HEWLETT-PACKARD

HP-65

Quick Reference Guide

This booklet is primarily intended for reference use after you read the *HP-65 Owner's Handbook*. The **Procedures** section (¶1–¶10) is a brief digest for the user who needs a quick review of the more important operating procedures. The **Key Dictionary** (¶11–¶67) provides easy access to the details of the individual key operations and switches. All symbols on the HP-65 keyboard are presented in alphabetic order in the dictionary; non alphabetic symbols (e.g., + , [π]) are at the front (¶11–¶14). A general **Index** is provided at the end.

Procedures (¶1–¶10)

¶1. W/PRGM-RUN Switch

Set to: **RUN** position to calculate, to run a program, to read a pre-programmed card.

Set to: **W/PRGM** position to clear program memory, to key in a program, to edit a program, to write a program on a magnetic card.

¶2. Doing Arithmetic in the Stack

Compute	By Pressing
8 − 2 = 6	8 ENTER↑ 2 −
8 ÷ 2 = 4	8 ENTER↑ 2 ÷

Compute: $\frac{(4\times5)}{(2+3)}-6=-2$ using the keys shown below.

Contents of Stack Registers

T											
Z						20	20				
Y		4	4		20	2	2	20		4	
X	4	4	5	20	2	2	3	5	4	6	-2
Key	4	↑	5	×	2	↑	3	+	÷	6	−

NOTE: ENTER↑ *is here abbreviated as* ↑

¶3. Data Entry

Entering Negative Numbers. ■ **Press** CHS (change sign) after keying in the positive value. **Example:** to key in −12, **press** 12 CHS .

Entering Big and Small Numbers (*Scientific Format*). **Method:** ① Key in **mantissa** ② Press CHS if negative number ③ Press EEX ④ Key in exponent ⑤ Press CHS if exponent is negative. ■ **Example:** To key in 5×10^{-8}, press 5 EEX 8 CHS .

Correcting Mistakes. Press CLX and reenter the number.

PRINTED IN U.S.A.

¶4. Display

Controlling Display. To display x, rounded to n fixed decimal places, **press** DSP [•] [n] (where n = 0, 1, . . . , 9). To display x in scientific notation, rounded to n places, **press** DSP [n]. ■ Display can also be set by a program. **Blinking Display.** ■ Blinking display occurs when an illegal operation is attempted (**Example:** 5 ÷ 0). Depressing any key stops the blinking without doing the key function. ■ Blinking occurs when a program card is misread (or blank). ■ See the Key Dictionary for the limitations on the following: [LN], [LOG], [√x], [SIN], [COS], [D.MS+], [→D.MS], [→OCT], [1/x], [y^x], [÷]. **Multiple Decimal Points.** The display also indicates **low battery power** (all decimal points light up).

¶5. Prefix Operations

Performing "Gold" Functions (Upshift). Rule: Prefix gold function keys by [f] to do the function or [f^-1] to do the inverse. ■ **Example:** Calculate log (100) = 2 by pressing 100 [f] [LOG]. ■ **Example:** Calculate antilog (2) = 100 (the inverse) by pressing [f^-1] [LOG]. **Performing "Blue" Functions (downshift).** ■ Prefix blue function keys by [g]. ■ **Example:** Calculate 5! = 120 by

pressing 5 g n! . **Correcting or Cancelling a Prefix.** ■ To **correct** a wrong prefix, merely press the correct prefix. ■ To **cancel** a prefix, press f PREFIX . ■ Other prefix keys are DSP, GTO , STO , RCL , LBL .

¶6. Angular Mode

Operations involving angles (namely, →D.MS , SIN , COS , TAN , R→P) assume the angles to be in the units (degrees, radians, or grads) of the prevailing **angular mode** as set by DEG (or power on), RAD , or GRD . ■ 360 degrees = 2π radians = 400 grads. ■ **Converting from One Angular Mode to Another:**

g { DEG / GRD / RAD } f →D.MS g { DEG / GRD / RAD } f^{-1} →D.MS

¶7. Storage Register Operations

Storing a Number in Addressable Register R_n. ① Key in number to be stored. ② Press STO n (where n is a digit 1, . . . , 9). **Recalling a Number from R_n.** Press RCL n (where n is a digit 1, . . . , 9).

