kann das Programm auf Seite 66 zum schrittweisen Programmdurchlauf mit den Adressen und den Codeziffern benutzt werden. Soll der Programmzähler neu adressiert werden, ist auf RUN zu schalten.

2. In RUN: Zur Prüfung des Programmablaufs mit Eingabewerten

SCHALTEN: **RUN**

DRÜCKEN: GO TO ( ) (+ oder −) Start-adresse

Für jeden Befehl wird STEP PRGM gedrückt, wobei Daten an der richtigen Stelle einzugeben sind, um die Rechenoperationen und deren Ergebnisse zu prüfen. Für jeden Programmschritt zeigen das X-, Y- und Z-Register die Ergebnisse an. In RUN arbeiten folgende Befehle nicht wie erwartet auf STEP PRGM:

CONT FMT IF $x<y$ IF $x=y$ IF $x>y$ IF FLAG

CONTinue muß durch STOP oder PAUSE ersetzt werden (siehe Berichtigung eines Programms) bevor das Programm in RUN mit STEP PRGM ausgelesen werden kann. Der Rechner setzt sonst auf CONT automatisch das Programm fort.

Durch Drücken von STEP PRGM liest der Rechner bei FMT diesen und den nächsten Befehl. Wird STEP PRGM nochmals gedrückt, wird der auf FMT folgende Befehl ausgeführt. Die bedingten Verzweigungsbefehle (IF FLAG, IF X<Y, IF X=Y und IF X>Y) veranlassen den Programm-

Prüfen eines Programms

zähler sofort zu verzweigen, wenn die geforderte Bedingung erfüllt ist (siehe BEDINGTE VERZWEIGUNG-Tasten, Seite 83).

Beispielsweise könnte der folgende Ablauf Teil eines Programmes sein, das in RUN ausgegeben werden soll:

Schritt	Taste
(+)03	+
08	IF $x=y$
09	0
0*a*	3
0*b*	×

Wird bei Schritt 08 STEP PRGM gedrückt und - wie erwartet - der Wert des X-Registers ist gleich dem Wert im Y-Register, so muß STEP PRGM bei Schritt 09 und 0a gedrückt werden, bevor der Programmzähler nach Adresse 03 springt.

Sind die Werte in X und Y nicht gleich, so läuft der Programmzähler durch Drücken von STEP PRGM automatisch zu Schritt 0b und multipliziert die Werte in X und Y.

Die anderen IF-Befehle arbeiten in der gleichen Weise.

BEISPIEL:

Ausgeben des $\frac{A \times B}{A + B}$ Programms (Seite 66) in RUN:

Adressieren des Programmzählers

SCHALTEN: **RUN**

DRÜCKEN: GO TO ( )( ) + 0 0

oder

DRÜCKEN: END

Eingeben der Werte

DRÜCKEN: irgendein Wert für A.

DRÜCKEN: ↑

DRÜCKEN: irgendein Wert für B.

Schrittweiser Programmablauf durch Drücken der STEP PRGM-Taste: Wie im Programm beschrieben, enthält das X-, Y- und Z-Register bei jedem Schritt die jeweiligen Ergebnisse.

Programm-Berichtigung

Jeder Programmschritt kann geändert oder gelöscht werden ohne das gesamte Programm erneut eingeben zu müssen.

SCHALTEN:  **RUN**

DRÜCKEN: GO TO ( )( ) (+ oder −) Adresse des zu ändernden Programmschritts

SCHALTEN: **PROGRAM** 

Der zu ändernde Programmschritt erscheint im Z-Register.

DRÜCKEN: richtige Befehlstaste oder zum Löschen CONTinue.

Der richtige Programmschritt erscheint im X-Register.

SCHALTEN:  **RUN**

BEISPIEL:

Änderung des $\frac{A \times B}{A + B}$ -Programms (Seite 66) in $\frac{A \times B}{A - B}$

SCHALTEN:  **RUN**

DRÜCKEN: GO TO ( )( ) + 0 7

SCHALTEN: **PROGRAM** 

DRÜCKEN: −

SCHALTEN:  **RUN**

DRÜCKEN: END

Der Programmablauf bleibt der gleiche, die Lösung jedoch entspricht der Gleichung $\frac{A \times B}{A - B}$; ebenso erscheint jeweils im X-, Y- und Z-Register A - B an Stelle von A + B.

Programm-Berichtigung

Ebenso können Programmschritte nachträglich eingeführt werden. Ist das Programm kurz, dann ist die manuelle Eingabe, beginnend mit dem einzufügenden Programmschritt, schneller. Bei einem langen Programm kann mittels Magnetkarte oder der nachfolgend beschriebenen „Flick-Methode" verfahren werden (nach dem Beispiel).

BEISPIEL:

Die Programmschritte unten sind als Teil eines langen Programms gedacht, in das weitere Programmbefehle (z. B. ROLL ↑ und |y|) zwischen Schritt 64 und 65 eingefügt werden sollen.

Schritt	Taste	Code-Zahl
(+)62	ACC +	60
63	↑	27
64	RCL	61
65	+	33
66	$\sqrt{x}$	76
67	END	46

Zunächst Eingabe des Programms in den Rechner:

SCHALTEN: **RUN**

DRÜCKEN: GO TO ( )( ) + 6 2

SCHALTEN: **PROGRAM**

DRÜCKEN: Die Tasten in der Programmfolge.

Zur Berichtigung des Programms:

Aufzeichnen des Programms auf eine Magnetkarte, beginnend bei (+) 65:

SCHALTEN: **RUN**

DRÜCKEN: GO TO ( )( ) + 6 5

EINSETZEN: Magnetkarte in Leser

DRÜCKEN: RECORD

Eingabe der zwei zusätzlichen Befehle als 65 und 66:

DRÜCKEN: GO TO ( )( ) + 6 5

SCHALTEN: **PROGRAM**

DRÜCKEN: ↑ ROLL |y|

Wiedereingabe der restlichen Schritte (auf der Magnetkarte aufgezeichnet) auf 67 und folgende:

SCHALTEN: **RUN**

EINSETZEN: Magnetkarte in Leser

DRÜCKEN: ENTER

Die Programmschritte auf der Magnetkarte werden automatisch im Kernspeicher um zwei Plätze verschoben. Die Befehlsfolge dieses Beispiels lautet jetzt:

Schritt	Taste	Code-Zahl
(+)62	ACC +	60
63	↑	27
64	RCL	61
65	ROLL ↑	22
66	|y|	55
67	+	33
68	$\sqrt{x}$	76
69	END	46

Enthält das Programm einen Verzweigungsbefehl, müssen die alphanumerischen Befehle entsprechend den geänderten Adressen berichtigt werden.

Bei der „Flick-Methode" ist keine Änderung der Adressen nach Verzweigungsbefehlen nötig. Wie im vorigen Beispiel sollen

zwei Befehle zwischen Schritt 64 und 65 eingefügt werden (ROLL ↑ and |y|)

Schritt	Taste	Code-Zahl
(+)62	ACC +	60
63	↑	27
64	RCL	61
65	+	33
66	$\sqrt{x}$	76
67	END	46

Eingabe der Schritte:

1. Eingabe (sinnvolle) Verzweigungsadresse als Schritte vor 65.

2. Auf den Speicherplätzen der neuen Adresse werden zunächst die gelöschten Befehle, dann die einzufügenden Befehle und dann der Verzweigungsbefehl zu Speicherplatz (+) 65 eingegeben.

Schritt	Taste	Code-Zahl
(+)62	GO TO ( )( )	44
63	7	07
64	0	00
65	+	33
66	$\sqrt{x}$	76
67	END	46

Schritt	Taste	Code-Zahl
(+)70	ACC +	60
71	↑	27
72	RCL	61
73	ROLL ↑	22
74	$\|y\|$	55
75	GO TO ( )( )	44
76	6	06
77	5	05

Bedingte Verzweigungen

Vier Tasten auf der rechten Seite des Tastenfeldes werden für bedingte Verzweigungen verwendet (siehe auch Seite 59). Diese sind: IF x<Y, IF x=Y, IF x>Y und IF FLAG (in Verbindung mit SET FLAG).

