OWNER'S MANUAL

TABLE OF CONTENTS

WELCOME

Dear user, now you have got your own WP 34S. It uses the mechanics and hardware of the *HP-20b Business Consultant* or the new *HP-30b Business Professional*, so you get their unexcelled processor speed and with the *HP-30b* also the famous rotate-and-click keys giving the tactile feedback appreciated in vintage *Hewlett-Packard* calculators for decades. On the other hand, the firmware and the user interface of the WP 34S are carefully new designed and written from scratch to give you a **fast and compact <u>scientific</u> calculator like you have never had before**.

Its function set is based on the one of the renowned *HP-42S RPN Scientific*, the most powerful programmable RPN calculator built so far [1]. We expanded this set, incorporating the functionality of the famous programmer's calculator *HP-16C*, the fraction mode of the *HP-32SII*, probability distributions as featured by the *HP-21S*, and added **many more useful functions for mathematics, statistics, physics, engineering, programming etc.** like

- **+** Euler's Beta function, Fibonacci number calculation, Lambert's W (all of these in the real and complex domains), the error function, incomplete regularized Beta and Gamma, Riemann's Zeta, the most 'popular' orthogonal polynomials, testing for primality,

- **+** many statistical distributions and their inverses like Poisson, Binomial, Geometric as well as Cauchy-Lorentz, Exponential, Logistic, Weibull for reliability analysis, Lognormal and Gaussian with arbitrary means and standard deviations,

- **+** programmable sums and products, first and second derivatives,

- **+** extended date and time calculations based on a real time clock,

- **+** integer computing in arbitrary bases from binary to hexadecimal,

- **+** financial operations like mean rate of return and margin calculations,

- **+** over 70 conversions, mainly between universal SI and old Imperial units,

- **+** nearly 50 fundamental physical constants as precise as known today by national standards institutes like NIST or PTB,

- **+** complete Greek and extended Latin letter fonts covering many languages (upper and lower case in two font sizes each).

The WP 34S is the first RPN calculator overcoming the limits of a 4-level stack – forget worries about stack overflow in calculations. It features a choice of two stack sizes expanded by a complex LASTx register: traditional 4 stack levels for HP compatibility, 8 levels for convenient calculations in complex domain, for more advanced real formulas, or for whatever application you have in your mind. The WP 34S features a full command set for navigation in either size.

Furthermore, the WP 34S features over 100 general purpose registers, 100 user flags, 506 program steps, 4 programmable hotkeys for your favorite functions or routines, and a 31 byte alpha register for message generation.

If you know how to deal with a good old HP RPN scientific calculator, you can start with your WP 34S right away. Else get an *HP-42S Owner's Manual*, e.g. on the DVDs

---

[1] Though the *HP-42S* was sold in 1988 already, this statement holds still. – Due to display restrictions, the *HP-42S'* matrix math cannot be supported by the WP 34S. Sorry for this.

distributed by the *Museum of Hewlett-Packard Calculators* (*www.hpmuseum.org*). Then please read Part 1 of it as a starter, including an excellent introduction to RPN. Part 2 will become beneficial when you are heading for programming your WP 34S. Further documentation, also about the other calculators mentioned, will add valuable information – it is all readily accessible on a single DVD from said source.

This little manual here is meant as a supplement showing you all the new features of the WP 34S. It starts presenting its keyboard as it will be active in various modes, so you know where to find what you are looking for. It continues explaining the memory and addressing items therein, and the display and indicators used to tell you what's going on. Then the major part of this booklet is taken by the index of all operations, catalog contents, constants and conversions featured. It closes with a list of messages the WP 34S will display if special input conditions prevent it from executing your command as expected.

Your WP 34S is the result of a long range collaboration of two individuals, an Australian and a German. We did this in our free time, so you may call it our hobby (though some people close to us found different names for this). Our project was discussed on the open forum in the Museum of HP Calculators from its beginning, so we want to express our gratitude to all contributors there who taught us a lot and brought their ideas and support in several stages of our project. Special thanks go to Marcus von Cube supporting us in bringing it to life, starting with an emulator for v1.14 allowing widespread use and easy testing. From v1.17 on, it is running on the real hardware.

We baptized it WP 34S in honor of one of the most powerful LED pocket calculators, the *HP-34C*, and since it is our humble approach – with the hardware given – to a future 43S we can only dream of becoming the successor of the *HP-42S*. Maybe it will help convincing those having access to more resources than us that it is worthwhile covering the market of serious scientific instruments.