Doing Storage Register Arithmetic.

Subtraction. ($r_n - x \rightarrow R_n$): STO − n.

Addition. ($r_n + x \rightarrow R_n$): STO + n.

Multiplication. ($r_n \times x \rightarrow R_n$): STO × n.

Division. ($r_n \div x \rightarrow R_n$): STO ÷ n.

where n is a digit 1, . . . , 9. x is unchanged in these operations.

¶8. Clearing Registers

■ Press CLX to clear **X register.** ■ Press f STK to clear **entire stack** (X, Y, Z, and T) ■ Press f REG to clear all **storage registers** (R_1,..., R_9) ■ Power on clears all registers.

¶9. Programming

Program Memory. ■ Used to contain user's stored program. Capacity: 100 locations ■ **Top** is above 1st location. ■ **Bottom** is last location ■ **The Pointer,** an internal part of the calculator, determines which memory location is executed or displayed. In W/PRGM Mode, keystrokes are stored in memory as **codes:** The address for top of memory is 00 00. The codes for keys 0–9, are 00 – 09. For other keys, the code denotes row and column. **Example:** Code for R/S (row 8, column 4) is 84.

User Defined Functions. The top row keys are used to call a user defined function. When the calculator is turned on, default functions are de-

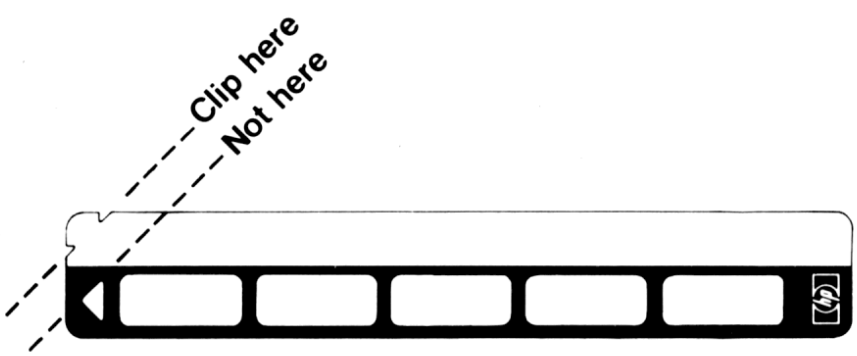
fined as shown in the window above the top row. Alternate functions may be keyed in or read from a magnetic program card. Pressing a top row key finds the function and executes it. **To Write a Function:** identify the beginning by LBL followed by the top row key that is to call it. End the function with RTN. **Example:** LBL E (ENTER↑ ENTER↑ × ×) RTN. The keys in parentheses calculate x^3. **Keying in a function:** ① Set W/PRGM-RUN switch to W/PRGM. ② Press f PRGM to clear program memory. ③ Press the keys in the order shown (LBL E . . . RTN in the sample case). If you make a mistake, press g DEL (to delete the error) and press the correct key. **To execute the function** (from keyboard): ① Set W/PRGM-RUN switch to RUN. ② **Press** the appropriate top row key. For the sample case, to compute $2^3 = 8$, press 2 E; to compute $3^3 = 27$, press 3 E; etc. ■ A user defined function can also be executed from a program by merely including the corresponding top row key in the program. **Revising (Editing) a Program.** ■ To move the pointer to the top of memory, press RTN in RUN mode. ■ To move the pointer to a label, press (in RUN mode) GTO n where n is the same digit or top row key as in LBL n (the label). ■ To step

through your program, use SST in W/PRGM mode (**See** ¶56). You will see the successive program codes in the display. ■ To insert a step below the currently displayed step, just key in the new operation in W/PRGM mode. (**See** ¶32). To delete a step or correct a mistake, press g DEL in W/PRGM mode (**See** ¶21). **Both overflow and underflow of a register will stop a program.**