IF $x<y$

IF $x=y$

IF $x>y$

Diese drei Befehle vergleichen die Werte im X- und Y-Register. Die Bedingung (z. B. ist X gleich Y?) wird getestet. Ist die Bedingung erfüllt, führt das Programm die beiden nächsten Schritte aus. Ist die Bedingung nicht erfüllt, überspringt das Programm die beiden nächsten Schritte und läuft weiter.

IF $x<y$

IF $x=y$

IF $x>y$

ANMERKUNG

Die IF-Befehle vergleichen alle zwölf (12) Stellen und den zweistelligen Exponenten der Zahlen im X- und Y-Register auf die eingegebene Bedingung. Besteht jedoch der Hauptwert einer Zahl in allen zwölf Stellen aus der Zahl neun (9), so wird diese Zahl als gleich der nächsten Potenz zu Zehn (10) angesehen (z. B. $9.999\ 999\ 999\ 99 \times 10^2 = 10^3$).

BEDINGUNG ERFÜLLT: Ist die Bedingung eines IF-Tests erfüllt, liest der Rechner ein GO TO. Sind die auf den IF-Befehl folgenden Schritte eine Adresse, so verzweigt das Programm zu dieser Adresse. Enthalten die Schritte jedoch Befehle und keine Adresse, so werden diese Befehle ausgeführt.

Folgt auf einen IF-Befehl einer der nachfolgend aufgeführten Befehle, bedeutet dies den Beginn einer Adressierung, so daß darauffolgend der Rest der Adresse stehen muß:

SUB/RETURN, (+) (–) und jeder alphanumerische Befehl.

IF $x<y$

IF $x=y$

IF $x>y$

ANMERKUNG

a) **Die Verwendung von (+) und (–) ist seitenabhängig (siehe GO TO-Taste, Seite 68) und deshalb nur bei Verzweigung von einer Seite zur anderen nötig.**

b) **Der SUB-Befehl bewirkt die Verzweigung zur Adresse des Unterprogramms.**

c) **Ein CONTinue kann nicht als Leerbefehl nach einem IF-Befehl programmiert werden, wenn einer der nachstehenden Befehle folgt:**

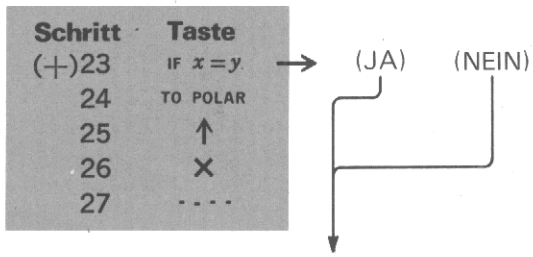
SUB/RETURN, (+), (–) oder ein alphanumerischer Befehl

Denn CONTinue löscht nicht den GO TO-Befehl bei erfüllter Bedingung eines IF-Tests.

Die nachfolgenden Beispiele 1 bis 4 sind Teile eines willkürlich gewählten Programms. Jedes Programm enthält eine Kombination möglicher Programmschritte, die auf einen IF-Befehl folgen können. Der Programmablauf ist für „Bedingung erfüllt" (JA) und „Bedingung nicht erfüllt" (NEIN) graphisch dargestellt.

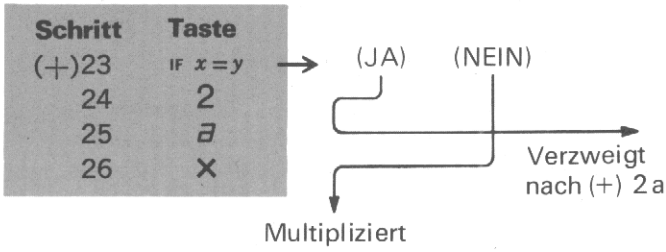
BEISPIEL (1)

Auf den IF-Befehl folgen BEFEHLE:



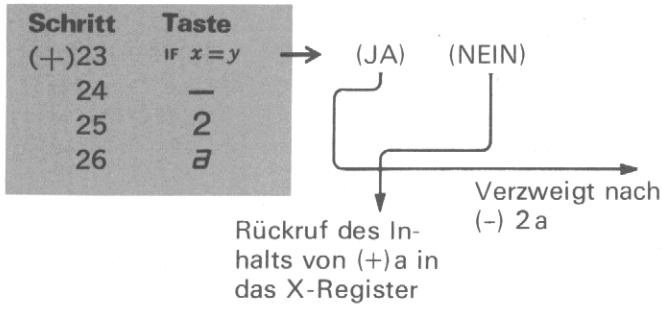
BEISPIEL (2)

Auf den IF-Befehl folgt ADRESSE:



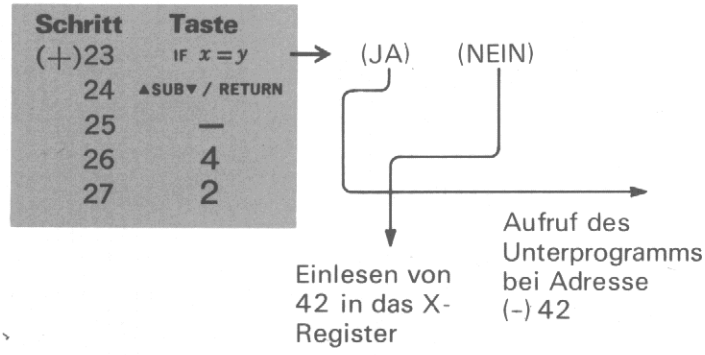
BEISPIEL (3)

Auf den IF-Befehl folgt ADRESSE MIT SEITENVORZEICHEN:



BEISPIEL (4)

Auf den IF-Befehl folgt SUB/RETURN:



Das folgende Programm zeigt die Verwendung eines IF-Befehls (in diesem Fall IF x <y):

dieses Programm addiert eine beliebige Zahl (N) auf, bis die Summe (Σn) gleich oder größer 100 ist:

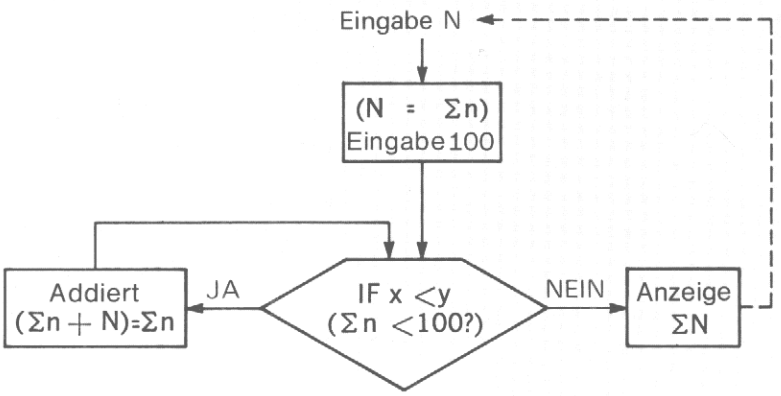
Die Endanzeige enthält N im Z-Register, 100 im Y-Register und ΣN im X-Register. „IF" x<y" wird zum Test ob Σn gleich oder größer 100 ist verwendet. Wenn N negativ ist, wird 100 nie erreicht und das Programm addiert - N bis zu Unendlich auf.

IF $x < y$

IF $x = y$

IF $x > y$

FLUSS-DIAGRAMM



PROGRAMM

Zum Programmablauf END drücken (nur das erste Mal), Eingabe N und CONTinue drücken.

Schritt	Taste	Code-Zahl	Anzeige X	Anzeige Y	Anzeige Z
(+)00	↑	27	N	N	- -
01	↑	27	N	N	(N)=Σn
02	ENTER EXP	26	1	N	Σn
03	2	02	100	N	Σn
04	ROLL ↑	22	Σn	100	N
05	PAUSE	57	Σn	100	N *
06	IF $x<y$	52	Σn	100	N
07	0	00	Σn	100	N
08	a	13	Σn	100	N
09	END	46	Σn=ΣN	100	N †
0a	ROLL ↑	22	N	Σn	100
0b	+	33	N	(Σn + N)=Σn	100
0c	ROLL ↓	31	Σn	100	N
0d	GO TO ( )( )	44	Σn	100	N
10	0	00	Σn	100	N
11	5	05	Σn	100	N

* Zwischenanzeige nach Pause
† End-Anzeige

Ist bei Schritt 06 Σn kleiner als 100 und damit die abgefragte Bedingung erfüllt, läuft das Programm mit Schritt 07 und 08 weiter. Diese enthalten eine Adresse und das Programm verzweigt nach (+) 0a. Ist Σn nicht kleiner als 100 (z. B. $\Sigma n = \Sigma N \geq 100$) so ist die Bedingung nicht erfüllt, die Schritte 07 und 08 werden dann übersprungen und der Befehl auf Speicherplatz 09 ausgeführt.

