HP-11C Quick Reference

Thimet

Memory & Display

Memory	Approx. 204 bytes of memory Default: 20 number storage registers (7 bytes each) and 64 program steps. Storage registers are automatically converted to program memory as needed. 4-level stack, Last-X, Index register. Nonvolatile memory, partially merged program commands
Number separator	Turn off, press & hold ON, press ".", release ON, release "." This toggles between using a dot or comma for the decimal separator.
Global reset	Turn off, press & hold ON, press "-", release ON, release "-" This clears all permanent memory!
MEM	Displays memory assignment in the form "`P-56 r-,9`" In this example there are 56 unused program steps and the next storage register converted to program memory is location ",9"
FIX 0-9	Select fix-point format
SCI 0-9	Select scientific format with exponent
ENG 0-9	Select engineering format with exponent always being a multiple of 3

Clearing Data

←	Deletes either the last digit during number entry or the entire X-register in case number entry has been terminated. Also used in programming mode, see there
CLEAR ∑	Clear stack and summation registers 0-5
CLEAR PRGM	RUN mode: Set program counter to 000 PRGM mode: Erase entire program memory
CLEAR REG	Clear all storage registers
CLEAR PREFIX	Clear prefix key and briefly display all 10 digits of the mantissa
CL X	RUN mode: Clear X-register PRGM mode: Store the CLX command as a program command

Storage Registers & Indirect Addressing

STO 0-9, .0-.9	Store X in the specified storage register. By default, 20 registers are available
STO + 0-9 STO – 0-9 STO x 0-9 STO ÷ 0-9	Register arithmetic. Only supported for registers 0-9. RCL register arithmetic is not supported. To perform register arithmetic with registers .0-.9 use indirect addressing (see below).
RCL 0-9, .0-.9	Recall number from storage register to X-register
STO I	Store X in index register
RCL I –*or*– f I	Recall value from index register
X↔I	Exchange X with index register

STO (i)	Store X in the register pointed to by I. Values of I and corresponding registers: 0-9 →R0-R9, 10-19 →R.0-R.9, 10 →I
STO +–x÷ (i)	Perform indirect register storage arithmetic
RCL (i) –*or*– f (i)	Recall value from the register pointed to by I
X←(i)	Exchange X with the register pointed to by I
RCL ∑+	Recall ∑x and ∑y from the summation registers into X & Y
LST X	Recall last value of X-register as is was before the previous operation

Functions (Selection)

RAN#	Create random number $0 \leq X < 1$
STO f RAN#	Store X as the new random number seed
→P	Convert (X=x,Y=x) from orthogonal to polar coordinates (X=r,Y=θ) See label on the back of the calculator
→R	Convert (X=r,Y=θ) from polar to orthogonal coordinates (X=x,Y=x)
→H.MS	Convert fractional hours to hours, minutes & seconds H.MMSSs
→H	Convert hours, minutes & seconds H.MMSSs to fractional hours
→RAD	Convert degress (360) to radians (2π)
→DEG	Convert radians (2π) to degress (360)
Py,x	Permutations = Y! / (Y-X)! Number of possibilities to select X elements from a group of Y different elements where different sequences count separately.
Cy,x	Combinations = Y! / [X! • (Y-X)!] Number of possibilities to select X elements from a group of Y different elements where different sequences *do not* count separately.
x!	Faculty and Gamma. Calculates Γ(x+1)=n! for positive and non-integer negative numbers
RND	Rounds X to the number of currently displayed digits
FRAC	Returns the fractional part of X
INT	Returns the integer part of X
y^x	Y to the power of X. Works also for negative Y in case X is integer
%	Calculates X percent of Y. Does not pop the stack!
Δ%	Percential difference from Y to X. Does not pop the stack!

Trigonometric Functions

DEG	Set trig mode "degrees" (360)		
RAD	Set trig mode "radians" (2π), indicated in display		
GRD	Set trig mode "grad" (400) , indicated in display		
SIN	COS	TAN	Trigonometric functions, performed in current mode (DEG, RAD, GRD)
SIN^{-1}	COS^{-1}	TAN^{-1}	Inverse trig functions
HYP SIN	HYP COS	HYP TAN	Hyperbolic functions (independent of trig mode!)
HYP^{-1} SIN	HYP^{-1} COS	HYP^{-1} TAN	Inverse hyperbolic functions