¶10. Using Magnetic Cards

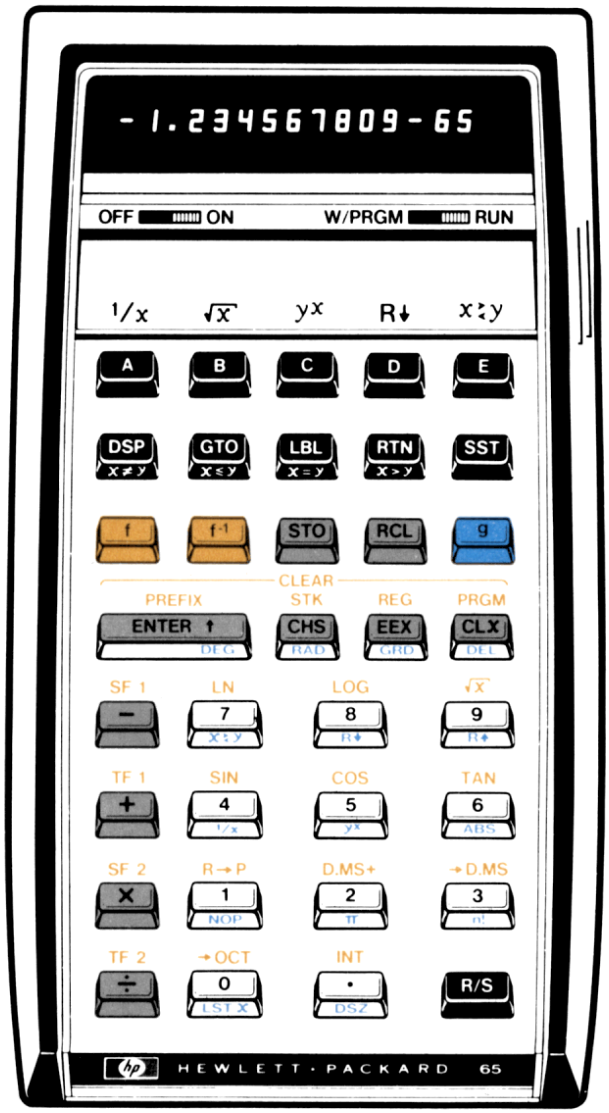
Reading a Pre-recorded Card. ■ Set W/PRGM-RUN switch to RUN. ■ Insert the card in the right lower slot. ■ If the card does not read properly, display will flash and program memory will be cleared (but, with no effect on registers). Press R/S and reinsert the card.

Recording (Writing) on a Program Card. ■ Set W/PRGM-RUN switch to W/PRGM. ■ Insert unprotected (unclipped) card in right lower slot. ■ Protect the card by clipping the notched corner.



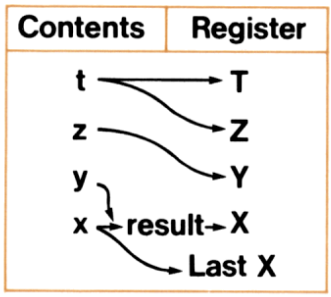
Key Dictionary (¶11–¶67)

This section contains reference entries for all keys with their associated symbols. In addition, the following entries are included. **Insert,** ¶32. **Last X,** ¶34. **Merged Codes,** ¶39. **OFF-ON Switch,** ¶43. **Stack Lift,** ¶57. **W/PRGM-RUN Switch,** ¶62.



¶11

−, **+**, **×**, **÷** **Arithmetic Operations.** Calculate respectively: (y − x), (y + x), (y × x), (y ÷ x). ■ Enable lift ■ Save x in Last X. ■ Drop stack as follows:



For **storage register arithmetic,** see ¶59.

¶12

0 . . . **9** **Digits.** Digits are used: ① to enter data (¶3), ② to specify registers (¶59, ¶49), ③ to label program steps (¶35, ¶31), ④ to specify displayed decimal places (¶24). ■ When used to enter numbers, digits enable the stack lift (¶57).

¶13

. **Decimal Point.** The **.** key is also used to specify fixed decimal display (¶24). ■ Enables stack lift.

