WP 34S
pocket reference

Firmware V3.2

A α	B β	C Γγ	D Δδ	⇨ E ε	CPX MODE F Φφ
αSTO G Γγ	αRCL VWα+ H Χχ	R↑ I ι	f	G	h
ENTER Hη		⇄ J ( )	± ¬ K κ	π L Λλ	⇦ CLα

	7 & M μ	8 | N ν	9 ≠ O Ωω	/ \ P Ππ
!	4 Q	5 R Ρρ	6 S Σσ	× T τ
?	1 TEST Θθ	2 U	3 X.FCN V	− W
EXIT OFF ⇧	0 ␣ Ψψ	. ./, X Ξξ	R/S Y υ	+ Z ζ

⇨ → ← ↑ ↓ √ ∫ ∞ ^ ↕

CPX accented characters

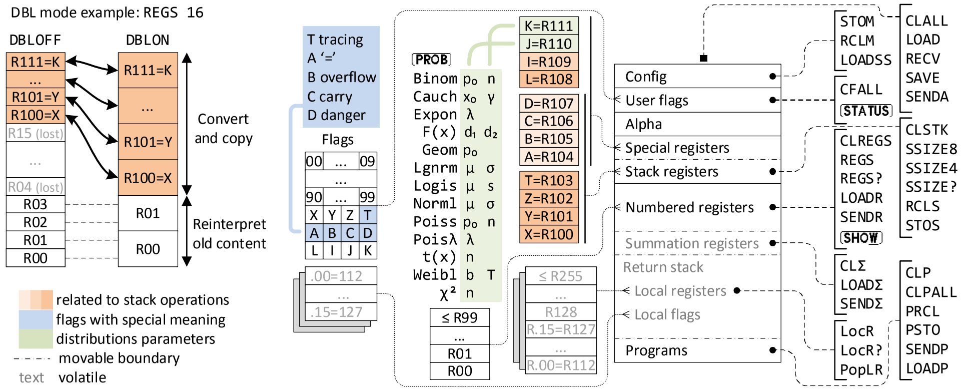
R↑ 0 ° 1 2 ^{2} ^{3} A B c ^{c} d e k m n p q u w X ^{X} Y µ $^{-1}$ * ∞

TEST < ≤ = ≈ ≥ > [ ] { }

./, , : ; ' " * @ _ ~ % $ € £ ¥ ⊙ ⊕ 🖶 ` #

Andrew Nikitin

2014-04-14



X	Y	Z	T	A	B	C	D	L	I	J	K	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09	.10	.11	.12	.13	.14	.15
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

This document briefly describes commands of WP 34s programmable calculator with firmware version 3.2. Please refer to WP 34S Owner's Manual for the definitive guide.

Command catalogs

[MATRIX]

DET	M+×	M-ROW	M.REG
LINEQS	M^{-1}	M×	nCOL
MROW+×	M-ALL	M.COPY	nROW
MROW×	M-COL	M.IJ	TRANSP
MROW⇄	M-DIAG	M.LU	

[MODE]

12h	D.MY	PowerF	SETTIM
1COMPL	E3OFF	RCLM	SETUK
24h	E3ON	RDX,	SETUSA
2COMPL	ExpF	RDX.	SIGNMT
BASE	FAST	REGS	SLOW
BestF	FRACT	RM	SSIZE4
DBLOFF	JG1582	SEPOFF	SSIZE8
DBLON	JG1752	SEPON	STOM
DENANY	LinF	SETCHN	UNSIGN
DENFAC	LogF	SETDAT	WSIZE
DENFIX	LZOFF	SETEUR	Y.MD
DENMAX	LZON	SETIND	🖶DLAY
DISP	M.DY	SETJPN	🖶MODE

[PROB]

Binom	Expon	Geom	Norml
Binom$_P$	…	…	…
Binom$_u$	F$_P$(x)	Lgnrm	Poiss
Binom^{-1}	F$_u$(x)	…	…
Cauch	F(x)	Logis	Poisλ
…	F^{-1}(p)	…	…

$t_P(x)$	$t^{-1}(p)$	$\Phi_u(x)$	χ^2INV
$t_u(x)$	Weibl	$\varphi(x)$	χ^2_P
$t(x)$	…	χ^2	χ^2_u

P.FCN

BACK	gFLP	REGS?	y⇄
BASE?	gPLOT	RESET	z⇄
CASE	gSET	RM?	αGTO
CFALL	GTOα	RTN+1	αOFF
CLALL	INC	R-CLR	αON
CLPALL	ISE	R-COPY	αXEQ
CLREGS	ISZ	R-SORT	⇄
CLSTK	LOADP	R-SWAP	🖶ADV
CLα	LOADR	SAVE	🖶CHR
CNST	LOADSS	SENDA	🖶$^c r_{XY}$
DEC	LOADS	SENDP	🖶PLOT
DROP	LocR	SENDR	🖶PROG
DSL	LocR?	SENDΣ	🖶r
DSZ	MEM?	SKIP	🖶REGS
END	MSG	SMODE?	🖶STK
ERR	NOP	SSIZE?	🖶TAB
FF	PopLR	STOS	🖶WIDTH
FLASH?	PRCL	TICKS	🖶α
f'(x)	PROMPT	t⇄	🖶α+
f"(x)	PSTO	VIEWα	🖶Σ
gCLR	PUTK	VWα+	🖶+α
gDIM	RCLS	WSIZE?	🖶#
gDIM?	RECV	XEQα	

STAT

COV	SUM	$\hat{x}$	σ_W
L.R.	s_W	ε	%Σ
SEED	s_{XY}	ε_m	
SERR	x_g	ε_P	
$SERR_W$	x_W	σ	

SUMS

nΣ	Σlnxy	Σx^{2}y	Σy^2
Σln^{2}x	Σlny	Σxlny	Σylnx
Σln^{2}y	Σx	Σxy	
Σlnx	Σx^2	Σy	

TEST

BC?	FP?	LEAP?	x≤?
BS?	FS?	M.SQR?	x=+0?
CNVG?	FS?C	NaN?	x=-0?
DBL?	FS?F	ODD?	x≈?
ENTRY?	FS?S	PRIME?	x≥?
EVEN?	gPIX?	REALM?	x>?
FC?	INTM?	SPEC?	∞?
FC?C	INT?	TOP?	⎙?
FC?F	KEY?	XTAL?	
FC?S	LBL?	x<?	

X.FCN

$\sqrt[3]{x}$	DECOMP	H_n	LNβ
AGM	DEG→	H_{np}	MANT
ANGLE	dRCL	H.MS+	MASKL
ASR	DROP	H.MS−	MASKR
B_n	D→J	IDIV	MAX
B^*_n	erf	iRCL	MIN
BATT	erfc	Iβ	MIRROR
CB	EXPT	$I\Gamma_p$	MOD
CEIL	e^x-1	$I\Gamma_q$	MONTH
DATE	FB	J→D	NAND
DATE→	FIB	LCM	nBITS
DAY	FLOOR	LJ	NEIGHB
DAYS+	g_d	L_n	NEXTP
DBL×	g_d^{-1}	LN1+x	NOR
DBL/	GCD	$L_n\alpha$	P_n
DBLR	GRAD→	LNΓ	RAD→

RDP	SLVQ	$^x\sqrt{y}$	Γ
RESET	SR	YEAR	γ_{XY}
RJ	sRCL	αDATE	Γ_{XY}
RL	STOPW	αDAY	ΔDAYS
RLC	TIME	αIP	ζ
ROUND	T_n	αLENG	$(-1)^X$
ROUNDI	ULP	αMONTH	→DATE
RR	U_n	αRC#	%+MG
RRC	VERS	αRCL	%Σ
RSD	WDAY	αRL	%MG
SB	WHO	αRR	%MRR
SDL	W_m	αSL	%T
SDR	W_p	αSR	×MOD
SEED	W^{-1}	αSTO	^MOD
SIGN	XNOR	αTIME	
SINC	x^3	α→x	
SL	x→α	β	

CPX X.FCN

$^{c3}\sqrt{x}$	^{c}DROP	^{c}LN1+x	$^cW^{-1}$
^{c}AGM	$^ce^x$-1	^{c}LNβ	$^cx^3$
^{c}CNST	^{c}FIB	^{c}LNΓ	$^{cx}\sqrt{y}$
^{c}CONJ	^cg_d	^{c}SIGN	$^c\beta$
^{c}CROSS	$^cg_d^{-1}$	^{c}SINC	$^c\Gamma$
^{c}DOT	^{c}IDIV	^cW_p	$^c(-1)^x$

WP 34S commands

The entry header contains the following information:

1) name of the command
2) effect on the stack
3) clues on how to enter the command

10^x $x \rightarrow r$ CPX f

Common antilogarithm, See also LOG_{10}

12h - → - MODE

12h time display mode. This will make a difference in αTIME only.

1COMPL - → - MODE

Set 1's complement mode for integers.

1/x $x \rightarrow r$ CPX f
CPX B

Inverse of a number.

24h - → - MODE

24h time display mode. Compare 12h.

2COMPL - → - MODE

Set 2's complement mode for integers.

2^x $x \rightarrow r$ CPX f

See also LOG_2

$\sqrt[3]{x}$ $x \rightarrow r$ CPX X.FCN

Cubic root.

ABS $x \rightarrow r$ CPX f

Absolute value.

ACOS **x → θ** CPX g

Principal value of arccos(x).

ACOSH **x → r** CPX g HYP⁻¹

$$\operatorname{csch}^{-1} x = \ln(x + \sqrt{x^2 - 1})$$

AGM **y x → r** CPX X.FCN

Arithmetic-geometric mean.
Starts with a_0=a, b_0=b and iterates

$$a_{n+1} = \frac{1}{2}(a_n + b_n);\ b_{n+1} = \sqrt{a_n b_n}$$

AGM can be expressed in terms of complete elliptic integral of first kind K(k)

$$\operatorname{agm}(a,b) = \frac{(a+b)\pi}{4K\left(\frac{a-b}{a+b}\right)}$$

ALL n **- → -** h

Numeric display format that shows all decimals whenever possible.
$x \geq 10^{13}$ is displayed in SCI or ENG with the maximum number of digits necessary (see SCIOVR and ENGOVR). The same happens if $x < 10^{-n}$ and more than 12 digits are required to show x completely.

