HP-32S Quick Reference

Miscellaneous

Memory	Total of 390 bytes for number storage and programs. 8 bytes are needed per number storage. A maximum of 216 bytes can be used for variables.
▲ and ▼	When shown in the display indicates that the keys ▲ and ▼ can be used to scroll thru a list or program
←	Deletes digits to the left if the cursor is displayed. Otherwise clears X register or steps back one level in the menu hierarchy
C	Clear X or close all menus
SHOW	Display all digits of the mantissa as long as key is held down
Functions	When a function keys is pressed and held down (ie. "LN") the function name is displayed. When the key is released the function is executed. Holding down a key for a longer time does *not* cancel the function!
XEQ A..Z, (i)	Execute a program starting at the specified label. For indirect program execution using (i) see section *Programming*
Contrast	Press an hold "C" key, press "+" for darker or "-" for lighter display.

Number Storage

Memory	There is a total of 27 storage registers: A..Z & index register i
STO A..Z	Save X in storage register A..Z
STO i	Save X in index register i. Do not confuse the index register i with the regular storage register I !!
STO (i)	Save X in register indexed by i=1..26. Only the absolute value of the integer part of i is used. It is not possible to indirectly access the indirect register or the summation registers
STO +-x÷ [reg]	Storage register arithmetic: Register OP X → Register
RCL A..Z, i, (i)	Recall storage register value into X
RCL +-x÷ [reg]	Recall register arithmetic: X OP Register → X
VIEW A..Z, i, (i)	View one of the storage registers without changing the stack. Press C or ← to clear
Last X	Contains the value of X before the most recent operation

Summation

Memory	6 summation registers, independent of storage registers A..Z. 48 bytes from the global 390 bytes of memory are allocated as soon as Σ+ is pressed for the first time. CLEAR Σ deallocates the registers and frees the memory
CLEAR Σ	Clear all summation registers
Σ+	Add X & Y to summation registers, increment and display n

Σ-	Substract X & Y from summation registers, decrement and display n	
Σ	n	Recall number of entries made with Σ+
	x	Recall sum of X values
	y	Recall sum of Y values
	x^2	Recall sum of X^2 values
	y^2	Recall sum of Y^2 values
	xy	Recall sum of XY values
$\bar{x}$, $\bar{y}$	$\bar{x}$	Mean value of X values
	$\bar{y}$	Mean value of Y values
	$\bar{x}w$	Weighted mean of X values: w= Σxy / Σy
s	sx	Standard deviation of X values
	sy	Standard deviation of Y values
L.R.	x	Given a y-value in X calculate an estimate for x based on a straight line fitted into the data points entered by Σ+
	y	Given a x-value in X calculate an estimate for y
	r	Return correlation coefficient for the straight-line fit: -1 = perfect line with negative slope 0 = no correlation +1 = perfect line with negative slope
	m	Return inclination of the fitted line
	b	Return y-offset of the fitted line

Functions (Selection)

y^x	Y to the power of X. Y may be negative if X is integer
%	Y percent of X. Stack doesn't drop
%CHG	Percentual change from Y to X. Stack doesn't drop
HYP	Prefix for hyperbolic trigonometrix or inverse trigonometric functions
COMPLX	Prefix for complex operations, see section *Complex Numbers*

Complex Numbers

Memory	For complex operations the stack registers X, Y, Z & T contains complex values (X + iY) and (Z + iT).
CMPLX	To perform a complex operation it must be preceeded by the CMPLX key
Functions	The following complex functions are supported: + - x ÷ +/- y^x 1/x LN e^x SIN COS TAN
Storage	Two storage registers must be used to store a complex number