¶14

π **Inserts** π (3.141592654) into the stack X register. ■ Lifts stack (¶57) if enabled. ■ Enables stack lift.

¶15

A, **B**, **C**, **D**, **E** **Top Row Keys.** When used

without a prefix, a top row key finds and executes a user defined function. ■ Used as suffix for LBL (¶35) and GTO (¶31). ■ **Precaution:** Power on automatically inserts 5 functions A - E in memory. Press f PRGM in **W/PRGM** mode to clear them. ■ Top row keys leave stack lift unaffected.

¶16

ABS **Absolute Value.** If x is negative, ABS makes it positive. ■ Saves x in Last X. ■ Enables stack lift.

B
C **See** ¶15.

¶17

CHS **Change Sign.** Changes the sign of the number in the stack X register unless the EEX key has been pushed; in this case the **exponent** sign is reversed, instead. ■ First non zero digit of a number entry gives x a positive sign. ■ Once a number is terminated (by a key other than a **digit,** CHS or EEX), the CHS key cannot reverse the exponent sign. In such a case CHS reverses the sign of the number. ■ No effect on stack lift.

¶18

CLX **Clear X Register.** Disables stack lift.

¶19

[COS] **Cosine** ([f] prefix) / **Arc Cosine** ([f⁻¹] prefix) (principal value, $0^o \leq$ result $\leq 180^o$ or equivalent in radians or grads). ■ Saves x in Last X. ■ Destroys register R_9 ■ **x** less than -1 or greater than $+1$ gives error (blinking zero) for arc cosine. ■ Enables stack lift.

D. **See** ¶15.

¶20

[DEG] **Set Degree Angular Mode.** ■ Affects angle operations ([→D.MS] , [SIN] , [COS] , [TAN] , [R→P]). ■ No effect on stack lift.

¶21

[DEL] **Delete Program Step.** ■ With W/PRGM-RUN switch in **W/PRGM** position, [g] [DEL] ① deletes the program step denoted by the program pointer (¶9), ② moves all the following steps up one step, and ③ inserts a [g] [NOP] code in the vacated bottom position of memory. ■ Inoperative during **run mode**, ([g] [DEL] acts as [CLX]) ■ [g] [DEL] can be used to back up the pointer (after which you reinsert the deleted codes). ■ **If memory is full** (i.e., a minus sign shows in display), [g] [DEL] loses bottom memory step. ■ **If the program pointer is at the bottom** (i.e., 2 minus signs show in display), [g] [DEL] deletes 2 locations. ■ No effect on stack lift.

¶22

[→D.MS] **Converts to** ([f] prefix) or **from** ([f⁻¹] prefix) **Degrees, Minutes, Seconds.** ■ Converts the contents of the X register (in degrees, radians, or grads) to the form DDDDD.MMSS (degrees, minutes, seconds) or the inverse, depending on the prefix. ■ Saves x in Last X. ■ Gives error (blinking zero) if the magnitude of x (degrees or equivalent in radians or grads) exceeds 99999.99999. ■ Enables stack lift.

¶23

[D.MS+] **Adds** ([f] prefix) / **Subtracts** ([f⁻¹] prefix) **Degrees, Minutes, Seconds.** ■ Used for adding/subtracting (y—x) degrees-minutes-seconds or hours-minutes-seconds in the stack X and Y registers. Operands are of the form **DDDDD.MMSS** in the stack X and Y registers. ■ Saves x in Last X. ■ Drops stack (as in ¶**11**). ■ Gives error (blinking 0.00), if the magnitude of an operand or the result exceeds 99999.59599 (degrees, minutes, seconds). ■ Enables stack lift.

¶24

DSP **Display** (prefix). ■ **DSP** [.] [n] displays x rounded to n **fixed decimal** places. ■ **DSP** [n] sets **scientific display** of x rounded to n decimal places where n = 0, 1, . . . , 9.

■ Does not alter internal value of x. ■ Power on sets DSP [•] [2] (0.00). ■ If x is too small for a specified display, (signed) zero is displayed. ■ If x is too large for the specified format, DSP [9] format is used. ■ No effect on stack lift.