Summation & Statistics

General	The statistics registers occupy the storage registers 0-5, see calculator's back label. See section **Clearing Data** for statistics register deletion. Stats registers can also be used for vector addition and substraction! Register usage: 0=n, 1=∑x, 2=∑x^2, 3=∑y, 4=∑y^2, 5=∑xy
∑+ STO ∑+	Add X and Y to the stats registers. This will display the total number of entries and disable stack lift so that the next entry will overwrite the count.
∑-	Substract X and Y from the stats registers
RCL ∑+	Recall ∑x and ∑y from the summation registers into X & Y
$\bar{x}$	Calculate ∑x & ∑y mean value and place result in X & Y
s	Calculate ∑x & ∑y standard deviation and place result in X & Y. sx=SQRT[{n∑x^2 – (∑x)^2} / {n(n-1)}]
L.R.	Linear regression. Calculates a straight line thru the (X,Y) data points and returns the slope of the line in Y and the y-offset in X
$\bar{y}$,r	This function assumes a straight line thru the (X,Y) data points and calculates for a given X the approximated $\bar{y}$ value which is returned in X. In Y this function returns an estimate how close the data points come to a straight line. +1 indicates that all points lie on a line with positive slope, -1 indicates that all points lie on a line with negative slope, 0 indicates that an approximation by a straight line isn't possible.

Programming

P/R	Toggles between RUN (program execution) and PRGM (program entry) mode. See section **Clearing Data** for program memory and program step deletion.
SST	RUN: Display and execute next program step PRGM: Step forward thru program
BST	RUN: Display and go back to previous program step but do not execute any program code PRGM: Step backwards thru program
Inserting & deleting steps	• Program entry starts with line number 1 • Line "`000-`" indicates the start of the program space • Commands are inserted after the currently displayed line • Delete the currently displayed instruction with ← • Program code values indicate the row & column of a command with the exception that numbers are displayed as such. Prefix key have their own code. Example: `001-42.21. 0` corresponds to "LBL 1" (42=f, 21=SST/LBL, 0=0)
f A-E	RUN: Execute program starting at the given label. An error occurs if the label is not found. Any keypress will halt the program! PRGM: Insert a "GSB label" command
USER	Normally, f A-E must be pressed to execute a program, see above. In USER mode the prefix-f is not needed, ie. pressing e^x will immediately execute the program starting at label B. Use the prefix-f to reach the keys normal function. USER mode is indicated in the display

R/S	RUN: Continue program at current program counter PRGM: Insert R/S command which will halt the program at this location
RTN	RUN: Set program coutner to 000 PRGM: Insert a RTN instruction. This will return from a subroutine or at the top level end the program and set the program counter to 000
GTO . nnn	RUN & PRGM mode: Jump to program line nnn
GT0 0-9, A-E	RUN: Set program counter to the specified label PRGM: Insert a GTO instruction
GSB 0-9, A-E	RUN: Execute the program starting at the given label PRGM: Insert a GSB instruction. A maximum of *four* subroutine calls can be nested
Flags	There are *two* flags, 0 and 1. SF n: Set flag n, CF n: Clear flag n F? n: Execute next step if flag is set, skip next step if flag is clear
Comparisn	X=0, X≠0, X>0, X≤0, X=Y, X≠Y, X>Y, X≤Y If camparisn is `false`: Skip the next program step If camparisn is `true` : Execute the next program step
PSE	Halt program for about 1 second and display the X-register

Using The Index Register In Programs

GTO I	Jump to the label indicated by the I register. Only the integer part of I will be used! Values of I and associated labels: If I≥0: 0...9 →LBL 0...LBL 9, 10...14 →LBL A...LBL E If I<0: Jump to the line number indicated by the absolute value of I. Ie. if I=–5.3 the jump will go to line number 5.
GSB I	Perform subroutine call to the label indicated by the I register
ISG	Increment and skip if greater. This loop command uses the index register I which must contain a value in the form `nnnnn.xxxyy` where: ±nnnnn: Current (initial) loop counter value xxx: Comparisn value for loop counter yy: Loop counter increment (or decrement for DSE), if y=0 then 1 is used instead ISG first increments n by y and then compares the new n to x: If n>x the next program step is skipped If n≤x the next program step is executed Ie. if initially I=0.023 then the loop will run from 0 to 22 (or 1 to 23)
DSE	Decrement and skip if equal (or smaller). DSE first decrements n by y and then compares the new n to x: If n≤x the next program step is skipped If n>x the next program step is executed