AND **y x → r** h

INT: bitwise AND.
DECM: logical AND; x and y meaning is ‘false’, when zero and ‘true’ when any other real number.

ANGLE **y x → θ** X.FCN

arctan(y/x) corrected for quadrant and singularities.

ASIN **x → θ** CPX g

Principal value of arcsin(x)

ASINH **x → r** CPX g HYP⁻¹

$$\sinh^{-1} x = \ln\left(x + \sqrt{x^2 + 1}\right)$$

ASR n **m → r** X.FCN

Right shift with sign propagation, n≤63. Corresponds to a division by 2.

ATAN **x → θ** CPX g

Principal value of arctan(x).

ATANH **x → r** CPX g HYP⁻¹

$$\tanh^{-1} x = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$$

BACK n **- → -** P.FCN

Jump n steps backwards (0≤n≤255).
`BACK 1` goes to the previous program step. If `BACK` attempts to cross an `END`, an error is thrown. Reaching step 000 stops program execution. Compare `SKIP`.

BASE n **- → -** MODE
BASE 10 f 10
BASE 16 g 16
BASE 2 f 2
BASE 8 g 8

Set integer mode with base 2≤n≤16. Popular bases are directly accessible on the keyboard. `BASE 0` sets DECM, `BASE 1` calls `FRACT`. See there.
ATTENTION: this command converts stack contents with possible truncation or loss of

precision. Other registers stay as they are. `BASE 10` is not DECM.

BASE? **- → r** **P.FCN**

INT: current integer base
DECM: integer base set before DECM

BATT **- → volts** **X.FCN**

DECM: Battery voltage in the range 1.9…3.4V.
INT: Battery voltage in units of 100mV.

BC? n **m → m** **TEST**

Test if n-th bit in X is 0.

BestF **- → -** **MODE**

Select the best curve fit model, maximizing the correlation..

Binom **x → p** **PROB**
Binom_P **x → r**
Binom_u **x → p**
Binom^{-1} **p → x**

Binomial distribution, the probability of a success p_0 in J and the sample size n in K.

B_n **n → r** **X.FCN**
B_{n*}

B_n returns the Bernoulli number for an integer n>0 given in X. B_{n*} works with the old definition instead.

$$B_n = (-1)^{n+1} n \cdot \zeta(1-n)$$
$$B_n^* = \frac{2 \cdot (2n)!}{(2\pi)^{2n}} \zeta(2n)$$

BS? n **m → m** **TEST**

Test if n-th bit in X is set.

CASE s **- → -** **P.FCN**

Like SKIP, but takes the number of steps to skip from s.

CAT **- → -** **h**

Alpha labels browser.
0, 1, or 2 – quick jump to RAM, LIB or BUP
▲, ▼ – browse alpha labels
f▲, f▼ – browse programs (separated by END statements)
ENTER↑ – go to alpha label with search
XEQ – execute alpha label with search; programming mode: insert XEQ'lbl'
GTO – programming mode: insert GTO'lbl'
R/S – execute alpha label without search
RCL, STO – PRCL, PSTO
f CLP – delete program in RAM or LIB

Cauch **x → p** **PROB**
$Cauch_P$ **x → r**
$Cauch_u$ **x → p**
$Cauch^{-1}$ **p → x**

Cauchy-Lorentz distribution (also known as Lorentz or Breit-Wigner distribution) with the location x_0 in J, the shape γ in K.

CB n **m → r** **X.FCN**

Clear n-th bit in X.

CEIL **x → r** **X.FCN**

Smallest integer ≥x

CF n **- → -** **g**

Clear flag n.

CFALL **- → -** **[P.FCN]**
Clear all user flags

CLALL **- → -** **[P.FCN]**
Clear all registers, flags, and programs in RAM if confirmed. Not programmable. Compare RESET.

CLP **- → -** **[f]**
Clear the current program, i.e. the one the program pointer is in. Not programmable.

CLPALL **- → -** **[P.FCN]**
Clear all programs in RAM if confirmed. Not programmable.

CLREGS **- → -** **[P.FCN]**
Clear all global and local general purpose registers (see REGS and LOCR), keep the contents of the stack, L, and I.

CLSTK **... → ...** **[P.FCN]**
Clear all stack registers currently allocated (i.e. X through T or X through D, respectively), keep all other registers.

CLx **x → 0** **[h]**
Clear register X, disable stack lift.

CLα **- → -** **[P.FCN]**
Input: [h][CLx]
Clear the alpha register.

CLΣ **- → -** **[g]**
Release the memory allocated for the summation registers.

CNST n - → r (P.FCN)

ᶜCNST n - → 0 r (CPX)(h)(X.FCN)

Indirect addressing of the content at position n in CONST catalog.

CNVG? n - → - (TEST)

Check for convergence by comparing x and y as determined by the lowest five bits of n=a+4b+16c

a – lowest two bits, tolerance limit:
0 = 10^{-14},
1 = 10^{-24},
2 = 10^{-32},
3 = choose the best for the mode set: 0 for single precision and 2 for double precision.

b – the next two bits, determines the comparison mode using the tolerance limit set:
0 = compare the real numbers x and y relatively,
1 = compare them absolutely,
2 = check the absolute difference between the complex values x+i·y and z+i·t,
3 = works as 0 so far.

c – the top bit, tells how special numbers are handled:
0 = NaN and infinities are considered converged,
1 = they are not considered converged.

COMB y x → r (CPX) (f)

The number of possible sets of y items taken x at a time. $C_{y,x} = \frac{y!}{x!(y-x)!}$. Compare PERM.

^{c}CONJ **y x → −y x** (CPX) (X.FCN)

Flip the sign of y, the complex conjugate of x_c.

CONST **- → r** (h)

Catalog of physical and mathematical constants.

- 1/2 =0.5
- a [d] Gregorian year
- a_0 [m] Bohr radius
- a_m [m] semi-major axis of the Moon's orbit
- $a_\oplus$ [m] semi-major axis of the Earth's orbit=1 AU
- c [m/s] speed of light in vacuum
- c_1 [m^2W] first radiation constant
- c_2 [m·K] second radiation constant
- e [C] electron charge
- eE Euler's e
- F [C/mol] Faraday's constant
- Fα Feigenbaum α
- Fδ ... and δ
- g [m/s^2] standard earth acceleration due to gravity
- G [$m^3/(kg\cdot s^2)$] Newton's gravitation constant
- G_0 [1/Ω] conductance quantum
- G_C Catalan's constant
- g_e Landé's electron g-factor
- GM [m^3/s^2] gravitation constant times earth mass, WGS84
- h [J·s] Planck constant
- ħ [J·s] h/2π
- k [J/K] Boltzmann constant
- K_J [Hz/V] Josephson constant
- l_P [m] Planck length
- m_e [kg] electron mass
- M_m [kg] mass of the Moon
- m_n [kg] neutron mass
- m_p [kg] proton mass
- M_P [kg] Planck mass
- m_u [kg] atomic mass unit
- m_uc^2 [J] energy equivalent of atomic mass unit
- m_μ [kg] muon mass
- $M_\odot$ [kg] mass of the Sun
- $M_\oplus$ [kg] mass of the Earth
- N_A [1/mol] Avogadro number
- NaN
- p_0 [Pa] standard atmospheric pressure
- q_P [A·s] Planck charge
- R [J/(mol·K)] molar gas constant
- r_e [m] classical electron radius
- R_K [Ω] von Klitzing const.
- R_∞ [1/m] Rydberg const.
- R_m [m] mean radius of the Moon
- $R_\odot$ [m] ... of the Sun
- $R_\oplus$ [m] ... of the Earth
- Sa [m] semi major axis of WGS84
- Sb [m] semi minor axis of

WGS84

Se^2 first eccentricity squared of WGS84

Se'^2 second eccentricity squared of WGS84

Sf^{-1} flattening parameter of WGS84

T_0 [K] 0°C

t_P [s] Planck time

T_P [K] Planck temperature

V_m [m³/mol] molar volume of an ideal gas

Z_0 [Ω] characteristic impedance of vacuum

α fine-structure constant

γEM Euler-Mascheroni constant

γ_p [1/(s·T)] proton gyromagnetic ratio

ε_0 [A·s/(V·m)] electric constant or vacuum permittivity

λ_C [m] Compton wavelengths of the electron

λ_{Cn} [m] … neutron

λ_{Cp} [m] … proton

μ_0 [V·s/(A·m)] magnetic constant or vacuum permeability

μ_B [J/T] Bohr's magneton

μ_e [J/T] magnetic moment of electron

μ_n [J/T] … neutron

μ_p [J/T] … proton

μ_μ [J/T] … muon

μ_u [J/T] nuclear magneton

σ_B [W/(m²K⁴)] Stefan-Boltzmann constant

Φ Golden ratio

Φ_0 [V·s] magnetic flux quantum

ω [rad/s] angular velocity of the Earth, WGS84

−∞, ∞, #

CONV **x → r** (h)

Catalog of unit conversions.

CORR **- → r** (g)

Correlation coefficient for the current statistical data and curve fitting model. For linear model

$$r = \frac{s_{XY}}{s_X s_Y}$$

For arbitrary model R(x), the value

$$r^2 = 1 - \frac{\sum[\mathrm{R}(x_i) - y_i]^2}{\sum(\bar{y} - y_i)^2}$$

is coefficient of determination. r^2=0.93 means that 93% of total variation of y is due to x.

COS **θ → r** (CPX) (f)

Cosine.

COSH **x → r** (CPX) (HYP)

Hyperbolic cosine, $\cosh x = \frac{e^x+e^{-x}}{2}$

COV **- → r** (STAT)

Population covariance of two data sets. It depends on the fit model. See s_{XY} for the sample covariance. For linear model

$$\mathrm{COV}_{XY} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n^2}$$

ᶜCROSS **t z y x → 0 r** (CPX) (X.FCN)

Interpret x and y as Cartesian components of a first vector, and z and t as those of a second one, and return X=r=x·t–y·z, Y=0, dropping two stack levels.