Das vorstehende Programm zeigt außerdem folgendes:

a) CLEAR ist nicht unbedingt erforderlich;
b) Die Verwendung von PAUSE zur Anzeige von Teilergebnissen des Programms;
c) Manipulation der Inhalte des X-, Y- und Z-Registers, so daß keine Speicher-Register benötigt werden;
d) Die Verwendung von END bei Schritt 09 um den Programmzähler automatisch nach (+) 00 zu setzen. Soll das Programm auf Magnetkarte aufgezeichnet werden, ist END durch STOP zu ersetzen.
e) Das Seitenvorzeichen wird bei der Verzweigung nicht benötigt (Schritt 07, 08 und 10, 11), da die Seite des Kernspeichers nicht gewechselt wird.

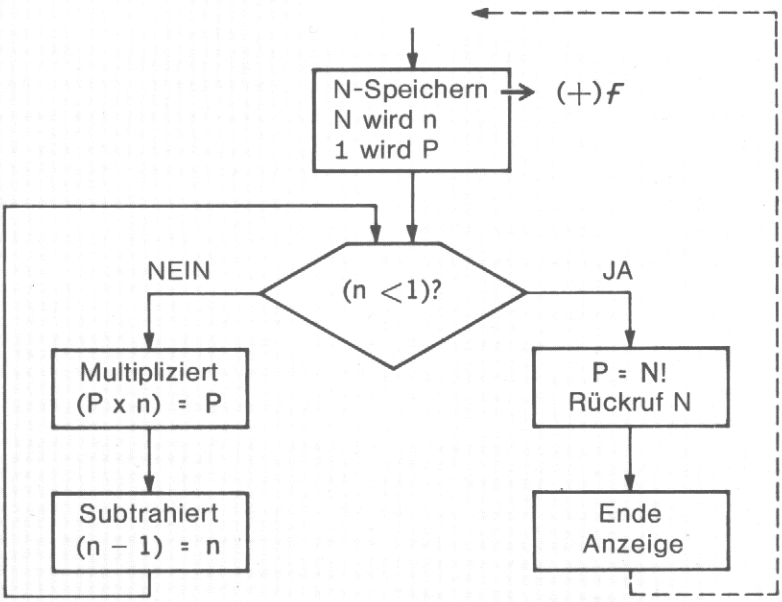
Das folgende Programm errechnet N! Ein neuer Wert für N kann zu Ende des Programms eingegeben werden, da END den Programmzähler nach (+) 00 zurücksetzt. Dieses Programm errechnet N! Für alle Werte von N bis einschließlich 69.

$$N! = N\,(N-1)\,(N-2)\cdot\cdots\cdot(N-N+2)\,(1)$$

$$6! = (6)\,(5)\,(4)\,(3)\,(2)\,(1) = 720$$

$$0! = 1 \text{ (lt. Definition)}$$

(Im Programm) N = Zahleneingabe
n = der laufende Multiplikationsausdruck N, (N-1), (N-2), usw.
P = die Teil-Fakultät N!



IF $x<y$

IF $x=y$

IF $x>y$

PROGRAMM

Zum Programmablauf END drücken (nur das erste Mal), Eingabe N und CONTinue drücken.

Schritt	Taste	Code-Zahl	Anzeige X	Anzeige Y	Anzeige Z	Speicher (+)f
(+)00	x→( )	23	N	-	-	-
01	f	15	N	-	-	N
02	↑	27	N	N = n	-	
03	1	01	1	n	-	
04	↑	27	1	1	n	
05	ROLL ↓	31	1	n	1 = P	
06	IF x>y	53	1	n	P	
07	1	01	1	n	P	
08	2	02	1	n	P	
09	ROLL ↓	31	n	P	1	
0a	×	36	n	(P x n) = P	1	
0b	ROLL ↑	22	1	n	P	
0c	−	34	1	(n − 1) = n	P	
0d	GO TO ( )( )	44	1	n	P	
(+)10	0	00	1	n	P	
11	6	06	1	n	P	Anzeige des Ergebnisses
12	f	15	N	n = 0	P = N!	
13	END	46	N	0	N!	

IF FLAG

SET FLAG

IF FLAG ist ein bedingter Verzweigungsbefehl, der eine „JA" oder „NEIN" Bedingung prüft, die im Rechner gespeichert ist. Die Bedingung wird durch den SET FLAG-Befehl erfüllt, entweder von Hand über das Tastenfeld oder als Programmschritt. Die Verzweigung nach einem IF FLAG-Befehl ist die gleiche wie sie bei den bedingten Verzweigungen (IF x<y, IF x=y und IF x>y) bereits beschrieben wurde.

Bedingung erfüllt FLAG GESETZT (JA) - das Programm läuft mit den beiden Schritten weiter, die auf den IF FLAG-Befehl folgen, und löscht die Marke (SET FLAG) zur NEIN-Bedingung.

BEDINGUNG NICHT ERFÜLLT: FLAG NICHT GESETZT (NEIN) - das Programm überspringt die beiden, dem IF FLAG-Befehl folgenden Programmschritte und führt den dritten Befehl aus. (Siehe SET FLAG als Beispiel für die Verwendung des IF FLAG-Befehls).

SET FLAG

SET FLAG setzt die „Ja"-Bedingung, die durch den IF FLAG-Befehl getestet wird. Kann entweder von Hand über das Tastenfeld bei angehaltenem Programm oder per Programmbefehl gesetzt werden.

Die „JA"-(SET FLAG)-Bedingung wird gelöscht (zu „NEIN"), sobald sie im Programm den IF FLAG-Befehl vorfindet, so daß der nächste IF FLAG-Test „NEIN" ergibt. Ebenso löscht CLEAR die SET FLAG-Bedingung von Hand bei angehaltenem Programm oder per Programmschritt.

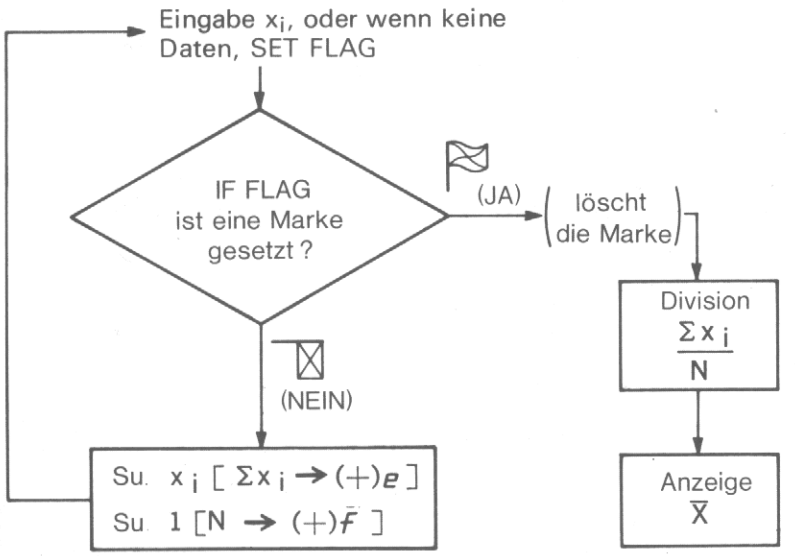
Das nachfolgende Programm errechnet den Durchschnittswert ($\bar{X}$) von N-Datenpunkten, x_i:

$$\frac{x_1 + x_2 + x_3 + \cdots x_n}{N} = \frac{\Sigma x_i}{N} = \bar{X}$$

Dieses Programm zeigt folgendes:

a) die Verwendung von IF FLAG wenn SET FLAG über das Tastenfeld eingegeben wird;
b) die Entwicklung eines Programmzählers (die Anzahl der Datenpunkte wird durch Addition von jeweils 1 bei Eingabe eines Datenpunktes weitergezählt);
c) die Verwendung der akkumulativen Speicher-Register (+) e und (+) f;
d) die Verwendung von CLEAR zum Löschen von (+) e und (+) f sowie SET FLAG.