¶25

[DSZ] **Decrement and Skip on Zero.** Subtracts 1 from an integer in register R_8, then skips two program memory locations if R_8 contains zero. ■ The decrement operation is suppressed outside the limits: $1 \leq |r_8| < 10^{10}$. ■ Useful for looping. **Rule:** To execute a labelled program segment n times, preset n in R_8 and use [DSZ] to determine whether or not to repeat the segment. **Example:** The following loop executes 12 times, then stops: 12 STO [8] LBL [1] . . . g [DSZ] GTO [1] R/S. ■ No effect on stack lift.

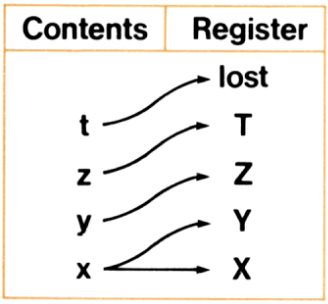
E See ¶15.

¶26

EEX **Enter Exponent.** Terminates the **mantissa** portion of a number and institutes the entry of a power of 10 multiplier (**exponent**) into the X register. If no **mantissa** was previously entered, EEX sets up a **mantissa** of 1. ■ Enables stack lift.

¶27

ENTER↑ **Copy x to Y.**



- Copies the contents of the X register into the Y register, pushing y into Z and z into T (t is lost).
- Disables stack lift.

¶28

f **f^{-1}** **Upshift** (prefix). The gold symbol above a key denotes the function of the key if preceded by **f**. ■ The inverse or complement is done if the key is prefixed by **f^{-1}**. ■ No inverses are defined for clear functions (4th row keys). For these keys **f^{-1}** gives the clear function. ■ To cancel an unwanted **f** or **f^{-1}**, press **PREFIX**. ■ No effect on stack lift.

¶29

g **Downshift** (prefix). When **g** is followed by a key having a blue symbol below it, the function denoted by the symbol is done. ■ To cancel an unwanted **g**, press **f** **PREFIX**. ■ No effect on stack lift.

¶30

GRD **Sets Grad *Angular Mode.*** Affects angle operations (→D.MS, SIN, COS, TAN, R→P). ■ No effect on stack lift.

¶31

GTO **Go to** (prefix). When followed by a digit (0, . . . , 9) or a letter (A, . . . , E), GTO advances the program pointer downward to the first occurrence of the corresponding program label: LBL followed by the same digit or letter. ■ No effect on stack lift.

¶32

Insert ■ Pressing a key in W/PRGM mode stores the instruction code in program memory between the displayed code and the following instruction code and moves the pointer to display the code just inserted. The bottom location drops off. ■ **Insert is not performed:** (a) For PRGM, DEL, SST. (b) For the second key of a merged code (¶39). (c) When the pointer is at the bottom. ■ No effect on stack lift.

¶33

INT **Truncates x to Integer** (f prefix) **or fraction** (f^{-1} prefix). ■ Saves x in Last X. ■ Retains the sign of the number. ■ Enables stack lift.

¶34

Last-X Register. ■ Contains the value of x before the latest f(x) or f(x, y) was computed. Used to: ■ compute functions that make multiple use of the same operand. ■ enable recovery from keystroke errors in certain instances. ■ To **Recall Last x, See** [LST x] (¶38). ■ The following save x in Last X before performing their functions: [+] [−] [×] [÷] [ABS] [COS] [→D.MS] [D.MS+] [INT] [LN] [LOG] [→OCT] [R→P] [SIN] [TAN] [n!] [√x] [1/x] [y^x] ■ Note that [CLx] does not affect the Last-X register.

¶35

[LBL] **Label** (prefix). [LBL] identifies its suffix (a digit [0], . . . , [9], or top row key, [A], . . . , [E]) as a label in a stored program. A branch to the part of the program thus labelled can then be done by executing [GTO] followed by the same suffix. For user defined functions, the label suffix must be a top row key ([A], . . . , [E]). ■ No effect on stack lift.