DATE **- → dc** (X.FCN)

Date from the real time clock. Actual presentation depends on date format. See `D.MY`, `M.DY`, and `Y.MD`. `DATE` shows the day of week in the dot matrix.

DATE→ **dc → y m d** (X.FCN)

Parse the date according to current date format and calculate Z=year, Y=month, X=day.

DAY **dc → r** (X.FCN)

Extract the day number from the date code.

DAYS+ **dc x → dc1** (X.FCN)

Add x days to a date in Y, display the resulting date including the day of week in the same format as WDAY does.

DBLOFF - → - (MODE)
DBLON
Toggle double precision mode. Setting becomes effective in DECM only and is indicated by D in the dot matrix.

DBL? - → - (TEST)
Test if double precision mode is turned on.

DBLR lo hi m → r (X.FCN)
DBL/ lo hi m → r (X.FCN)
DBL× y x → lo hi (X.FCN)
Double word length commands for integer remainder, multiplication and division. DBLR and DBL/ accept a double size dividend in Y and Z (most significant bits in Y), the divisor in X as usual, and return the result in X. DBL× takes x and y as factors as usual but returns their product in X and Y (most significant bits in X)..

DEC s - → - (P.FCN)
Decrement s by 1.

DECM - → - (f)(H.d)
Set decimal floating point mode.

DECOMP x → num den (X.FCN)
Decompose x (after converting it into an improper fraction, if applicable), into numerator (in Y) and denominator (in X). Reversible by division.

DEG **- → -** **[g]**

Set angular mode to degrees.

DEG→ **x → θ** **[X.FCN]**

Convert x degrees to current angular units.

DENANY **- → -** **[MODE]**

Set fraction display subformat, which allows any denominator up to the value set by DENMAX may appear.
Example: If DENMAX=5 then DENANY allows denominators 1, 2, 3, 4, and 5.

DENFAC **- → -** **[MODE]**

Set fraction display subformat, which allows integer factors of the DENMAX as denominators.
Example: If DENMAX=60 then DENFAC will allow denominators 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, and 60.

DENFIX **- → -** **[MODE]**

Set fraction display subformat where the only denominator allowed is DENMAX.

DENMAX **x → x** **[MODE]**
1 → r

Set the maximum allowed denominator in fraction display mode. Valid 2≤ x≤9999. For x=1, recall current setting.

DET **mat → r** **[MATRIX]**

Determinant of a square matrix. Matrix descriptor is in X. The matrix is not modified.

DISP n - → - (MODE)

Change the number of decimals shown while keeping the basic display format (FIX, SCI, ENG) as is. In ALL, DISP changes the switch-over point (see ALL).

[c]DOT t z y x → 0 r (CPX) (X.FCN)

X and Y are Cartesian components of a first vector, Z and T of a second one. Return r=x·z+y·t in X, 0 in Y.

dRCL s - → r (P.FCN)

Interpret s as double precision and recall it.

DROP x → - (CPX) (X.FCN)

Drop X.

DSE s - → - (f)

Given *cccccc.fffii* in s, DSE decrements s by *ii*, skips next program line if *ccccccc≤fff*. If s has no fractional part then *fff*=0 and *ii*=1.

DSL s - → - (P.FCN)

Like DSE but skips if *ccccccc<fff*.

DSZ s - → - (P.FCN)

Decrement s by 1, and skip the next step if |s|<1 thereafter.

D.MY - → - (MODE)

Set the *dd.mmyyyy* date format.

D→J dc → r (X.FCN)

Julian day number of a date. To get julian day number for 0:00:00 of that date, subtract 0.5. See also JG...

E3OFF - → - (MODE)
E3ON
Toggle the thousands separators for DECM.

END - → - (P.FCN)
Last command in a routine and a terminator for local labels search. Cannot be skipped by false test. Works like RTN in all other aspects.

ENG n - → - (h)
Engineer's display format. Exponent is always a multiple of 3.

ENGOVR - → - (h)(ENG)(ENTER)
Use ENG mode to display numbers that cannot be displayed in ALL or FIX. Compare SCIOVR.

ENTER↑ x → x x (CPX)
Push x on the stack, disable stack lift.

ENTRY? - → - (TEST)
Test the entry flag. This internal flag is set if:
* any character is entered in alpha mode, or
* any command is accepted for entry (be it via (ENTER↑), a function key, or (R/S) with a partial command line).

erf x → r (X.FCN)
erfc
Error function and its complement.

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$
$$\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$$

ERR n - → - (P.FCN)
Raise the error *n*. The consequences are the

same as if error *n* really occurred, so e.g. a running program will be stopped. Compare MSG.

1	Domain error	14	Word size too small
2	Bad time or date	15	Too few data points
3	Undefined op-code	16	Invalid parameter
4	+∞ error	17	I/O error
5	–∞ error	18	Invalid data
6	No such label	19	Write protected
7	Illegal operation	20	No root found
8	Out of range error	21	Matrix mismatch
9	Bad digit error	22	Singular error
10	Too long error	23	Flash is full
11	RAM is full	24	No crystal installed
12	Stack clash	25	∫ ≈
13	Bad mode error		

EVEN? **x → x** **TEST**

Test if x is integer and even.

e^x **x → r** CPX **f**

Exponent. See also LN.

ExpF **- → -** **MODE**

Set the exponential curve fit model

$$R(x) = a_0 e^{a_1 x}$$

Expon **x → p** **PROB**
Expon$_P$ **x → r**
Expon$_u$ **x → p**
Expon^{-1} **p → x**

Exponential distribution, λ in J.

EXPT **x → r** **X.FCN**

Exponent h of the number $x=m\cdot10^h$. Compare MANT.

e^x−1 **x → r** CPX **X.FCN**

For x≈0, returns a more accurate result for the fractional part than e^x does. See also LN1+x.

FAST **- → -** **MODE**

Set the processor speed to 'fast'. This is start-up default and is kept for fresh batteries. Compare SLOW.

FB n **m → r** **X.FCN**

Invert ('flip') the n-th bit in x.

FC? n **- → -** **TEST**

Test if the n-th user flag is clear.

FC?C n **- → -** **TEST**
FC?F n
FC?S n

Test if the n-th user flag is clear. Clear, flip, or set this flag after testing.

FF n **- → -** **P.FCN**

Flip the n-th user flag.

FIB **x → r** CPX **X.FCN**

INT: Fibonacci number.
DECM: extended Fibonacci number.

FILL **x → … x x** CPX **g**

Copy x to all stack levels.

FIX n **- → -** **h**

Fixed point display format.

FLASH? **- → r** **P.FCN**

Number of free words in FM.

FLOOR **x → r** **X.FCN**
Largest integer ≤x.

FP **x → r** CPX **g**
Fractional part of x.

FP? **x → x** **TEST**
Test if x has a nonzero fractional part.

FRACT **- → -** **MODE**
Switch to fraction display mode, keep the format as set by PROFRC or IMPFRC earlier.

FS? n **- → -** **TEST**
Test if n-th user flag is set.

FS?C n **- → -** **TEST**
FS?F n
FS?S n
Test if n-th user flag is set. Clear, flip, or set this flag after testing.

F_P(x) **x →**
r **PROB**
F_u(x) **x → p**
F(x) **x → p**
F^{-1}(p) **p → x**
Fisher's F-distribution. The degrees of freedom are in J and K.

f'(x) lbl **x → 0 0 0 f'** **P.FCN**
First derivative of the function f(x) at position x. f(x) must be specified in a routine starting with LBL lbl. After return, Y, Z, and T are cleared x is in L.
f'(x) looks for a user routine labeled 'δx',

which returns a fixed step size dx in X. If 'δx' is not defined, dx=0.1. Then, `f'(x)` evaluates f(x) at ten points equally spaced in the interval x±5 dx. If you expect any irregularities within this interval, change δx to exclude them.

`f"(x) lbl` **x → 0 0 0 f"** **P.FCN**

Like `f'(x)` but return the second derivative.

GCD **y x → r** **X.FCN**

Greatest Common Divisor of x and y. Always positive.

`gCLR s` **y x → -** **P.FCN**

Clear the pixel at position x, y in the graphic block starting at register address s. Valid ranges are 0≤x≤w–1 and 0≤y≤h–1. Pixel 0, 0 is top left. See `gDIM` for more.

g_d **x → r** CPX **X.FCN**

g_d^{-1} **x → r** CPX **X.FCN**

Gudermann function and its inverse.

$$g_d(x) = \int_0^x \frac{d\xi}{\cosh \xi}, g_d^{-1}(x) = \int_0^x \frac{d\xi}{\cos \xi}$$

`gDIM s` **y x → y x** **P.FCN**

Initialize a set of registers (a graphic block) for graphic data starting at address s, featuring x (≤166) pixel columns and y pixel rows. For x≤0, the width w is set to 166. For y≤0, the height h is set to 8. The first two bytes in the block are reserved to hold w and $\check{h} =$

$\left\lfloor\frac{h+7}{8}\right\rfloor$

The number of registers needed for the set is $n = \left\lfloor\frac{w\cdot\breve{h}+9}{8}\right\rfloor$ in startup standard mode. E.g. 21 registers are required for maximum width and standard height.

The command can be exactly emulated in integer mode by storing $256 \cdot \breve{h} + w$ in the first register and clearing the rest. See 🖶PLOT.

gDIM? s **- → h w** (P.FCN)

Recall Y=h and X=w for a graphic block starting at address s. See gDIM for more.

Geom **x → p** (PROB)

Geom$_P$ **x → r**

Geom$_u$ **x → p**

Geom^{-1} **p → x**

Geometric distribution: The cdf returns the probability for a first success after m=x Bernoulli experiments. The probability p_0 for a success in each such experiment is in J.

gFLP s **y x → -** (P.FCN)

gPIX? s **y x → y x** (TEST)

Flip or test the pixel at position x, y in the graphic block at address s. See gCLR for more.

gPLOT s **- → -** (P.FCN)

Display the top left sector of the graphic block (starting at address s) in the dot matrix section of the LCD. See gDIM for more.

GRAD **- → -** (g)

Set angular unit to gon (grad).