FLUSS-DIAGRAMM



IF FLAG

SET FLAG

Programmablauf:

1) DRÜCKEN: END CONT

2) Eingabe x_1 in das X-Register

3) DRÜCKEN: CONT

4) Eingabe x_2 in das X-Register

5) DRÜCKEN: CONT

6) Wiederholung der Schritte 4 und 5 für alle restlichen Datenpunkte bis einschließlich x_n

7) DRÜCKEN: SET FLAG CONT

END-ANZEIGE: $\bar{X} \rightarrow y$
$N \rightarrow x$

Schritt	Taste	Code-Zahl	Anzeige X	Anzeige Y	Anzeige Z	Speicher (+)f	Speicher (+)e
(+)00	CLEAR	20	0	0	0	0	0
01	STOP	41	[Eingabe x_i]				
02	IF FLAG	43					
03	0	00					
04	b	14					
05	↑	27	x_i	x_i	(x_{i-1})		
06	1	01	1	x_i	(x_{i-1})		
07	ACC +	60	1	x_i	(x_{i-1})	N+1=N	Σx_i
08	GO TO ( )( )	44					
09	0	00					
0a	1	01					
0b	RCL	61	N	Σx_i	(x_{i-1})	N	Σx_i
0c	÷	35	N	$\frac{\Sigma x_i}{N}$	(x_{i-1})		
0d	END	46	N	$\bar{X}$	(x_{i-1})	→Endanzeige	

Das nächste Programm veranschaulicht den Gebrauch von SET FLAG als Programmschritt. Das Programm berechnet eine wechselnde Reihe:

$$1 - 3 + 5 - 7 + 9 \cdots\cdots\cdots\cdots \infty$$

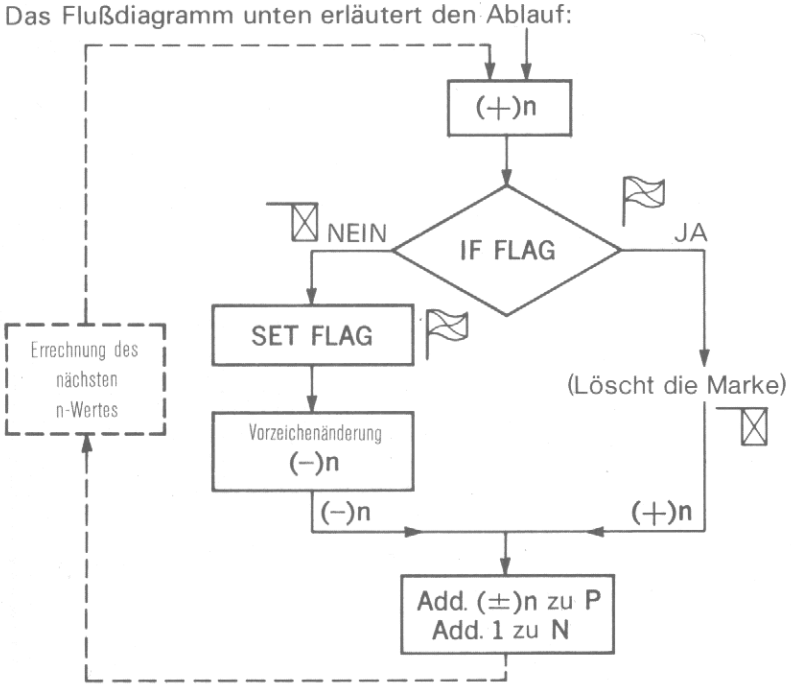
ANMERKUNG

Es wurde ein Programm mit einfachen Ausdrücken zur Erklärung von IF FLAG und SET FLAG gewählt, um die Verzweigung von einer Seite der Schleife zur anderen anschaulich zu machen. Der Ablauf der Verzweigung kann jedoch auch auf komplexere Ausdrücke leicht erweitert werden.

In diesem Programm wurde jeder Ausdruck (n) entwickelt und zu einer akkumulativen Summe hinzuaddiert. Dabei wurde der Programmzähler jedesmal um 1 erhöht, um die Zahl der aufaddierten Ausdrücke (N) zu erhalten.

Das Vorzeichen von n muß sich abwechselnd ändern. Die Marke entscheidet, ob n positiv bleibt oder negativ wird.

Das Flußdiagramm unten erläutert den Ablauf:



Ist die Marke gesetzt:

(a) positives (+) n wird zu P addiert;
(b) die Marke ist gelöscht.

Beim nächsten Durchlauf ist keine Marke gesetzt:

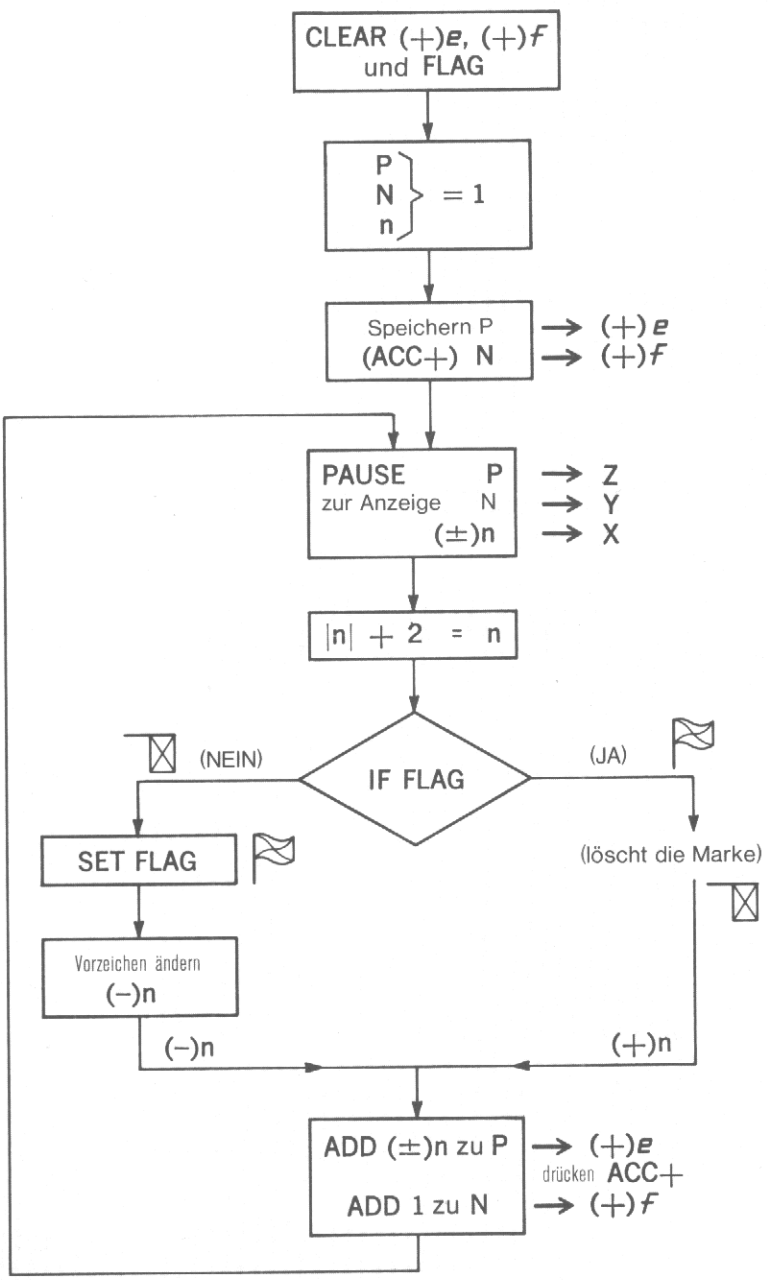
(a) negatives (−) n wird zu P addiert;
(b) die Marke wird gesetzt.
(so daß beim nächsten Schleifendurchlauf durch
(a) der Wert (+) n zu P addiert wird).