¶36

[LN] **Natural Log (x)** ([f] prefix) **or e^x** ([f^{-1}] prefix). ■ Saves x in Last X. ■ [f] [LN] gives error (blinking zero) if x is zero or negative. ■ Enables stack lift.

¶37

[LOG] **Common Log(x)** ([f] prefix) **or 10^x** ([f^{-1}] prefix). ■ Saves x in Last X. ■ [f] [LOG] gives error (blinking zero) if x is zero or negative. ■ Enables stack lift.

¶38

[LST x] **Recall Last x to the X Register.** ■ Performs automatic stack lift unless the lift is disabled. See ¶34. ■ Enables stack lift.

¶39

Merged Codes. Program codes for the following are merged with their respective prefix codes: [LST x], [NOP], [x≷y], [R↓], [R↑], [x≠y], [x≤y], [x=y], [x>y], [1] . . . [8] when prefixed by [STO] or [RCL]. ■ **Example:** [g] [LST x] in program mode is merged and displayed as 35 00; [STO] [5] as 33 05, etc. For explanation of **codes,** see ¶9.

¶40

[n!] **Integer Factorial.** ■ Computes the **factorial** of a nonnegative integer n in the X register.

$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$

$0! = 1$

■ Saves x in Last X; ■ Negative or noninteger x gives error (blinking zero) ■ If x exceeds 69, **overflow** occurs. ■ Enables stack lift.

¶41

NOP **No Operation.** ■ Useful as a filler in tests. ■ f PRGM in W/PRGM mode clears the entire memory to g NOP (merged code 35 01). ■ No effect on stack lift.

¶42

→OCT **Convert Integer x to/from Octal.** ■ Saves x in Last X. ■ Non-integer or x larger in magnitude than 1073741823_{10} gives error (blinking zero). ■ Enables stack lift.

¶43

OFF-ON Switch. "Power on": clears all registers and flags. ■ sets the display rounding to 2 fixed places (0.00). ■ leaves program pointer at top of memory. ■ inserts 5 functions at the top of memory that are callable from the top row keys to allow single stroke execution of the 5 functions shown in the window above the top row.

¶44

PREFIX **Clear Prefix.** When prefixed by f or f^{-1}, PREFIX cancels the effect of an impending prefix, so that a nonprefix operation can be done. ■ If a wrong prefix key is depressed before keying in the suffix of a prefixed operation, the error can be cor-

rected by simply pressing the correct prefix and proceeding from there. ■ No effect on stack lift.

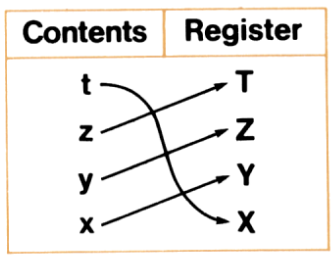
¶45

PRGM **Clear Program Memory.** In program mode, f PRGM (or f^{-1} PRGM) clears the entire program memory to "no operation" codes (35 01), leaving the **pointer** at the top of memory. ■ In run mode, f PRGM is equivalent to CLX. ■ No effect on stack lift.

¶46

R↑ **Roll Stack Up.**

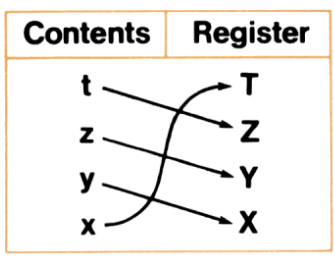
Enables stack lift.



¶47

R↓ **Roll Stack Down.**

Enables stack lift.



¶48

RAD **Set Radian Angular Mode.** ■ Affects angle operations (→D.MS, SIN, COS, TAN, R→P) ■ No effect on stack lift.

¶49

RCL **Recall** (prefix). Recalls storage register R_n

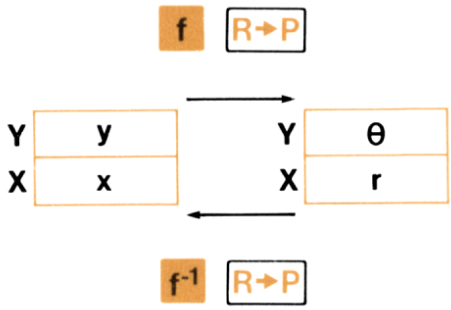
(n is a digit 1, . . . , 9) to the X register. ■ Lifts stack unless stack lift is disabled. ■ Enables stack lift.