GRAD→ $x \to \theta$ (X.FCN)

Convert angle of x gon (grad) the current angular unit.

gSET s $y\ x \to$ - (P.FCN)

Set the pixel at position x, y in the graphic block starting at address s. See gCLR for more.

GTO lbl - → -

(GTO)(.)(A) or (B) (C) (D) – position at label
(GTO)(.)(▲) – top of current program
(GTO)(.)(▼) – top of next program
(GTO)(.)(.) – step 000

GTOα - → - (P.FCN)

Take the first three characters of alpha (or all if there are fewer than three) as a label and positions the program pointer to it.

H_n $n\ x \to r$ (X.FCN)
H_{np}

Hermite polynomials for probability (H_n) and for physics (H_{np}).

$$H_n(x) = (-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n}\left(e^{-\frac{x^2}{2}}\right)$$

$$H_{np}(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}\left(e^{-x^2}\right)$$

H.MS $x \to x$ (f)

Display X (containing decimal hours or degrees), in the format hhhh°mm’ss.dd” temporarily until any key.

H.MS+ **tc1 tc2 → tc3** X.FCN
H.MS–
Add or subtract times or degrees in the format *hhhh.mmssdd* in X and Y.

IDIV **y x → r** CPX X.FCN
Integer division, like / IP in DECM and like / in integer modes.

IMPFRC **- → -** g d/c
Fraction display mode. Displays numbers as improper fractions (e.g. 5/3 instead of 1 2/3). Numbers |x|≥100,000 display as decimals. Compare PROFRC.

INC s **- → -** P.FCN
Increment s by 1.

INTM? **- → -** TEST
Test if WP 34S is in integer mode.

INT? **x → x** TEST
Test if x is integer. Compare FP?.

IP **x → r** CPX f
Integer part of x.

iRCL s **- → r** X.FCN
Interpret s as integer data and recall it.

ISE s **- → -** P.FCN
Like ISG but skip if *ccccccc≤fff*

ISG s **- → -** g
Given *cccccc.fffii* in s, ISG increments s by *ii*, skipping next program line if then *ccccccc>fff*. If s has no fractional part then fff=0, and *ii*=1.

Neither *fff* nor *ii* can be negative, but *cccccc* can.

ISZ s - → - (P.FCN)

Increment s by 1, skipping next program line if |s|<1.

Iβ z y x → r (X.FCN)

Regularized (incomplete) beta. See also β

$$I_\beta = \frac{\beta_x(x,y,z)}{\beta(y,z)},$$

($\beta_x(x,y,z) = \int_0^x t^{y-1}(1-t)^{z-1}dt$)

$I\Gamma_p$ y x → r (X.FCN)

$I\Gamma_q$

Regularized (incomplete) gamma function (two flavors). See also γ_{XY}, Γ_{XY}

$$I\Gamma_p(x,y) = \frac{\gamma(x,y)}{\Gamma(x)}, I\Gamma_q(x,y) = \frac{\Gamma_u(x,y)}{\Gamma(x)}$$

JG1582 - → - (MODE)

JG1752

Set one of two dates of the Gregorian calendar introduction in different large areas of the world (1582-10-15 and 1752-09-14). Affects D→J and J→D.

J→D x → dc (X.FCN)

Convert x as a Julian day number to a date according to JG... and date format settings.

KEY? s - → - (TEST)

Test if a key was pressed while a program was running or paused. If no key was pressed in that interval, the next program step after

KEY? is executed, else it is skipped and the code of pressed key is stored in s. Key codes reflect the rows and columns on the keyboard.

KTP? s **- → r** **P.FCN**

Key type of a key code in s (see KEY?).
* 0 … 9 if digit 0 … 9,
* 10 if ., EEX, or +/-,
* 11 if f, g, or h
* 12 if any other key.

LASTx **- → r** CPX **RCL L**

Use RCL L in place of LASTx. Complex version takes imaginary part from reg. I

LBL lbl **- → -** f

Identify programs and routines for execution and branching.

LBL? lbl **- → -** **TEST**

Test for the existence of the label anywhere in program memory.

LCM **y x → r** **X.FCN**

Least Common Multiple of x and y. Always positive.

LEAP? **dc → dc** **TEST**
m → m

If X is a date in the date format, extract the year, and test for a leap year. If X is integer, test if it is leap year.

LgNrm **x → p** **PROB**
$LgNrm_P$ **x → r**
$LgNrm_u$ **x → p**
$LgNrm^{-1}$ **p → x**

Lognormal distribution with μ=ln($\bar{x}$) in J and σ=lnε in K. See $\bar{x}_g$ and **ε** below.
$LgNrm^{-1}$ returns x for a given probability p in X, with μ in J and σ in K.

LINEQS **mat vec i → r** **MATRIX**

Solve a system of linear equations Z·X=Y. Take a base register number in X, a vector descriptor in Y, and a descriptor of a square matrix in Z. Return the filled in vector descriptor in X.

LinF **- → -** **MODE**

Set linear curve fit model

$$R(x) = a_0 + a_1 x$$

LJ **m → y x** **X.FCN**

Left justify a bit pattern within the word size. Left justified word is placed in Y and the count (number of bitshifts necessary to left justify the word) in X.
Example: for word size 8, 10110_2 LJ results in x=3 and y=10110000_2.

LN **x → r** CPX **g**

Natural logarithm of x.

L_n **n x → r** **X.FCN**
$L_n\alpha$ **α n x → r** **X.FCN**

Laguerre polynomials and generalized polynomials.

$$L_n(x) = L_n^{(0)}(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}(x^n e^{-x})$$
$$L_n^{(\alpha)}(x) = \frac{x^{-\alpha}e^x}{n!}\frac{d^n}{dx^n}(x^{n+\alpha}e^{-x})$$

LN1+x **x → r** CPX **X.FCN**

For x≈0, this returns a more accurate result for the fractional part than ln(x) does.

LNβ **y x → r** CPX **X.FCN**

Natural logarithm of Euler's Beta function. See β.

LNΓ **x → r** CPX **X.FCN**

Natural logarithm of Γ(x).

LOAD **- → -** **P.FCN**

Restore the entire backup from FM, i.e. execute LOADP, LOADR, LOADSS, LOADS, and display *Restored*. Not programmable. Compare SAVE.

LOADP **- → -** **P.FCN**

Load the complete program memory from the backup and append it to the programs already in RAM. This only works if there is enough space, otherwise an error is thrown. Not programmable.

LOADR **- → -** **P.FCN**

Recover numbered general purpose registers from the backup (see SAVE). Lettered registers are not recalled. The number of registers copied, is the minimum number of the registers in the backup and in RAM.

LOADSS - → - [P.FCN]

Recover the system state from the backup.

LOADΣ - → - [P.FCN]

Recover the summation registers from the backup. Throw an error if there are none.

LocR n - → - [P.FCN]

Allocate n local registers (≤144) and 16 local flags for the current subroutine.

LocR? - → r [P.FCN]

Number of local registers currently allocated.

LOG_{10} x → r CPX [g][LG]

Inverse of 10^x

LOG_2 x → r CPX [g][LB]

Inverse of 2^x

LogF - → - [MODE]

Set logarithmic curve fit model

$$R(x) = a_0 + a_1 \ln x$$

Logis x → p [PROB]
$Logis_P$ x → r
$Logis_u$ x → p
$Logis^{-1}$ p → x

Logistic distribution with μ in J and s in K.

LOG_x y x → r CPX [g]

Logarithm of y for the base x.

LZOFF - → - [MODE]
LZON

Toggle leading zeros display. Relevant in bases 2, 4, 8, and 16 only.

L.R. - → a1 a0 [STAT]

Return the parameters a_1 and a_0 of the fit curve through the data points accumulated in the summation registers, according to the curve fit model selected (see LINF, EXPF, POWERF, and LOGF). For a straight line (LINF), a_0 is the y-intercept and a_1 the slope.

MANT x → r [X.FCN]

Mantissa m of the number $x=m \cdot 10^h$. Compare EXPT.

MASKL n - → r [X.FCN]
MASKR n

Generate a bit pattern where lowest (MASKL) or highest (MASKR) n bits are set.
Example: For WSIZE 8, MASKL 3 returns a mask word 11100000_2.

MAX y x → r [X.FCN]

Maximum of x and y.

MEM? - → r [P.FCN]

Number of free words in program memory, taking into account the local registers.

MIN y x → r [X.FCN]

Minimum of x and y.

MIRROR m → r [X.FCN]

Reflect the bit pattern in x (e.g. 00010111_2 becomes 11101000_2 for word size 8).

MOD y x → r [X.FCN]

y mod x. Compare RMDR.

MONTH dc → r (X.FCN)

Extract month number from a date.

MROW+× t z y x → t z y x (MATRIX)

Take a matrix descriptor x, a destination row number y, a source row number z, and a real number t. Multiply each element m_{zi} of (X) by t and add it to m_{yi}. The stack remains unchanged.

MROW× z y x → z y x (MATRIX)

Take a matrix descriptor x, a row number y, and a real number z. Multiply each element m_{yi} of (X) by z.

MROW⇄ z y x → z y x (MATRIX)

Take a matrix descriptor x and two row numbers y and z. Swap the contents of rows y and z in (X). The stack remains unchanged.

MSG n - → - (P.FCN)

Show the message for error n. This will be a temporary message. Compare ERR.

M+× z y x → r (MATRIX)

Take two matrix descriptors x and y, and a real number z. Return (X)+(Y)·z=(X). Thus a scalar multiple of one matrix is added to another matrix. The multiply/adds are done in internal high precision and results should be exactly rounded.

M^{-1} mat → mat (MATRIX)

Inverts square matrix in place. Doesn't alter the stack.

M-ALL **mat → r** (MATRIX)

Take a matrix descriptor in X and return a value suitable for ISG or DSL looping in that matrix. The loop shall process all elements in (X). The loop counter is for DSL if the descriptor was negative and for ISG otherwise.