IF FLAG

SET FLAG

FLUSSDIAGRAMM: Für Reihen $1 - 3 + 5 - 7 + 9 \cdots\cdots \infty$

n = Glieder der Reihe
N = Zahl der Glieder
P = Akkumulative Summe der Reihe



Zum Ablauf des Programms:

DRÜCKEN: END CONT

Zum Anhalten und Anzeigen von P, N und n:

DRÜCKEN: PAUSE ; CONT zum Weiterlauf des Programms drücken.

Um das Programm neu zu starten:

DRÜCKEN: PAUSE (oder STOP) END CONT

Schritt	Taste	Code-Zahl	Anzeige X	Anzeige Y	Anzeige Z	Speicher (+)f	Speicher (+)e
(+)00	CLEAR	20	0	0	0	0	0
01	1	01	1=P	0	0	↓	↓
02	↑	27	1=N	P	0		
03	ACC +	60	N	P	0	N	P
04	↑	27	1=n	N	P		
05	PAUSE	57	Anzeige				
06	PAUSE	57	±n	N	P		
07	↑	27	±n	±n	N		
08	|y|	55	±n	|n|=n	N		
09	2	02	2	n	N		
0a	+	33	2	n+2=n	N		
0b	↓	25	n	N	N		
0c	IF FLAG	43					
0d	1	01					
(+)10	3	03					
11	SET FLAG	54					
12	CHG SIGN	32	−n	N	N		
13	↑	27	±n	±n	N		
14	1	01	1	±n	N		
15	ACC +	60	1	±n	N	N+1=N	P+(±)n=P
16	↑	27	1	1	±n	N	P
17	RCL	61	N	P	±n		
18	ROLL ↑	22	±n	N	P		
19	GO TO ( )( )	44					
1a	0	00					
1b	5	05					

Unterprogramme

Bewirkt die unbedingte Verzweigung zu einem Unterprogramm, dessen Ausführung und Rücksprung in das Hauptprogramm (siehe auch Seite 60).

Zum Aufrufen (Verzweigen) eines Unterprogramms wird folgende Schrittfolge verwendet:

ANMERKUNG

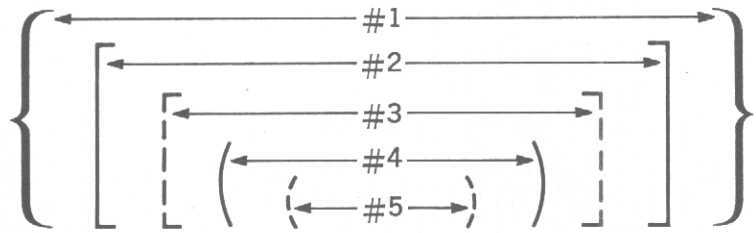
Die Verwendung von (+) und (–)-Vorzeichen ist seitenabhängig; siehe GO TO auf Seite 68.

Rücksprung vom Unterprogramm:
SUB/RETURN muß der letzte Befehl des Unterprogramms sein. Ist das Unterprogramm vollständig abgearbeitet, kehrt das Programm automatisch zu dem Programmschritt zurück, der unmittelbar auf die Adressierung des Unterprogramms folgt.

ANMERKUNG

Folgt auf den GO TO-Befehl SUB und Adresse, so speichert der Rechner intern die Adresse, zu der er nach Abarbeiten des Unterprogramms zurückspringen muß. Wird der Programmzähler über das Tastenfeld zur Startadresse des Unterprogramms adressiert und CONTinue gedrückt, springt der Rechner nach Beendigung des Unterprogramms nicht zur richtigen Adresse zurück.

VERSCHACHTELN VON UNTERPROGRAMMEN: Unterprogramme können auch während des Ablaufs eines Unterprogramms aufgerufen werden. Dies wird als Verschachteln bezeichnet. Ein Programm kann jede Anzahl Unterprogramme enthalten (durch Speicherkapazität begrenzt); jedoch können nicht mehr als fünf Unterprogramme verschachtelt werden. Nachfolgende Abbildung veranschaulicht das Verschachteln der Unterprogramme 1 bis 5.



Beim Aufruf eines Unterprogramms speichert der Rechner die Rücksprung-Adresse. Bis zu fünf Rücksprung-Adressen können gleichzeitig gespeichert werden. Nach dem Rücksprung wird diese Adresse „vergessen"; der Rechner ist zum Speichern einer neuen Rücksprung-Adresse bereit.

In der vorherigen Abbildung ist im Verlauf des Unterprogramms No. 1 die Zahl der weiteren Unterprogramme auf vier begrenzt (# 2 bis 5) weil die Unterprogramme 1 bis 5 „verschachtelt" sind. Es könnten jedoch im Unterprogramm 1 weitere Unterprogramme aufgerufen werden, da nach dem Rücksprung aus 4 die Rücksprung-Adressen von 4 und 5 vergessen werden. Jetzt ist für zwei weitere Unterprogramme Platz.

Zusammenfassung:

1. Es kann eine beliebige Anzahl Unterprogramme in einem Hauptprogramm verwendet werden.
2. In einem Unterprogramm kann eine beliebige Anzahl weiterer Unterprogramme verwendet werden, jedoch
3. können nicht mehr als fünf Unterprogramme gleichzeitig in einem Programm gefahren werden.

Der folgende Programmausschnitt ist ein Teil eines Programms (auf Seite 97), in dem Unterprogramme benutzt werden.

Schritt	Taste
(+)17	↓
18	GO TO ( )( )
19	▲SUB▼ / RETURN
1a	2
1b	4
1c	y⇄( )

Schritt	Taste
(+)24	x→( )
38	PAUSE
39	PAUSE
3a	▲SUB▼ / RETURN
3b	-----

Bei Schritt (+) 1b verzweigt das Programm zur Startadresse (+) 24 des Unterprogramms und führt es aus; bei Schritt (+)3a springt das Unterprogramm zur Adresse (+) 1c zurück in das Hauptprogramm.

Das folgende Programm erläutert die Verwendung eines Unterprogramms; das N-Programm ist modifiziert als Unterprogramm verwendet.

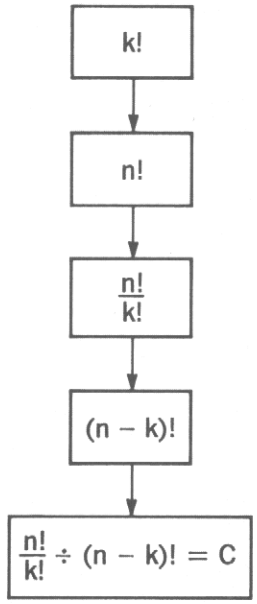
Das Programm berechnet mögliche Kombinationen (C) von n Objekten für k Fälle.

Zum Beispiel: Eine Kiste enthält (n) 15 verschiedenfarbige Bälle, wieviel Farbkombinationen (C) sind möglich, wenn jeweils 5 Bälle (K) herausgenommen (und wieder hineingelegt) werden. Antwort: 3003

▲SUB▼
RETURN

Die Formel lautet: $C\,{}^{n}_{k} = \frac{n!}{k!\,(n-k)!}$

VEREINFACHTES FLUSS-DIAGRAMM:



ENDGÜLTIGES FLUSS-DIAGRAMM:

Programmablauf: END drücken (nur das erste Mal), CONT; dann Eingabe n und k und CONT drücken.