¶50

REG **Clear Registers.** When prefixed by f or f^{-1}, REG clears storage registers R_1 . . . R_9 to zero.

¶51

R→P **Rectangular to Polar.** ■ Transforms the respective contents of the X and Y stack registers from rectangular form (x, y) to polar form (r, θ) (f prefix) or the inverse (f^{-1} prefix). ■ Saves x in Last X. ■ Destroys previous contents of register R_9. ■ Enables stack lift.



¶52

R/S **Run/Stop.** ■ If R/S is pressed and a stored program is not executing, the stored program starts executing at the step denoted by the program pointer. ■ If exe-

cuted in a stored program, R/S stops the program, allowing keystrokes from the operator. The program pointer is positioned at the R/S . ■ If a R/S in a program is immediately preceded by a numerical entry from the program, the automatic lift is disabled upon return to the keyboard. This allows a program to display prompting information that will not be lifted in the stack if you enter a number from the keyboard. Except for this case, R/S does not affect the stack lift.

¶53

RTN **Return.** ■ If executed from the keyboard, RTN merely resets the program pointer to the top of memory. ■ In a stored program, RTN is the logical end of a user defined function. ■ If a function is executed from the keyboard, RTN **stops** the program. ■ If a function is executed in a program, execution of RTN **resumes** the calling program. ■ A function executed from the keyboard or a nonfunction program can execute another function. The latter, however, cannot properly execute yet another function. ■ No effect on stack lift.

¶54

SF 1, SF 2 **Set flag 1, Set flag 2.** ■ f SF 1 sets flag 1 **on** while f^{-1} SF 1 sets it **off.** ■ f SF 2 performs similarly, but using flag

2. To test the flags, **see** ¶61. ■ No effect on stack lift.

¶55

SIN **Sine** (f prefix) / **arc sine** (f^{-1} prefix). ■ Arc sine calculates the principal value ($-90^o \leq$ result $\leq +90^o$ or equivalent in radians or grads). ■ Saves x in Last X. ■ Destroys register R_9. ■ x less than -1 or greater than $+1$ gives error (blinking zero) for arc sine. ■ Enables stack lift.

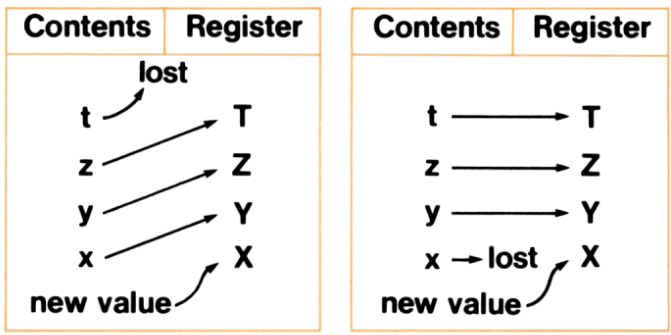
¶56

SST **Single Step.** ■ In W/PRGM mode, **SST** advances the program pointer to the next memory location, displaying the step code. Repeated use of the key enables you to review a program and to position the pointer for editing. ■ In RUN mode, **SST** executes the program step denoted by the program pointer. In the case of single stepping a call to a user defined function, the entire function executes (as one step) before returning control to the keyboard. ■ No direct effect on stack lift.

¶57

Stack Lift. The stack lift is an automatic response of the HP-65 to allow you to put a new value in the X register and to simultaneously lift the previous values x, y, and z into the respectively higher registers Y, Z and T (t is lost) for future use—

this obviates the use of ENTER↑ in many instances. If the lift is enabled (i.e., not disabled), the lift takes place (as shown in the **left** diagram) at the beginning of [π], [LST x], RCL [n], and on the first keystroke of a new number. If the lift is disabled, the previous x is lost as shown in the **right** diagram.