M-COL **y mat → r** (MATRIX)

Loop counter for processing all elements m_{iy} of the matrix column y only. See M-ALL

M-DIAG **mat → r** (MATRIX)

Loop to process all elements along the matrix diagonal, i.e. all elements m_{ii} in (X). See M-ALL

M-ROW **y mat → r** (MATRIX)

Loop counter for processing all elements m_{yi} of matrix row y only. See M-ALL

M× **z y x → r** (MATRIX)

Take two matrix descriptors y and z, and the integer part of x as the base address of the result. Returns (Z)·(Y)=(X). All calculations are done in internal high precision (39 digits). The fractional part of x is updated to match the resulting matrix – no overlap checking is performed.

M.COPY **mat i → r** (MATRIX)

Take a matrix descriptor in Y and a base register number in X. Copy the matrix (Y) into registers starting at Rx. Return a properly formatted matrix descriptor in X.

M.DY **- → -** (MODE)

Set the *mm.ddyyyy* date format.

M.IJ **i mat → c r** **MATRIX**

Column (Y) and row (X) of a matrix that register i represents. Compare `M.REG`.

M.LU **mat → r** **MATRIX**

Take a descriptor of a square matrix in X. Transform (X) into its LU decomposition in-situ. The value in X is replaced by a descriptor that defines the pivots that were required to calculate the decomposition. The most significant digit is the pivot for the first diagonal entry, the next most significant for the second and so forth.

M.REG **c r mat → i** **MATRIX**

Take a matrix descriptor in X, a row number in Y, and a column number in Z. `M.REG` returns the register number in X. Compare `M.IJ`.

M.SQR? **mat → mat** **MATRIX**

Test if a matrix descriptor x defines square matrix.

NAND **y x → r** **X.FCN**

¬(x ∧ y). See AND.

NaN? **x → x** **TEST**

Test x for being ‘Not a Number’.

nBITS **x → r** **X.FCN**

Count set bits in x.

nCOL **mat → r** **MATRIX**

Number of columns of matrix (X).

NEIB **y x → r** **X.FCN**

Nearest machine-representable number to x in the direction toward y in the mode set. For x<y (or x>y), this is the machine successor (or predecessor) of x; for x=y it is y.

NEXTP **x → r** **X.FCN**

Next prime number greater than x.

NOP **- → -** **X.FCN**

Empty step

NOR **y x → r** **X.FCN**

¬(x ∨ y). See AND.

Norml **x → p** **PROB**
Norml$_P$ **x → r**
Norml$_u$ **x → p**
Norml^{-1} **p → x**

Normal distribution with an arbitrary mean μ in J and a standard deviation σ in K.

NOT **x → r** **h**

INT: Invert x bitwise.
DECM: 1 for x=0, and 0 for x≠0.

nROW **mat → r** **MATRIX**

Number of rows of matrix (X).

nΣ **- → r** **SUMS**

Number of accumulated statistical data points.

ODD **x → x** **TEST**

Test if x is integer and odd.

OFF **- → -** [h]

Turn off your WP 34S.

OR **y x → r** [h]

See AND

PERM **y x → r** CPX [g]

Number of possible arrangements of y items taken x at a time. Compare COMB.

$$P_{x,y} = \frac{y!}{(y-x)!}$$

P_n **n x → r** [X.FCN]

Legendre polynomials.

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^2]$$

Poiss **x → p** [PROB]
$Poiss_P$ **x → r**
$Poiss_u$ **x → p**
$Poiss^{-1}$ **p → x**

Poisson distribution with the number of successes in X, the gross error probability p_0 in J, and the sample size n in K. The Poisson parameter is calculated automatically. See Poisλ.

Poisλ **x → p** [PROB]
$Pois\lambda_P$ **x → r**
$Pois\lambda_u$ **x → p**
$Pois\lambda^{-1}$ **p → x**

Poisλ works like Poiss but with λ in J and without using K.

PopLR - → - P.FCN

Pop the local registers allocated to the current routine without returning. See `LocR` and `RTN`.

PowerF - → - MODE

Set power curve fit model

$$R(x) = a_0 x^{a_1}$$

PRCL - → - P.FCN

CAT: RCL

Copy the current program (from FM or RAM) and appends it to RAM, where it can then be edited. Can have duplicate program labels in RAM. Only works with enough space at destination.

PRIME? x → x TEST

Test if the absolute value of the integer part of x is a prime. The method is believed to work for integers up to $9 \cdot 10^{18}$.

PROFRC - → - f a b/c

Set fraction display mode. Display numbers as proper fractions (e.g. 1 2/3 instead of 5/3). Numbers |x|≥100,000 display as decimals. Compare `IMPFRC`.

PROMPT - → - P.FCN

Display alpha and stop program execution.

PSE n - → - h

Refresh the display and pause program execution for n ticks (see `TICKS`), 0≤n≤99. The pause terminates when you press a key.

PSTO **- → -** **P.FCN**
CAT: STO

Copy the current program from RAM and append it to the FM library. Not programmable. The program must have at least one alpha `LBL`, preferably at its beginning. If a program with the same label already exists in the library it is deleted first.

PUTK s **- → -** **P.FCN**

Stop program execution and place key code from s in the keyboard buffer, resulting in immediate execution of the corresponding keystroke. After that, **R/S** is required to resume program execution.

RAD **- → -** **g**

Set angular unit to radians.

RAD→ **x → θ** **X.FCN**

Convert x radians to current angular unit.

RAN# **- → r** **f**

DECM: random number between 0 and 1.
INT: random bit pattern for the word size set.

RCL s **- → r** **CPX**

Recall the number from the source register.

RCLM s **- → -** **RCL MODE**

Recall modes stored by `STOM`. No need to press **h**.

RCLS s **- → … t z y x** **P.FCN**

Recall 4 or 8 values from a set of registers starting at address s, and push them on the stack. This is the converse command of STOS.

RCL+ s **x → r** CPX
RCL− s CPX
RCL× s CPX
RCL/ s CPX
RCL↑ s RCL ▲
RCL↓ s RCL ▼

Recall s, execute operation and push the result on the stack. E.g. RCL−12 subtracts r12 from x and displays the result (acting like RCL 12 −, but without losing a stack level). ᶜRCL−12 subtracts r12 from x and r13 from y. RCL↑ (↓) replaces x with the maximum (minimum) of s and x.

RDP n **x → r** X.FCN

Round x to n decimal places (0≤n≤99), taking the RM setting into account. See RM and compare RSD.

RDX. **- → -** MODE
RDX, ./,

Toggle the radix mark.

REALM? **- → -** TEST

Test if in real mode (DECM).

RECV **- → -** P.FCN

Prepare to receive data via serial I/O. See SEND...

REGS n **- → -** MODE

Specify the number of global general purpose registers. 0≤n≤100

REGS? - → r (P.FCN)

Number of global general purpose registers (0…100).

RESET - → - (P.FCN)

If confirmed, execute CLALL and reset all modes to start-up default, i.e. 24h, 2COMPL, ALL 0, DBLOFF, DEG, DENANY, DENMAX 0, D.MY, E3ON, LinF, LocR 0, LZOFF, PROFRC, RDX., REGS 100, SCIOVR, SEPON, SSIZE4, WSIZE 64, and finally DECM. See these commands for more. Not programmable.

RJ m → y x (X.FCN)

Right justify. Example: 101100 RJ results in Y=1011 and X=2. See LJ.

RL n x → r (X.FCN)

RLC n x → r (X.FCN)

Rotate left/rotate left through carry. For RL, 0≤n≤63. For RLC, 0≤n≤64.

RM n - → - (MODE)

Set floating point rounding mode. This is only used when converting from the high precision internal format to packed real numbers. It will not alter the display nor change the behavior of ROUND. The following modes are supported:

0: round half even; 0.5 rounds to next even number (default).

1: round half up; 0.5 rounds up ('businessman's rounding').

2: round half down; 0.5 round down.

3: round up; round away from 0.

4: round down; round towards 0 (truncate).
5: ceiling; round towards +∞.
6: floor; round towards –∞.

RMDR **y x → r** [h]

Remainder of a division. Works for real numbers as well. Compare MOD.

RM? **- → r** [P.FCN]

Floating point rounding mode. See RM for details.

ROUND **x → r** CPX [g]

Round x using the current display format. In fraction mode, round x using the current denominator.

ROUNDI **x → r** [X.FCN]

Round x to next integer. ½ rounds to 1.

RR n **x → r** [X.FCN]
RRC n

Rotate right/rotate right through carry. For RR, 0≤n≤63. For RRC, 0≤n≤64.

RSD n **x → r** [X.FCN]

Round x to n significant digits, taking the RM setting into account. See RM and compare RDP.

RTN **- → -** [g]

Execution: Last command in a typical routine. Pop the local data (like PopLR) and return control to the calling routine in program execution, i.e. moves the program pointer one step behind the XEQ instruction that called

the routine. If there is none, program execution halts and the program pointer is set to a beginning of current program.
Other: Reset the program pointer to 000 in RAM.

RTN+1 **- → -** **P.FCN**
Like RTN, but move the program pointer two steps after the XEQ instruction that called said routine. Halt if there is none.

R-CLR **x → x** **P.FCN**
x is in the form *sss.nn*. Clear *nn* registers starting with address *sss*. If *nn*=0, it clears the maximum available.
Example: For x=34.567, R-CLR will clear R34 through R89.
ATTENTION for *nn*=0 : For *sss*=0…99, clearing will stop at the highest allocated global numbered register. For *sss*=100…111, clearing will stop at K. For *sss*≥112, clearing will stop at the highest allocated local register.

R-COPY **x → x** **P.FCN**
x is in the form *sss.nnddd*. Copy *nn* registers starting with address *sss* to *ddd*. If *nn*=00, it will take the maximum available.
Example: For x=7.03045678, r07, r08, r09 will be copied into R45, R46, R47, respectively.
For x<0, R-COPY takes *nn* registers from backup FM, starting with register number |*sss*|. Destination is always RAM.
See R-CLR

R-SORT x → x (P.FCN)

x is in the form *sss.nn*. Sort the contents of *nn* registers starting with address *sss*. If *nn*=0, it sorts the maximum available.
Example: Assume x=49.0369, r49=1.2, r50= –3.4, and r51=0 ; then R-SORT will return r49=–3.4, r50=0, and r51=1.2.
See R-CLR

R-SWAP x → x (P.FCN)

Like R-COPY but swap the contents of source and destination registers.