Schritt	Taste	Code-Zahl	Anzeige X	Anzeige Y	Anzeige Z
(+)00	CLEAR	20	Eingabe		
01	STOP	41	k	n	
02	ACC +	60	k	n	
03	GO TO ( )( )	44			
04	▲SUB▼ / RETURN	77			
05	2	02			
06	4	04			
07	$x\rightarrow$( )	23	k!		
08	d	17			
09	e	12	n		
0a	GO TO ( )( )	44			
0b	▲SUB▼ / RETURN	77			
0c	2	02			
0d	4	04			
(+)10	↑	27	n!	n!	
11	d	17	k!	n!	
12	÷	35	k!	n!/k!	
13	$y\rightarrow$( )	40			
14	d	17			
15	RCL	61	k	n	
16	−	34	k	n − k	
17	↓	25	n − k		
18	GO TO ( )( )	44			
19	▲SUB▼ / RETURN	77			
1a	2	02			
1b	4	04			
1c	$y\rightleftarrows$( )	24	(n−k)!		
1d	d	17	(n−k)!	n!/k!	
(+)20	÷	35		C	
FORTSETZUNG					

Speicher	
(+)f	Ø, k
(+)e	Ø, n
(+)d	k!, n!/k!
(+)c	N
(+)b	

Schritt	Taste	Code-Zahl	Anzeige X	Anzeige Y	Anzeige Z
21	↑	27			C
22	RCL	61	k	n	C
23	END	46	k	n	C
			END Anzeige		
	In STOP (41) ändern bei Aufzeichnung auf Magnetkarte				
(+)24	x→( )	23	N		
25	C	16			
26	↑	27			
27	1	01			
28	↑	27			
29	ROLL ↓	31			
2a	IF x>y	53			
2b	3	03			
2c	6	06			
2d	ROLL ↓	31			
(+)30	×	36			
31	ROLL ↑	22			
32	−	34			
33	GO TO ( )( )	44			
34	2	02			
35	a	13			
36	C	16	N	0	N!
37	ROLL ↑	22	N!	N	0
38	PAUSE	57	Anzeige blinkt auf		
39	PAUSE	57	N!	N	0
3a	▲SUB▼ / RETURN	77			

Weitere Programmier-Hinweise

Bestmögliche Ausnutzung des Kernspeichers

Nachfolgend einige Richtlinien zur bestmöglichen Ausnutzung des Kernspeichers:

ANMERKUNG

Programmschritte und Daten können nicht gleichzeitig in einem Register gespeichert werden.

Datenspeicherung auf der (+) Seite erfordert weniger Programmschritte beim Speichern und Rückruf als auf der (-) Seite.

Die Speicher (±)e und (±)f werden nur als Datenspeicher benutzt.

Die Inhalte der Speicher (±)e und (±)f (und X, Y und Z) können nicht auf Magnetkarte aufgezeichnet werden.

1) Im allgemeinen beginnt das Programm bei Adresse (+)00 und füllt die (+) Seite, springt dann zu Adresse (-)00 und füllt die (-) Seite.

2) Die Datenspeicherung beginnt auf der (+) Seite mit (+)f bis (+)a, in dieser Folge. Danach werden (-)e und (-)f belegt, bevor die numerischen Speicher (+)9, (+)8 usw. belegt werden. (+)e und (+)f sollten bei Bedarf für ACC+, ACC- und RCL (z.B. bei Laufanweisungen) reserviert werden.

3) Erfordert ein Programm das Speichern von Daten über eine ganze Seite, so sollte das Programm bei Adresse (-)00 beginnen um die (+) Seite mit Daten belegen zu können.

4) Lassen Programmschritte nicht mehr genug Speicherplatz für Datenspeicherung, so kann „selbst-zerstörend" verfahren werden: Werden Programmschritte im Programmablauf nur einmal benötigt, so kann z.B. bei Adresse (+)a0 begonnen werden, die nach Durchlauf mit Daten belegt wird. Dieses Verfahren bedingt die Aufzeichnung auf Magnetkarte, da das Programm bei jedem neuen Ablauf mit Magnetkarte neu eingelesen werden muß.

5) Ist ein Programm für eine Magnetkarte zu lang, können mehrere Magnetkarten nacheinander eingegeben werden, wobei jeweils das Programm der vorhergehenden Karte durchlaufen wird und dann die nächste Karte eingelesen wird.
Die Eingabe erfolgt über das Tastenfeld in folgender Reihenfolge: Eingabe Tastenfeld Teil 1, Aufzeichnung auf Magnetkarte 1, Teil A; Eingabe Tastenfeld Teil 2, Aufzeichnung auf Magnetkarte 1, Teil B; Eingabe Tastenfeld Teil 3, Aufzeichnung auf Magnetkarte 2, Teil A, usw. bis das Programm vollständig auf Magnetkarten aufgezeichnet ist. Zum Programmablauf wird entsprechend verfahren.
Es ist besonders darauf zu achten, daß die gespeicherten

Daten nicht zerstört werden. Der END-Befehl kann verwendet werden. Dadurch stoppt der Kartenleser das Einlesen in den Kernspeicher an der gewünschten Stelle. Sind alle Daten auf der (+) Seite und alle Programmschritte auf der (–) Seite gespeichert, ist gewährleistet, daß keine Überschreibung der Daten mit Programmschritten erfolgen kann.

Periphere Geräte

Sollen später periphere Geräte angeschlossen werden, ist es zweckmäßig, dies beim Programmieren bereits zu berücksichtigen. An den gewünschten Stellen kann CONT als Programmschritt ohne Funktion programmiert werden. Dies ist für den Programmablauf unerheblich, kann aber später durch den Steuerbefehl für das betreffende periphere Gerät ersetzt werden, ohne daß eine Änderung der Adressen im Programm notwendig wird.

Für den Drucker 9120A wird CONT an der Stelle programmiert, an der später der PRINT-Befehl stehen soll:
PRINT/SPACE druckt die auf dem Drucker angewählte Kombination mit den Inhalten des X-, Y- und Z-Registers;
darauf folgende PRINT/SPACE-Befehle ergeben den Vorschub nach dem Ausdrucken.

Für die Ansteuerung des X-Y-Schreibers 9125A sollte ein Unterprogramm vorgesehen werden. Dafür müßten im Hauptprogramm fünf aufeinanderfolgende CONT programmiert werden, die durch GO TO, SUB, (+ oder –) und Adresse ( ) ( ) zu ersetzen sind. Das Unterprogramm kann an beliebiger Stelle programmiert werden.

Daten nicht zerstört werden. Der END-Befehl kann verwendet werden. Dadurch stoppt der Kartenleser das Einlesen in den Kernspeicher an der gewünschten Stelle. Sind alle Daten auf der (+) Seite und alle Programmschritte auf der (–) Seite gespeichert, ist gewährleistet, daß keine Überschreibung der Daten mit Programmschritten erfolgen kann.

Periphere Geräte

Sollen später periphere Geräte angeschlossen werden, ist es zweckmäßig, dies beim Programmieren bereits zu berücksichtigen. An den gewünschten Stellen kann CONT als Programmschritt ohne Funktion programmiert werden. Dies ist für den Programmablauf unerheblich, kann aber später durch den Steuerbefehl für das betreffende periphere Gerät ersetzt werden, ohne daß eine Änderung der Adressen im Programm notwendig wird.

Für den Drucker 9120A wird CONT an der Stelle programmiert, an der später der PRINT-Befehl stehen soll:
PRINT/SPACE druckt die auf dem Drucker angewählte Kombination mit den Inhalten des X-, Y- und Z-Registers;
darauf folgende PRINT/SPACE-Befehle ergeben den Vorschub nach dem Ausdrucken.

Für die Ansteuerung des X-Y-Schreibers 9125A sollte ein Unterprogramm vorgesehen werden. Dafür müßten im Hauptprogramm fünf aufeinanderfolgende CONT programmiert werden, die durch GO TO, SUB, (+ oder –) und Adresse ( ) ( ) zu ersetzen sind. Das Unterprogramm kann an beliebiger Stelle programmiert werden.

Sparen von Programmschritten

Eingabe einer Konstanten

Nachfolgend einige Tricks zum Einsparen von Programmschritten:
Wird im Programm eine Konstante benötigt, so kann diese als Programmschritt eingegeben werden. Dadurch spart man einen Teil des Speicher-Registers. Dies ist besonders in einem Unterprogramm sinnvoll.

Schritt	Taste
22	ROLL ↑
23	2
24	•
25	1
26	3
27	

nach Schritt 26 steht 2,13 im X-Register.

Potenzen von 10 können mit ENT EXP eingegeben werden:

Schritt	Taste
22	↑
23	ENTER EXP
24	4
25	

nach Schritt 24 steht 1×10^4 (10,000) im X-Register.

Multiplikation x 2

Für Multiplikation mit 2 kann der (+)-Befehl verwendet werden:

Schritt	Taste
34	π
35	↑
36	+
37	

nach Schritt 36 steht 2π im X-Register.