Lift Disable Keys: R/S in a program if the program has just put a new number in the X register from program memory; CLx, ENTER↑ at any time.

Lift Enable Keys: All number entry keys [0],..., [9], [•], EEX, [π], **but not** CHS. ■ **All calculating keys** −, +, ×, ÷, [ABS], [COS], [→D.MS], [D.MS+], [INT], [LN], [LOG], [LST x], [n!], [→OCT], [R→P], [SIN], [TAN], [1/x], [√x], [y^x]. ■ **Stack manipulating keys** [R↑], [R↓], [x⇄y], but **not** ENTER↑ ■ **Storage registers keys:** STO, RCL.

No Effect Keys: All other keys have no effect on the lift status. They include: all programming keys, angular mode keys, display control keys, clear keys (except CLx), and CHS.

¶58

STK **Clear Stack.** (f or f^{-1} prefix).

¶59

STO **Store** (prefix). STO n copies the contents of register X into storage register R_n (n = a digit 1, . . . , 9). To perform an arithmetic function (+, −, ×, ÷) of x and a storage register R_n, insert the corresponding arithmetic key between STO and n.

STO + n	adds	$(r_n + x) \rightarrow R_n$
STO − n	subtracts	$(r_n - x) \rightarrow R_n$
STO × n	multiplies	$(r_n \times x) \rightarrow R_n$
STO ÷ n	divides	$(r_n \div x) \rightarrow R_n$

■ Stack registers and Last X are unchanged. ■ Storage arithmetic codes are unmerged. ■ Enables stack lift.

¶60

TAN **Tangent** (f prefix / **Arc Tangent** (f^{-1} prefix). ■ **Arc tangent x** calculates the principal value ($-90^o \leq$ result $\leq +90^o$ or equivalent in grads or radians). ■ Saves x in Last X. ■ Destroys register R_9. ■ Enables stack lift.

¶61

[TF 1] , [TF 2] . **Test Flag 1, Test Flag 2.** ■ [f] [TF 1] tests flag 1, skipping 2 memory locations if flag 1 is **off**, while [f⁻¹] [TF 1] skips if flag 1 is **on**. ■ [f] [TF 2] performs similarly, but using flag 2. ■ To set the flags, **see** ¶54. ■ No effect on stack lift.

¶62

W/PRGM-RUN Switch. ■ **W/PRGM** position sets **program mode**, used to: ■ create and edit a stored program or ■ write program memory on a magnetic card. ■ **RUN** position sets **run mode**, used to. ■ read a magnetic card into program memory ■ do calculations ■ execute stored programs.

¶63

[1/x] **Reciprocal of x.** ■ Saves x in Last X. ■ Gives error (blinking zero) if reciprocal of 0 is attempted. ■ Enables stack lift.

¶64

[√x] **Square Root** ([f] prefix) / **Square** ([f⁻¹] prefix). ■ Saves x in Last X. ■ For [f] [√x], negative x gives error (blinking zero). ■ Enables stack lift.

¶65

[x⇄y] **Exchange x and y.**

Contents	Register
y → Y x → X	

Enables stack lift.

¶66

[x≠y], [x≤y], [x=y], [x>y] **Relational Tests of x and y.** Each test compares the value in the X and Y registers and skips 2 memory locations if the test condition is not met. ■ Destroys register R_9. ■ No effect on lift.

¶67

[y^x] **Exponential.** Raises the contents of the stack Y register to the power specified in the X register. ■ Saves x in Last X. ■ Drops stack (as in ¶11). ■ Negative or zero value of y gives error (blinking zero). ■ Enables stack lift.

Index

NOTE: *To reference keyboard symbols, please see the preceding* ***Key Dictionary.***

D

E

F

G

I

L

M

N

O

P

R

S

T

U

W

HEWLETT hp PACKARD

Sales, service and support in 172 centers in 65 countries
19310-19320 Pruneridge Ave., Cupertino, CA 95014

For Additional Sales and Service Information Contact Your Local Hewlett-Packard Sales Office or Call 408/996-0100 (Ask for Calculator Customer Service).

00065-90203 Printed in U.S.A.