R↑ ... → ... (CPX) (h)
R↓ (CPX)

Rotate the stack contents one level up or down, respectively.

s - → sy sx (g)

Sample standard deviations s_y and s_x for the data in statistics registers.

$$s_x = \sqrt{\frac{\sum x_i^2 - n\bar{x}^2}{n-1}}$$

SAVE - → - (P.FCN)

Save user program space, registers and system state to backup FM, and display *Saved*. Use LOAD... to recall your backup. Not programmable.

SB n m → r (X.FCN)

Set the n-th bit in x.

SCI n - → - (h)

Scientific display format.

SCIOVR **- → -** h SCI ENTER

Use SCI mode to display numbers that cannot be displayed in ALL or FIX. Compare ENGOVR, see RESET.

SDL n **x → r** X.FCN

SDR n **x → r** X.FCN

Shift digits left (right) by n decimal positions, equivalent to multiplying (dividing) x by 10^n. Compare SL and SR for integers.

SEED **x → x** STAT

Store a seed for a random number generator.

SENDA **- → -** X.FCN

SENDP

SENDR

SENDΣ

SENDA sends all RAM data, SENDP – the program memory, SENDR – the global general purpose registers, and SENDΣ – the summation registers, to the device connected via serial I/O. See RECV.

SEPOF **- → -** MODE

SEPON **INT:** h ./,

Toggle the digit group separators for integers. Display separators every …

… four digits in bases 2 and 4,

… two digits in base 16,

… three digits in all other integer bases.

SERR **- → sy sx** STAT

Standard errors (i.e. the standard deviations of x̄ and ȳ) of the statistical data. See s.

$$s_{Ex} = \frac{s_x}{\sqrt{n}}$$

SERR$_w$ - → sx (STAT)

Standard error for weighted data, i.e. the standard deviation of $\bar{x}_w$. See s_W.

$$S_{Ew} = \frac{s_w}{\sqrt{\sum y_i}}$$

SETCHN - → - (MODE)
SETEUR
SETIND
SETJPN
SETUK
SETUSA

Set regional preferences.

SETDAT dc → dc (MODE)

Set the date for the real time clock (the emulator takes this information from the PC clock).

SETTIM tc → tc (MODE)

Set the time for the real time clock (the emulator takes this information from the PC clock).

SF n - → - (f)

Set the n-th user flag.

SHOW - → - (g)

Stack and registers browser.
(n)(n) – set current register address (CRA)
(▲)(▼) – increment or decrement CRA
(.) – turn to local registers

ENTER↑ or RCL – recall current register

SIGN **x → r** CPX X.FCN

1 for x>0, –1 for x<0, and 0 for x=0 or non-numeric data. Complex version returns unit vector of x+i·y in X and Y.

SIGNMT **- → -** MODE

Set sign-and-mantissa mode for integers.

SIN **θ → r** CPX f

Sine of an angle.

SINC **θ → r** CPX X.FCN

Unnormalized sinc: $\frac{\sin x}{x}$ for x ≠0 ; 1 for x=0.

SINH **x → r** CPX f HYP

Hyperbolic sine, $\sinh x = \frac{e^x - e^{-x}}{2}$

SKIP n **- → -** P.FCN

Skip n program steps forwards (0≤n≤255). So e.g. SKIP 2 skips over the next two steps, going e.g. from step 123 to step 126. If SKIP attempts to cross an END, an error is thrown.

SL n **m → r** X.FCN

Shift bits left. n≤63.

SLOW **- → -** MODE

Set the processor speed to 'slow'. This mode is automatically entered for low battery voltage. Compare FAST.

SLV lbl **x1 x2 → f(x) xn x** f

Solve the equation f(x)=0, with f(x) calculated

by the routine at label *lbl*. Two initial estimates of the root must be supplied in X and Y when calling SLV. For the rest, the user interface is as in the HP-15C. This means SLV returns root in X, the second last x-value tested in Y, and $f(x_{root})$ in Z. Also, SLV acts as a test, so the next program step will be skipped if SLV fails.
Please refer to the HP-15C Owner's Handbook (Section 13 and Appendix D) for more information about automatic root finding.

SLVQ **a b c → r x2 x1** **X.FCN**
Solve the quadratic equation

$$ax^2 + bx + c = 0$$

and test the result.
* If $r = b^2 - 4av \geq 0$, SLVQ returns $-\frac{b \pm \sqrt{r}}{2a}$ in Y and X. In a program, the step after SLVQ will be executed.
* Else, SLVQ returns the real part of the first complex root in X and its imaginary part in Y (the 2nd root is the complex conjugate of the first – see CONJ). If run directly from the keyboard, the complex indicator C is lit then – in a program, the step after SLVQ will be skipped.
In either case, SLVQ returns r in Z. Higher stack levels are kept unchanged. L will contain equation parameter c.

SMODE? **- → r** **P.FCN**
Integer sign mode :
2 (meaning 'true') for 2's complement,

1 ('true' again) for 1's complement,
0 (i.e. 'false') for unsigned, or
–1 (i.e. 'true') for sign and mantissa mode.

SPEC? **x → x** **TEST**
Test if x is 'special', i.e. infinite or non-numeric.

SR n **m → r** **X.FCN**
Shift bits right, n≤63.

sRCL s **- → r** **X.FCN**
Interpret contents of s as single precision data and recall it.

SSIZE4 **- → -** **MODE**
SSIZE8
Set the stack size to 4 or 8 levels. Register contents will remain unchanged in this operation. The same happens if stack size is modified by any other operation (e.g. by RCLM).

SSIZE? **- → r** **P.FCN**
Number of stack levels.

STO s **x → x** CPX **STO**
Save x to register s.

STOM s **- → -** **STO** **MODE**
Store mode settings in s (no need to press **h**). RCLM recalls mode data.

0…3	LCD contrast setting
4, 5	0=DENANY, 1=DENFAC, 2=DENFIX
6…19	DENMAX (14 bits for 0 … 9999)
20	0=PROFRC, 1=IMPFRC
21	1=fraction mode is on
22,23	0=ALL, 1=FIX, 2=SCI, 3=ENG

24…27	Number of decimals (4 bits for 0 … 11)
28	0=SCIOVR, 1=ENGOVR
29	0=RDX. 1=RDX,
30	1=E3OFF
31	1=SEPOFF
32	1=integer mode
33	1=LZON
34, 35	1=1COMPL, 0=2COMPL, 2=UNSIGN, 3=SIGNMT
36…39	Integer base (4 bits for 2…16 coded as 1…15)
40…45	WSIZE (6 bits for 1…64 coded as 0…63)
46	1=DBLON
47	0=24h, 1=12h
48, 49	Print mode=0…3, see 🖶MODE
50	Not used
51	0=SSIZE4, 1=SSIZE8
52, 53	1=Y.MD, 2=M.DY, 3=D.MY
54, 55	0=DEG, 1=RAD, 2=GRAD
56…58	0=LINF, 1=EXPF, 2=POWERF, 3=LOGF, 4=BESTF
59	0=FAST, 1=SLOW
60…62	Rounding mode (0…7, see RM)
63	0=JG1782, 1=JG1582

STOP - → - [R/S]

Stop program execution.

STOPW - → - [CPX][R/S] [X.FCN]

Stopwatch application based on the real time clock and following the timer of the HP-55. See also XTAL?

[R/S] – start/stop the timer

[←] – set the timer to zero without changing its status (running or stopped).

[EEX] – hide/show tens of seconds

[n][n] – set current register address (CRA)

[ENTER↑] – store H.MS timer value into current register, increment CRA.

[▲], [▼] – increment or decrement CRA

[.] – same as [ENTER↑][←]
[A] – convert timer value to H.d and add to statistics registers
[+] – same as [A][.]
[RCL] *nn* – recall r*nn* without changing status
[EXIT] – leave application. If counting, timer continues to count, indicated by small '=' annunciator.

STOS s **- → -** [P.FCN]
Store all current stack levels in a set of 4 or 8 registers, starting at destination address s. See RCLS.

STO+ s **x → x** [CPX]
STO– s [CPX]
STO× s [CPX]
STO/ s [CPX]
STO↑ s [STO] [▲]
STO↓ s [STO] [▼]
Execute the specified operation on s and store the result there. ↑ is maximum, ↓ is minimum. E.g. STO−12 subtracts x from r12 like the keystrokes [RCL]12[x⇄y][−][STO]12 would do, but the stack remains unchanged.

SUM **- → Σy Σx** [STAT]
Recall the linear sums Σy and Σx. Also useful for elementary vector algebra in 2D.

s_w **- → r** [STAT]
Standard deviation for weighted data (where the weight y of each data point x was entered via Σ+). See $\bar{x}_w$, compare $SERR_w$

$$s_w = \sqrt{\frac{\sum y_i \sum y_i x_i^2 - [\sum y_i x_i]^2}{\sum (y_i - 1) \sum y_i}}$$

s_{XY} **- → r** **STAT**

Sample covariance for the two data sets entered via Σ+. It depends on the fit model. For linear fit

$$s_{XY} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n(n-1)}$$

See COV for the population covariance.

TAN **θ → r** CPX **f**

Tangent of an angle.

TANH **x → r** CPX **f** **HYP**

Hyperbolic tangent of x. $\tanh x = \frac{e^{2x}-1}{e^{2x}+1}$

TICKS **- → r** **P.FCN**

Number of ticks from the real time clock. With the crystal oscillator installed, 1 tick is 0.1 s. Without, it may be some 10% more or less. TICKS does not require crystal.

TIME **- → tc** **X.FCN**

Time from the real time clock at in the format *hh.mmss*. See XTAL?

T_n **n x → r** **X.FCN**

Chebychev polynomials of first kind.

$$(1 - x^2) f''(s) - x f'(x) + n^2 f(x) = 0$$

TOP? **- → -** **TEST**

Tests false if called in a subroutine, true if the program-running flag is set and the subroutine return stack pointer is clear.