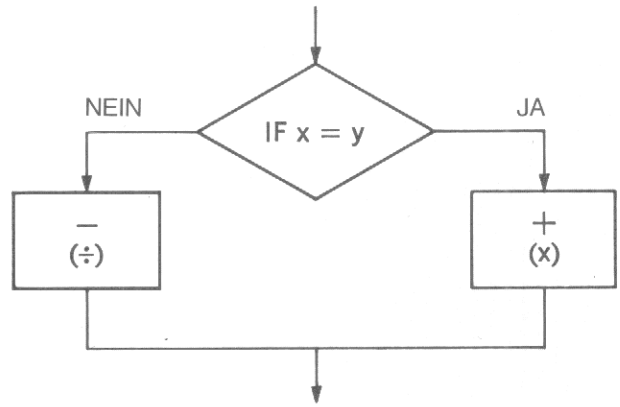
$\sqrt{\text{Summe zweier Quadrate}}$

Zur Berechnung der Wurzel aus der Summe zweier Quadrate ($\sqrt{A^2 + B^2}$):

	Taste
Eingabe A	( )
	↑
Eingabe B	( )
	TO POLAR

$\sqrt{A^2 + B^2}$ erscheint im X-Register. TO POLAR rechnet rechtwinklige Koordinaten in Polar-Koordinaten um, so daß der Winkel (θ) in Grad oder Bogenmaß im Y-Register erscheint. Um bedingt zu addieren oder zu subtrahieren (multiplizieren oder dividieren):

Bedingte Operationen



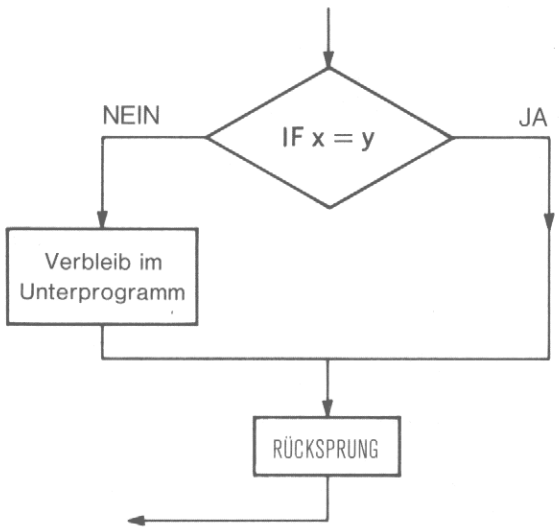
Schritt	Taste
61	. . . .
62	IF $x=y$
63	ACC + (or) ×
64	ACC + (or) ×
65	ACC − (or) ÷

wenn x = y, ergeben Schritte 63-65 eine Addition (Multiplikation);

wenn $x \neq y$, werden Schritt 63 und 64 übersprungen und subtrahiert.

Das (+)- und (−)-Vorzeichen kann nicht verwendet werden, da dies als Anfang einer Verzweigungs-Adresse gelesen würde.

Bedingter Rücksprung aus einem Unterprogramm:



Schritt	Taste
. . .	. . . .
75	IF x=y
76	9
77	2
78	. . . .
. . .	. . . .
92	▲SUB▼ / RETURN

Anhang

Bereich der Argumente

Bereich der Argumente für trigonometrische, hyperbolische, logarithmische und exponentielle Funktionen.

sin x, cos x, tan x	$\|x\| \leq 1 \times 10^{10}$
arcsin x	$\|x\| \leq 1$
arccos x	$\|x\| \leq 1$
arctan x	any x
sinh x	$\|x\| < 230.25$
cosh x	$\|x\| < 230.25$
tanh x	$\|x\| < 230.25$
arcsinh x	any x
arccosh x	$\|x\| \geq 1$
arctanh x	$\|x\| < 1$
ln x	$x > 0$
Log x	$x > 0$
e^x	$-227.95 < x < 230.25$

Bereich des 9100B 9.999 999 999 x 10^{99} to 1 x 10^{-98}

Fehler - Summe

Nachfolgend die maximale Fehler-Summe des Rechners. Diese sind absolut angegeben, solange keine relativen Fehler auftreten.

Absoluter Fehler $= |f(X) - \hat{f}(X)|$

Relativer Fehler $= \frac{|f(X) - \hat{f}(X)|}{f(X)}$

$f(X)$ = genauer Wert

$\hat{f}(X)$ = errechneter Wert

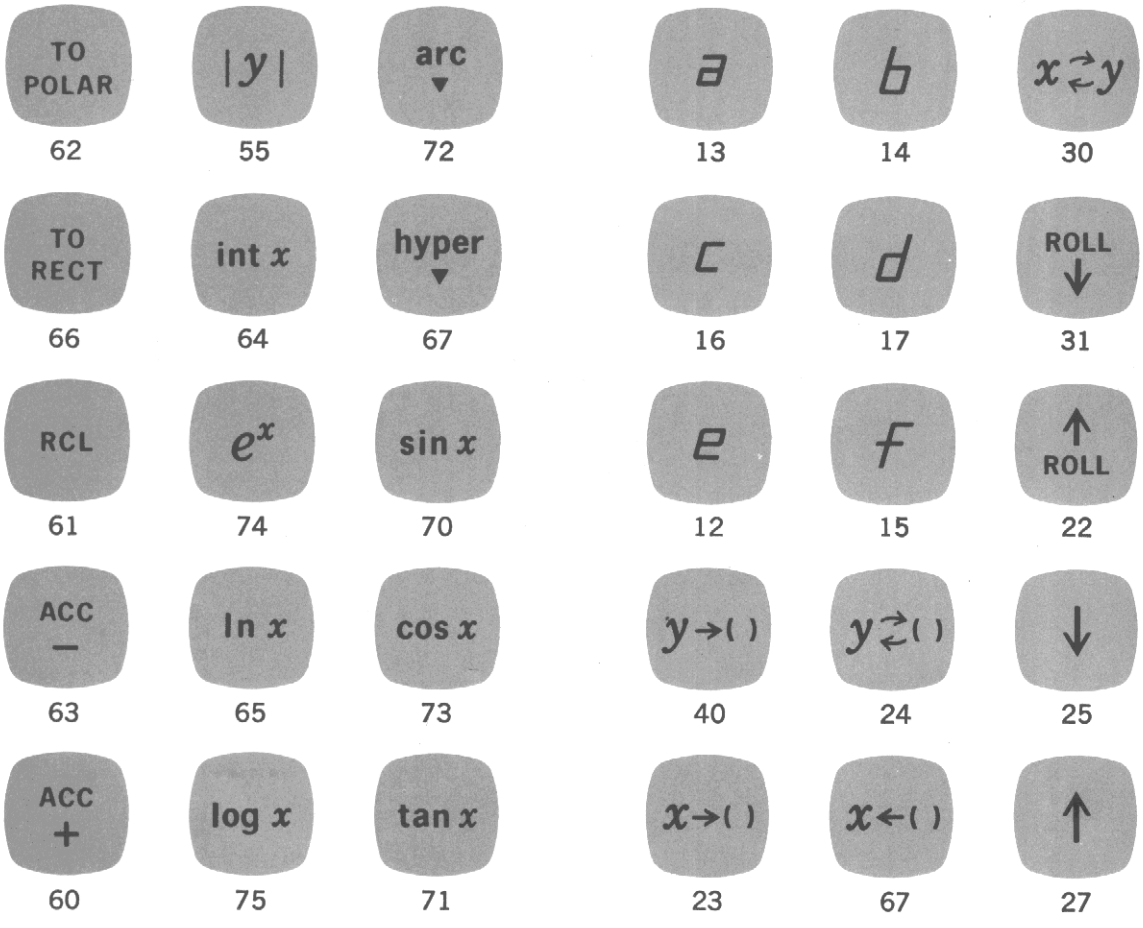
Anhang

Anmerkung:

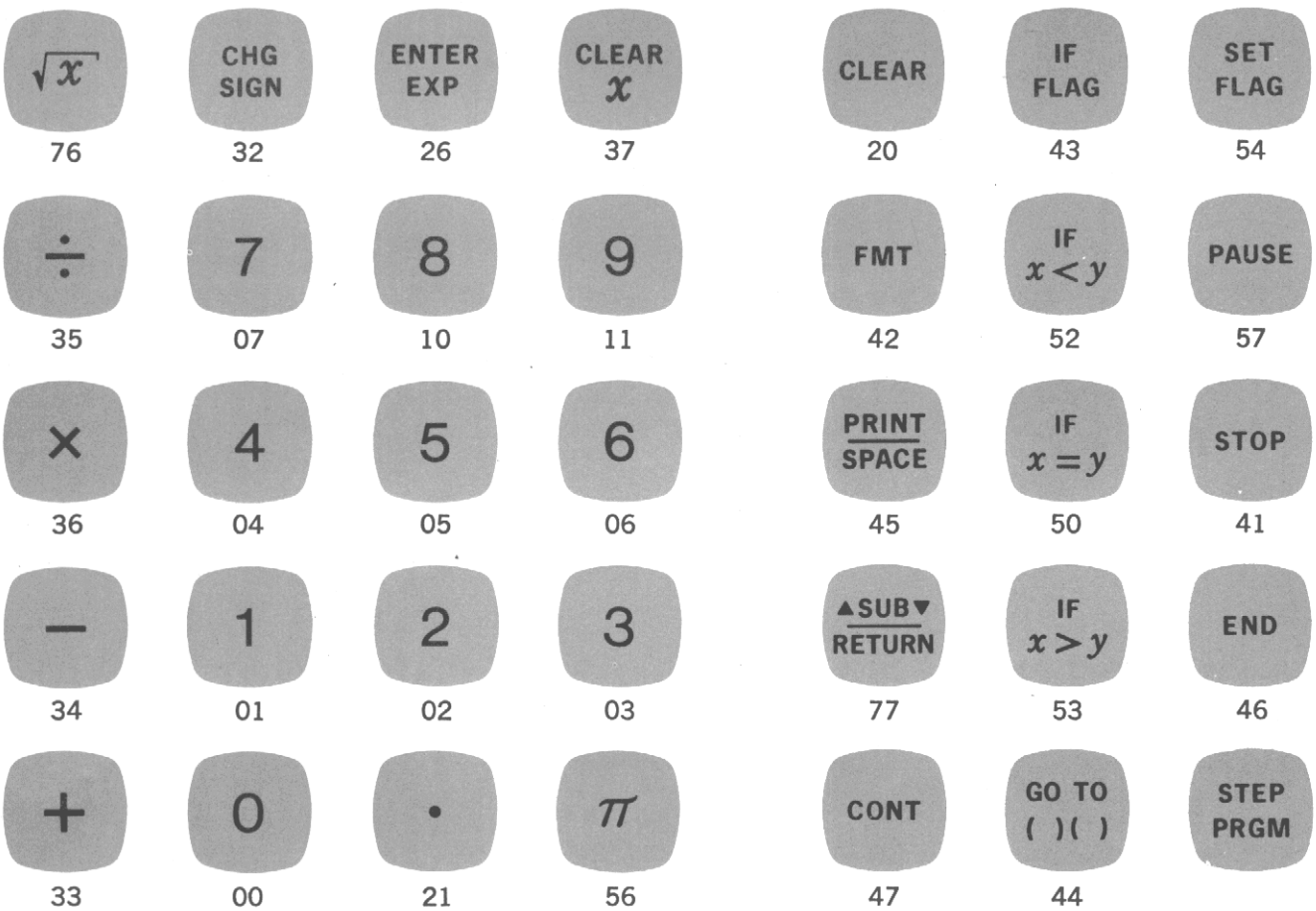
a) Eine „sichtbare Ziffer" ist eine Ziffer in der zehnten Stelle.

b) „Zahl der dekadischen Durchläufe" bezieht sich auf die Anzahl ganzzahliger Durchläufe dividiert durch 10.

I.	ADD(+, −)		$\pm 5 \times 10^{-11} \times 10$ (höherer add. Exponent)
II.	MPY		$\pm 1/2$ sichtb. Ziffer
III.	DIV		$\pm 1/2$ sichtb. Ziffer
IV.	SQRT		$\pm 1/12$ sichtb. Ziffer
V.	COS θ	$\theta < 1$ rad	$\pm 4 \times 10^{-11}$
		$1 \text{ rad} \leq \theta < \pi$	$\pm 9 \times 10^{-11}$
		$\pi \leq \theta < 2\pi$	$\pm 28 \times 10^{-11}$
		$2\pi \leq \theta$	$\pm 18 \times 10^{-10} \times 10$ (Zahl der dekadischen Durchläufe)
VI.	SIN θ	$\theta < 1$ rad	± 1 sichtb. Ziffer
		$1 \text{ rad} \leq \theta < \pi$	$\pm 12 \times 10^{-11}$
		$\pi \leq \theta < 2\pi$	$\pm 31 \times 10^{-11}$
		$2\pi \leq \theta$	$\pm 18 \times 10^{-10} \times 10$ (Zahl der dekadischen Durchläufe)
VII.	TAN θ	$\theta < 1$ rad	$\pm 1\ 1/2$ sichtb. Ziffer
		$1 \text{ rad} \leq \theta < \pi$	$\pm 1/2$ (Exp. des Ergebnis)
			$\pm 2 \times 10^{-10}$ sichtb. Ziffer
		$\pi \leq \theta < 2\pi$	$\pm 1 \times$ (Exp. des Ergebnis)
			$\pm 3 \times 10^{-10}$ sichtb. Ziffer
		$2\pi \leq \theta$	$\pm 5 \times$ Exp. des Ergebnis)
			$\times 10$ (Zahl der dekadischen Durchläufe)
			$\pm 31 \times 10^{-10} \times 10$
			of circles) sichtb. Ziffer
VIII.	POLAR IN RECHTWINKLIGE KOORDINATEN		gleich cos θ und sin θ
IX.	ARCTAN (a)	Bogenmaß	$\pm 1/10$ sichtb. Ziffer
		Grad	$\pm 1/4$ sichtb. Ziffer
X.	ARCSIN (a)	$a < .707$	$\pm 1/2$ sichtb. Ziffer
XI.	ARCCOS (a)	$a < .707$	$\pm 1/2$ sichtb. Ziffer $\pm 10^{-11}$
XII.	ARCSIN (a)	$a > .707$	Bogenmaß $\pm 10^{-10}$
			Grad $\pm 70 \times 10^{-10}$
XIII.	ARCCOS (a)	$a > .707$	Bogenmaß $\pm 10^{-10}$
			Grad $\pm 70 \times 10^{-10}$


TO POLAR
62
|y|
55
arc ▼
72
a
13
b
14
x⇄y
30
TO RECT
66
int x
64
hyper ▼
67
c
16
d
17
ROLL ↓
31
RCL
61
e^x
74
sin x
70
e
12
f
15
↑ ROLL
22
ACC −
63
ln x
65
cos x
73
y→( )
40
y⇄( )
24
↓
25
ACC +
60
log x
75
tan x
71
x→( )
23
x←( )
67
↑
27

Tasten-Codeziffern



Die Zahlen unter den Tasten sind der (oktale) Befehls-Code. Sie sind ebenfalls auf der herausziehbaren Karte vorne im Rechner angegeben.

Auf Seite 58 sind diese Code-Ziffern erläutert.

Hewlett-Packard Verkaufsbüros

Bundesrepublik und West-Berlin

❶ 6 Nieder Eschbach/Ffm 56, Berliner Straße 117, Tel.: (06 11) 50 10 64
❷ 1 West-Berlin 30, Lietzenburgerstr. 30,
❸ 2 Hamburg 1, Beim Strohhause 28, Tel.: 24 05 51/52
❹ 4 Düsseldorf 1, Achenbachstr. 15, Tel.: 68 52 58/59
❺ 703 Böblingen, Herrenbergerstr. 110, Tel.: 0 70 31-66 72 86
❻ 8 München 90, Reginfriedstr. 13, Tel.: 69 59 71

Schweiz

❼ Hewlett-Packard AG, 8952 Schlieren/Zürich, Zürcherstr. 20, Tel.: 0 51-98 18 21

Österreich

❽ Unilabor GmbH, Wien IX/71, Rummelhardtgasse 6/3, Tel.: 42 61 81
❾ **Bulgarien, Tschechoslowakei, DDR, Polen, Rumänien, Rußland und Ungarn sowie alle anderen europäischen Länder:**
❿ Hewlett-Packard SA, Rue du Bois-du-Lan 7, 1217 Meyrin, Genf, SCHWEIZ

PART NO. 09100-90021
SEP 1969

PRINTED IN U.S.A.