Menus

PARTS	**Parts of numbers**
IP	Integer part of X
FP	Fractional part of X
RN	Round X to the number of digits of the current display format (SCI, ENG, FIX)
ABS	Absolute value of X
MODES	**Trigoometric & display modes**
DG	Degress (360, default)
RD	Radiants (2π), indicated by "RAD" in the display
GR	Grad (400) , indicated by "GRAD" in the display
.	Use a doit for the decimal point
,	Use a comma for the decimal point
DISP	**Display formats**
FX n	Fix point format with n digits after the decimal point. n=0..11 and enter ".0" and ".1" for n=10 and n=11
SC n	Scientific (exponential) format with n digits after the decimal point
EN n	Engineering (exponent multiple of 3) format with n digits after the decimal point
ALL	Displays all non-zero digits after the deciml point
CLEAR	**Clearing data** See MEM command for clearing individual programs and variables
x	X register
VARS	Variables
ALL	Everything! Use must confirm in another menu
Σ	Summartion registers
P↔RECT	**Rectangular/polar coordinate conversion**
y,x→θ,r	Convert rectangular to polar coordinates
θ,r→y,x	Convert polar to rectangular coordinates
H↔HMS	**Fractional hours to hours/minutes/seconds conversion**
→HR	Convert fractional hours to h.mmss format
→HMS	Convert h.mmss to fractional hours
D↔RAD	**Degrees/radians conversion**
→DEG	Convert radians (2π) to degrees (360)
→RAD	Convert degrees (360) to radians (2π)
BASE	**Number base selection**
DEC	Decimal number base (normal operation)
HX	Hexadecimal number base. Tow row keys A..F correspond to hexadecimal numbers. Non-decimal number have a precision of 36 bits
OC	Octal
BN	Binary. The display shows arrow symbols → and ← in case the number doesn't fit in the display. Use the top-row keys A and F to scroll

SOLVE/∫	**Root finding and integration**
FN A..Z, (i)	Specify which user-defined function f (defined by a program starting at the specified label) shall be SOLVE'd or integrated by ∫ FN
SOLVE A..Z, i, (i)	Find a root: Adjust the specified variable (= unknown variable in storage register *reg*) so that the function value f(*reg*) is 0. The program defining the function must use the unknown variable (=storage register *reg*) to calculate the function result. The function result f(*reg*) must be returned in the X register. The function must be ended with a RTN instruction. The program can use INPUT instructions to read in variable values: The INPUT instruction for the unknown variable *reg* will be ignored; INPUT instructions for other variables will only be executed once. Before SOLVE is executed two initial guesses for the unknown variable *reg* can be placed in the variable's storage register and the X register. After SOLVE terminated the following values are available: • Unknown variable's storage register: Value of the variable where f(*reg*)=0, this is the "root" • X register: The root value • Y register: Previous estimate for the root, should be identical with X • Z register: f(*reg*) which should ideally be 0 If no root can be found an error occurs. R/S can be used to stop the solver. The variable *reg* contains the currently best estimate for the root. The calculation can be continued by presing R/S. SOLVE cannot call a function which calls FN, SOLVE or ∫ FN and uses 33.5 bytes of memory. SOLVE can be used in a program: If a root has been found the next instruction after SOLVE will be executed, otherwise skipped
∫ FN A..Z, i, (i)	Integrate function defined by FN over the specified variable *reg*. See SOLVE for the requirements that the function f(*reg*) must meet. Before ∫ FN can be executed Y must contain the lower and X the upper integration limit. After the integration ends: • X contains the integral • Y contains an error estimate. The precision of the result is determined by the display setting (ENG, SCI, FIX). By using reduced precision it is possible to speed up the integration process. • T & Z contain the lower and upper integration limits • The integration variable *reg* does not contain any useful information ∫ FN cannot call a function which calls FN, SOLVE or ∫ FN and uses 140 (!) bytes of memory. ∫ FN can be used in a program
STAT	**Statistics operations in conjunction with Σ** See section *Summation* above