TRANSP **mat → r** MATRIX

Take a matrix descriptor x and return the descriptor of its transpose. The transpose is done in-situ and does not require any additional memory.

t_P(x) **x → r** PROB
t_u(x) **x → p**
t(x) **x → p**
t^{-1}(p) **p → x**

Student's t distribution. The degrees of freedom are stored in J.

t⇄ s **… → …** P.FCN

Swap T and s. See x⇄.

ULP **x → r** X.FCN

1 times the smallest power of ten which can be added to x or subtracted from x to actually change the value of x in the mode set. 1 in integer mode.

U_n **n x → r** X.FCN

Chebychev polynomials of second kind with n in Y. They are solutions to

$$(1-x^2)f''(s)-3xf'(x)+n(n+2)f(x)=0$$

UNSIGN **- → -** MODE

Set unsigned integer mode.

VERS **- → -** X.FCN

Show firmware version and build number.

VIEW s - → - (h)

Show s until a key is pressed.

VIEWα - → - (P.FCN)

Input: (VIEW)(−)

Display alpha in the top row and - - - in the bottom row until next key is pressed.

VWα+ s - → - (P.FCN)

Input: (h)(VIEW)

Display alpha in the top row and s in the bottom row until the next key is pressed.

WHO - → - (X.FCN)

Display credits to the brave men who made this project work.

WDAY dc → r (X.FCN)

Day number of a date. Show day name in the dot matrix. (1=Monday, 7=Sunday).

W_m x → r (X.FCN)

W_p x → r CPX (X.FCN)

W^{-1} x → r CPX (X.FCN)

W_p returns the principal branch of Lambert's W (solution of $x = We^W$) for given x≥−1/e. W_m returns its negative branch.
W^{-1} returns xe^x for x≥−1.

Weibl x → p (PROB)

$Weibl_P$ x → r

$Weibl_u$ x → p

$Weibl^{-1}$ p → x

Weibull distribution with its shape parameter b in J and its characteristic lifetime T in K.

WSIZE n **- → -** **MODE**

Set word size in integer mode. Reducing the word size truncates the values in the stack registers and in L. WSIZE 0 sets the word size to maximum, 64 bits.

WSIZE? **- → r** **P.FCN**

Current word size.

x^2 **x → r** CPX **g**

x^3 **x → r** CPX **X.FCN**

Square and cube.

XEQ lbl **- → -**

Call the subroutine with the label specified.

XEQα **- → -** **P.FCN**

Take the first three characters of alpha as a label and execute the respective routine.

XNOR **y x → r** **X.FCN**

1 when both inputs are equal. See AND.

XOR **y x → r** **h**

1 when both inputs are different. See AND.

XTAL? **- → -** **TEST**

Test for presence of the crystal necessary for a precise real time clock, DATE, TIME and printing commands.

$\bar{x}$ **- → ry rx** **f**

Arithmetic means of the x- and y- accumulated data. See also s, SERR, and σ.

$\bar{x}_g$ **- → ry rx** **STAT**

Geometric means of the accumulated data.

$\bar{x}_g = \sqrt[n]{\prod x_i}$. See also ε, εm, and εP

x̄w **- → r** **STAT**

Arithmetic mean for weighted data (where the weight y of each data point x was entered via Σ+) $\bar{x}_w = \frac{\sum x_i y_i}{\sum y_i}$. See also sw and SERRw.

x̂ **x → r** **X.FCN**

Forecast x for a given y (in X) following the fit model chosen. See L.R. for more.

x! **x → r** CPX **h**

DECM: Γ(x+1), INT: x!

x→α **x → x** **X.FCN**

Append the character with code x to alpha.

x⇄ s **x → r** CPX **h**

Swap X and s, similar to x⇄y.

x⇄ Y **y x → x y** CPX **x⇄y**

Swap the stack contents x and y. Complex swap displays as ᶜx⇄ Z.

x<? s	**x → x**	**TEST**
x≤? s	**x → x**	**TEST**
x=? s	**x → x**	CPX **f**
x=+0?	**x → x**	**TEST**
x=-0?	**x → x**	**TEST**
x≈? s	**x → x**	**TEST**
x≠? s	**x → x**	CPX **g**
x≥? s	**x → x**	**TEST**
x>? s	**x → x**	**TEST**

Compare x with s.

x≈? is true if the rounded values of x and s are equal (see ROUND).
The signed tests x=+0? and x=−0? are meant for integer modes 1COMPL and SIGNMT, and for DECM if flag D is set. In all these cases, e.g. 0 divided by –7 will display –0.
CPX f x=? s and CPX g x≠? s compare the complex number x+i·y.

ˣ√y **y x → r** CPX X.FCN
x-th root of y

YEAR **dc → r** X.FCN
Year of a date.

y^x **y x → r** CPX f
In integer modes, x must be ≥0.

ŷ **x → r** f
Forecast y for a given x following the fit model chosen. See L.R. for more.

Y.MD **- → -** MODE
Set the *yyyy.mmdd* date format.

y⇄ s **… → …** P.FCN
z⇄ s
Swap Y or Z with s. See x⇄ and t⇄.

α **- → -** f
STO&¬INPUT: Turn on alpha mode for keyboard entry of alpha constants. INPUT is set and the previous program step stays displayed until a character is entered. Each such character (e.g. '?') is stored in one program step (such as α ? here) and will be appended

to alpha during program execution.
STO&INPUT: Turn on alpha group mode for direct entry of up to three characters in one program step taking two words. Your WP 34S will display `α'` in the top line. Enter the characters you want to append to alpha.
Example: [f] [α] T [f] [↓] E S [f] [α] T [h] [PSE] 1 will result in two program steps stored:

```
a'Tes'
a't 1'
```

and *Test 1* appended to alpha during program execution.
¬STO&¬INPUT: Enter alpha mode for appending characters to alpha. To start a new string, use `CLα` first.
¬STO&INPUT: Leave alpha mode.

αDATE **dc → dc** [X.FCN]
Append formatted date to alpha. See `DATE`. To append a date stamp to alpha, call `DATE` `αDATE`. For a short European date stamp, set `FIX 2`, `RDX.` and call `DATE` `αRC# X`.

αDAY **dc → dc** [X.FCN]
Append first three letters of day of week for date in x to alpha.

αGTO s **- → -** [P.FCN]
Take the first three characters of s, interpreted as string, and position the program pointer to the alpha label with same name.

αIP **x → x** [X.FCN]
Append the integer part of x to alpha.

αLENG **- → r** **X.FCN**

Number of characters in alpha.

αMONTH **dc → dc** **X.FCN**

Append first three letters of month name for date in x to alpha.

αOFF **- → -** **P.FCN**

αON

Switch alpha mode off and on.

αRCL s **- → -** **X.FCN**

Input: f RCL

Interpret contents of s as string and append it to alpha.

αRC# s **x → x** **X.FCN**

Interpret s as a number, convert it to a string in the format set, and append this to alpha.
Example: If s is 1234 and `ENG 2` and `RDX.` are set, then 1.23e3 will be appended.
See also `αDATE` for an application.

αRL n **- → -** **X.FCN**

Rotate alpha left by n characters.

αRR n **- → -** **X.FCN**

Like `αRL` but rotates to the right.

αSL n **- → -** **X.FCN**

Shift the n leftmost characters out of alpha.

αSR n **- → -** **X.FCN**

Insert n spaces in the beginning of alpha.

αSTO s **- → -** **X.FCN**

INPUT: f STO

Store the first (leftmost) 8 characters of alpha in the destination s.

αTIME **tc → tc** **X.FCN**

Append formatted time to alpha. See 12h, 24h, and TIME. To append a time stamp to alpha, call TIME αTIME.

αXEQ s **- → -** **P.FCN**

Execute routine with alpha label equal to first 3 characters of s interpreted as string.

α→x **- → r** **X.FCN**

Remove first (leftmost) character from alpha and return its code.

β **y x → r** CPX **X.FCN**

Euler's Beta for Re(x)>0, Re(y)>0.

$$\mathrm{B}(x,y) = \frac{\Gamma(x)\,\Gamma(y)}{\Gamma(x+y)}$$

Named β to avoid ambiguity.

Γ **x → r** CPX **X.FCN**

Γ(x). Additionally, h x! calls Γ(x+1). See also LNΓ.

γ_{XY} **y x → r** **X.FCN**

Γ_{XY}

Lower or upper incomplete gamma function.

$$\gamma(x,y) = \int_0^y t^{x-1}e^{-t}dt$$

$$\Gamma_{\mathrm{u}}(x,y) = \int_y^\infty t^{x-1}e^{-t}dt$$

ΔDAYS **dc1 dc2 → r** X.FCN

Number of days between 2 dates.

Δ% **y x → y r** X.FCN

$100 \frac{x-y}{y}$. Preserves Y.

ε **- → ry rx** STAT

Scattering factors ε_y and ε_x for log-normally distributed sample data. ε_x is to the geometric mean x_g as the standard deviation s to the arithmetic mean x̄ but multiplicative instead of additive.

$$\ln \varepsilon_x = \sqrt{\frac{\sum \ln^2 x_i - 2n \ln \overline{x_g}}{n-1}}$$

ε_m **- → ry rx** STAT

Like ε but returns the scattering factors of the two geometric means. $\varepsilon_m = \varepsilon^{\frac{1}{\sqrt{n}}}$

ε_P **- → ry rx** STAT

Like ε but returns the scattering factors of the two populations.

ζ **x → r** X.FCN

Riemann's Zeta. Analytical continuation of $\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x}$

π **- → π** CPX h

Recall π.

∏ lbl **x → r** f

Compute a product using the routine lbl. Initially, X contains the loop control number in the format *cccccc.fffii*, and the product is set to 1. Each run through the routine specified by lbl computes a factor. At its end, this factor is multiplied with the product; the operation then decrements *cccccc* by *ii* and runs said routine again if then *cccccc=fff*, else returns the resulting product in X.

σ **- → ry rx** **STAT**

Standard deviations of the two populations. See also `s`

$$\sigma_x = \frac{1}{n}\sqrt{\sum(x_i - \bar{x})}$$

`Σ lbl` **x → r** **g**

Compute a sum using the routine specified at `LBL`. Initially, X contains the loop control number in the format *cccccc.fffii*, and the sum is set to 0. Each run through the routine specified by `lbl` computes a summand. Then, this summand is added to the sum; the operation then decrements *cccccc* by *ii* and repeats until *cccccc≤fff*.

`Σln²x` **- → r** **SUMS**
`Σln²y`
`Σlnx`
`Σlnxy`
`Σlny`
`Σxlny`
`Σylnx`

Recall the respective statistical sums. These sums are necessary for curve fitting models beyond pure linear. These sums are stored in special registers.

ATTENTION: Depending on input data, some or all of these sums may become non-numeric.

σ_w **- → r** **STAT**

Like s_w but returns the standard deviation of the population instead.

$$\sigma_w = \sqrt{\frac{\sum y_i (x_i - \bar{x}_w)^2}{\sum y_i}}$$

Σx **- → r** **SUMS**

Σx²

Σx²y

Σxy

Σy

Σy²

Recall the respective statistical sums. These sums are necessary for basic statistics and linear curve fitting. These sums are stored in special registers.

Σ+ **y x → y n** **h** **A**

Σ– **y x → y n** **h**

Σ+ adds a data point to the statistical sums. Shortcut works if label A is not defined.

Σ- subtracts a data point from the statistical sums.

Both functions preserve Y, return number of points in X, disable stack lift.

Both may be used for 2D vector adding and subtracting as well.

Φ_u(x) **x → p** (PROB)

Standard normal error probability cdf.

φ(x) **x → r** (PROB)

Standard normal pdf. $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

Φ(x) **x → p** (f)

Standard normal cdf. $\Phi(x) = \int_{-\infty}^{x} \varphi(t) dt$

Φ^{-1}(p) **p → x** (g)

Inverse of standard normal cdf.

χ^2 **x → p** (PROB)
χ^2INV **p → x**
χ^2_P **x → r**
χ^2_u **x → p**

χ^2 distribution, degrees of freedom in J.

$(-1)^X$ **x → r** (CPX) (X.FCN)

cos(π·x) for non-integers.

×MOD **z y x → r** (X.FCN)

(z·y)mod x for x>1, y>0, z>0.

/ **y x → r** (CPX) (÷)

y/x. Compare IDIV.

+/- **x → r** (CPX)

Unary minus, corresponding to x·(−1) or x^c·(−1), respectively.

→DATE **y m d → dc** (X.FCN)

Convert three components of a date (year, month, and day) to a date according to date format. Inverse of DATE→.

→DEG **θ → r** [→][DEG]

Convert x in current angular units to degrees. Prefix [g] may be omitted.

→GRAD **θ → r** [→][RAD]

Like →DEG, but converts to gon (grad).

→HR **x → r** [→][f][H.d]

Convert hours or degrees in the format *hhhh.mmssdd* to a decimal time or angle, allowing for using standard arithmetic operations then.

→H.MS **x → r** [→][f]

Convert x as decimal hours or degrees to the format *hhhh.mmssdd*. See H.MS+, H.MS−.

→POL **y x → θ r** [g]

Assume X and Y contain 2D Cartesian coordinates (x, y) of a point or components of a vector and convert them to the polar coordinates/components (r, θ)

→RAD **θ → r** [→][RAD]

Like →DEG, but converts to radians.

→REC **θ r → y x** [f]

Assumes X and Y containing 2D polar coordinates (r, θ) of a point or components of a vector and converts them to the Cartesian coordinates or components (x, y).

⇄???? **… → …** [P.FCN]

Shuffle the contents of the bottom four stack levels. Examples:

⇄XXYZ works like ENTER↑ (but does not disable stack lift),
⇄YZTX works like R↓,
⇄ZTXY works like ᶜx⇄y,
but ⇄ZZZX is possible as well.
This command does not affect the higher levels in an 8-level stack.

% **y x → y r** (f)
$^{xy}/_{100}$, keeps Y. Disables stack lift.

%MG **y x → r** (X.FCN)
Margin $100\frac{x-y}{x}$ in % for a price x and cost y.

%MRR **z y x → r** (X.FCN)
Mean rate of return in percent per period, i.e. $100\left((x/y)^{1/z}-1\right)$ with x= future value after z periods, y= present value.
For z=1, **Δ%** returns the same result easier.

%T **y x → y r** (X.FCN)
$100\,^{x}/_{y}$, interpreted as % of total. Keeps Y.

%Σ **x → r** (X.FCN) (STAT)

$100\frac{x}{\Sigma x}$

%+MG **y x → r** (X.FCN)
Calculate a sales price by adding a margin of x% to the cost y. $r=\frac{y}{1-x/100}$
You may use **%+MG** for calculating net amounts as well. Just enter a negative percentage in x.

Example: Total billed =221,82 €, VAT=19%. What is the net?
221,82 ENTER↑ 19 +/− X.FCN %+MG returns 186,40.

√ **x → r** CPX f

Square root.

∫ lbl **y x → r** g

Integrate the function given in the routine specified. Lower and upper integration limits must be supplied in Y and X, respectively. Otherwise, the user interface is as in the HP-15C.
Please turn to the HP-15C Owner's Handbook (Section 14 and Appendix E) for more information about automatic integration and some caveats.

∞? **x → x** TEST

Test x for infinity.

^MOD **z y x → r** X.FCN

(z^y) mod x for x>1, y>0, z>0.
Example:
f 10 73 ENTER↑ 55 ENTER↑ 31 X.FCN ^MOD returns 26.

|| **y x → r** CPX g

$\frac{1}{1/x+1/y}$, or 0 if x or y is zero.

🖶ADV **- → -** P.FCN

Print the current contents of the print buffer plus a linefeed. The printer will actually print only when a line feed is sent to it.

ATTENTION: Any printing command works only with a hardware modification or in emulator in combination with a printer emulator. Otherwise, print commands will be ignored. See 🖶? and XTAL?.

🖶CHR n - → - (P.FCN)

Send a single character (with the code *n*) to the printer. Character codes *n*>127 can only be specified indirectly. Honor 🖶MODE setting. Compare 🖶#. See 🖶ADV.

🖶PLOT s - → - (P.FCN)

Send the graphic block starting at address s to the printer. If its width is 166, the data will be trailed by a line feed. See 🖶ADV and gDIM.

🖶$^c r_{XY}$ s - → - (P.FCN)

Print the registers s and s+1. A semicolon separates both components in the output. Works like 🖶r otherwise.

🖶DLAY n - → - (P.FCN)

Set a delay of n ticks (see TICKS) to be used with each line feed on the printer.

🖶MODE n - → - (P.FCN)

Set print mode.
0 (default): Use the printer font and character set wherever possible. All characters feature the same width, 5 pixels + 2 pixels.
1: Use the variable pitch display font.
2: Use the small display font.
3: Send the output to the serial channel. Works for plain ASCII only – no characters

will be translated. Line setup is the same as for serial communication: 9600 baud, 8 bits, no parity.

🖶PROG - → - (P.FCN)

Print the listing of the current program, one step per line. See 🖶ADV. Not programmable.

🖶r s - → - (P.FCN)

Prints s, right adjusted, without label. Shortcut (f)(↑) in run mode prints X. See 🖶ADV.

🖶REGS x →

x (P.FCN)

Interpret x in the form *sss.nn*. Print the contents of *nn* registers starting with number *sss*. Each register takes one line starting with a label.

ATTENTION: for *nn*=0:

For s=0…99, printing stops at the highest allocated global numbered register.

For s=100…111, printing stops at K.

For s≥112, printing stops at the highest allocated local register.

See also 🖶ADV.

🖶STK - → - (P.FCN)

Print the stack contents. Each level prints in a separate line starting with a label. See 🖶ADV.

🖶TAB n - → - (P.FCN)

Position the printer head to print column *n* (0 to 165, *n*>127 can only be specified indirectly). If n is less than current position, output linefeed to reach the new position. See 🖶ADV.

🖶WIDTH - → r (P.FCN)

Number of print columns alpha would take in the print mode set. See 🖶ADV and 🖶MODE. Second use: in 🖶MODE 1 or 2, 🖶WIDTH returns the width of alpha in pixels (including the last column being always blank) in the specified font.

🖶α - → - (P.FCN)

Append alpha to the print line, trailed by a line feed. Compare 🖶α+ and 🖶+α. See 🖶ADV.

🖶α+ - → - (P.FCN)

Send alpha to the printer without a trailing line feed, allowing further information to be appended to this line. May be repeated. See also 🖶ADV, 🖶r and 🖶+α.

🖶Σ - → - (P.FCN)

Print the summation registers. Each register prints in one line starting with a label. See 🖶ADV.

🖶+α - → - (P.FCN)

Append alpha to the print line, adjusted to the right and trailed by a line feed. Compare 🖶α and 🖶α+. See 🖶ADV.

🖶? - → - (P.FCN)

Test if the crystal and the necessary firmware are installed for printing.

🖶# n - → - (P.FCN)

Send a single byte, without translation, to the printer (e.g. a control code). *n*>127 can only be specified indirectly. Do not honor 🖶MODE. Compare 🖶CHR. See 🖶ADV.

 # n - → r (CPX) (CONST)
(CPX) n

Insert integer constant 0≤n≤255 in a single step. ᶜ# works like # but also clears y. The shortcut works only for 1≤n≤9.

User flags

T – tracing
A – large "=" annunciator
B – 'big'; overflow in integer modes
C – carry; used in integer operations
D – 'danger'; allow infinite or non-numeric results without error

(ON) combinations

(ON)+(+), (ON)+(−) increase/decrease LCD contrast.
(ON)+(STO)+(STO) – create a copy of the RAM in BUP, like SAVE.
(ON)+(RCL)+(RCL) – restore RAM from BUP, like LOAD.
(ON)+(C) tell the system that crystal oscillator is installed. (Keep holding (ON) and press (C) second time to confirm)
(ON)+(D) toggle debugging mode
(ON)+(S) keep holding (ON) and press (S) second time to clear GPNVM1 bit and turn calculator off. Works only in debugging mode.

WARNING: this clears the entire firmware and brings calculator in SAM-BA boot mode. You will need a SAM-BA software and communication cable to restore it to operational state.

Copyright notice

The latest version of this document can be found at http://nsg.upor.net/wp34spr