PROB	**Probability operations**	
Cn,r	Combinations of r values taken from a group of n different values where the different order of the r values do *not* count separately	
Pn,r	Combinations of r values taken from a group of n different values where the different order of the r values count separately	
x!	Faculty of X or Gamma of X-1, also for negative X	
R	RANDOM	Return random number $0 \leq X < 1$
	SEED	Use X as the seed for the random number generator
MEM	**Memory overview** Displays the amount of free bytes for program and data on the left	
VAR	Display list of non-zero variables A..Z • Pressing CLEAR deletes a variable • Use ▲ and ▼ to scroll thru variables	
PGM	Display list of programs labelled A..Z. For each program the memory requirement in bytes is displayed • Pressing CLEAR deletes the program • Pressing SHOW displays a program checksum value which can be used to verify correct program entry • Pressing PRGM enters program mode and display the selcted program • Use ▲ and ▼ to scroll thru programs labels	

Programming

Memory	One program instruction requires 1.5 bytes of memory. Numbers 0..99 occupy 1.5 bytes, all other numbers 9.5 bytes
PRGM	Activate/deactivate programming mode
Program editing	• Newly typed instructions will be inserted after the currently displayed line • ← deletes an instruction and backs up the program counter • Program instructions are printed in clear text and numbered. Preceeding the line number the program's label is displayed • "C" switches to RUN mode. Use the CLEAR menu to insert a CLx instruction
▲ and ▼	RUN mode: Single-steps thru program. Displays next program step while key is held down. Backstepping doesn't execute any code PRGM mode: Step thru program instructions. Holding down the key scrolls thru instructions
GTO . .	RUN and PRGM mode: Jump to the top of the program memory. After pressing PRGM "PRGM TOP" will be displayed
GTO . A..Z nn	RUN and PRGM mode: Goto line number nn of program starting at specified label.
GTO A..Z, (i)	RUN mode: Jump to the specified label PRGM mode: Insert jump instruction to the specified label Values 1..26 of index register i correspond to program labels A..Z. Only the absolute value of the integer part of i is used for indexing
XEQ A..Z, (i)	RUN mode: Execute program starting at the specified label

	PRGM mode: Insert subroutine call to the specified label. At most 4 subroutine calls can be nested
INPUT A..Z, i, (i)	Prompt the user for a variable. The user can accept the displayed value of the variable (which is its current value) or execute various operations to calculate the desired value. Press R/S or to continue the program. Note that the input value is also stored in the X register

Menus Related to Programming

LBL/RTN	**Label & return instructions for programming**
LBL	Label A..Z. A label can only occur once in program memory space
RTN	Return instruction
PSE	Pause – will halt the program for about 1 second
LOOP	**Loop instructions for programming**
DSE A..Z, i, (i)	Decrement and skip if equal (or less). Decrements a variable and skips next instruction if result is equal or less than a given limit. The variable must be in the form "ccccccc.fffii" where: ccccccc: Current value of the loop counter, can be negative. This value changes with every execution of DSE (or ISG) fff : Limit value for the loop counter, does not change ii : Loop counter decrement (increment for ISG), does not change. If ii=0 then ii=1 is assumed
ISG A..Z, i, (i)	Increment and skip if greater. Increments a variable and skips next instruction if result is greater than a given limit. For the formatting requirements of the variable see DSE above
FLAGS	**Flag instructions for programming**
SF	Set flag. There are seven flags 0..6. • The status of flags 0..3 is shown in the display • If flag 5 is set an overflow condition will halt a program. If lag 5 is clear OVERFLOW is briefly displayed and the program continues using a value of 1E500 • Flag 5 is automatically set upon overflow
CF	Clear flag
FS?	Test whether flag is set: Flag set: Execute next program instruction Flag clear: Skip next program instruction
TESTS	**Comparisn instructions for programming**
x?y	Comparisns between X & Y register. Relation true: Execute next program instruction Relation false: Skip next program instruction Submenu choices: ≠y <y >y =y
x?0	Comparisns between X register and 0: Submenu choices: ≠0 <0 >0 =0