Firmware-wise, we have carefully checked everything we could think of to our best knowledge, so our hope may be justified the WP 34S is bug-free. We cannot guarantee this, however, nor can we bear any liability for errors in calculations nor their possible consequences. Nevertheless, we promise we will improve the WP 34S whenever it turns out being necessary – so if you discover any strange result, please report it to us, and if it is revealed to be an internal error we will provide you with an update as soon as we have one ourselves.

Enjoy!

Paul Dale and Walter Bonin

PRINT CONVENTIONS

Throughout this manual, calculator commands are generally called by their names, usually written in CAPITALS. Each and every command featured is listed in the Index of Operations. This $\boxed{\text{CPX}}$ font is taken for explicit references to keys.

Register addresses are printed using Times New Roman. Lower case italic letters of this font are employed for register contents (so e.g. y lives in stack level Y, $r45$ in general purpose register R45, or $alpha$ in the alpha register, respectively). Lower case bold *italic* Arial letters like $\boldsymbol{n}$ are used for variables.

All this holds unless stated otherwise explicitly.

KEYBOARD

Σ+ A ALL A HYP ⁻¹	1/x B FIX B SIN ⁻¹	yˣ C SCI C COS ⁻¹	√x D ENG D TAN ⁻¹	→ CONV E R◂▸P	CPX MODE F aᵇ/c ᵈ/c

STO CAT G H.MS DEG	RCL VIEW H H .d RAD	R↓ R↑ I RAN# GRAD	f	g	h

ENTER↑ CONST α		x◂▸y x◂▸ J x◂▸α	+/− NOT K 2 8	E CLP L 10 16	← CLx CLαΣ
	FILL				

XEQ GTO eˣ LN	7 AND M 10ˣ LG	8 OR N 2ˣ LB	9 XOR O yˣ LOGₓ	/ RMDR P 1/x //

▲ ! C P y,x	4 PROB Q Φ ⁻¹	5 STAT R x̄ s	6 L.R. S ŷ r	× STATUS T √x x²

▼ X.FCN ◂( )▸	1 TEST x = ≠ ?	2 P.FCN U SLV ∫	3 π V Π Σ	− Σ − W Δ %

EXIT SHOW ⬆ ONOFF	0 PSE IxI RND	· · /, X IP FP	R/S P/R Y LBL RTN	+ Σ + Z DSE ISG

a ᵇ/c

Generally speaking, white labels execute the *default primary function* of the respective key. To access a golden, blue, or green label, use *prefix* 🟧 f, 🟦 g, or 🟩 h, respectively. Any label underlined opens a *catalog*. For example, ⑤ preceded by

- **f** calculates the mean values of the data in the statistic registers via $\boxed{\overline{x}}$,

- **g** returns the standard deviations for the same data via $\boxed{s}$,

- **h** opens a catalog of supplementary statistic functions via $\boxed{\text{STAT}}$.

- The dark red letter **R** will become relevant in *alpha mode*.

You may keep the respective prefix pressed if you want to call several functions showing the same label color in sequence.

Please note that numeric entry fills the command line and is interpreted when completed, not earlier.

Further remarks (all meant as appetizers – find more about these topics in dedicated texts below):

- The *hotkeys* $\boxed{A}$, $\boxed{B}$, $\boxed{C}$, and $\boxed{D}$ immediately call the user programs carrying these labels if defined, else they act as $\boxed{\Sigma+}$, $\boxed{1/x}$, $\boxed{y^x}$, or $\boxed{\overline{x}}$, respectively.

- $\boxed{\rightarrow}$ trailed by $\boxed{\text{H.MS}}$, $\boxed{\text{H.d}}$, $\boxed{\text{DEG}}$, $\boxed{\text{RAD}}$, $\boxed{\text{GRAD}}$, $\boxed{2}$, $\boxed{8}$, $\boxed{10}$, or $\boxed{16}$, converts x , i.e. the value currently displayed.

- If $\boxed{.}$ is used twice in numeric input, the WP 34S enters *fraction mode*.

- $\boxed{\text{CPX}}$ is employed as a prefix for calling functions in complex domain.

Please see the following text and especially the *index of operations* for a complete list and the necessary explanations of all the commands provided.
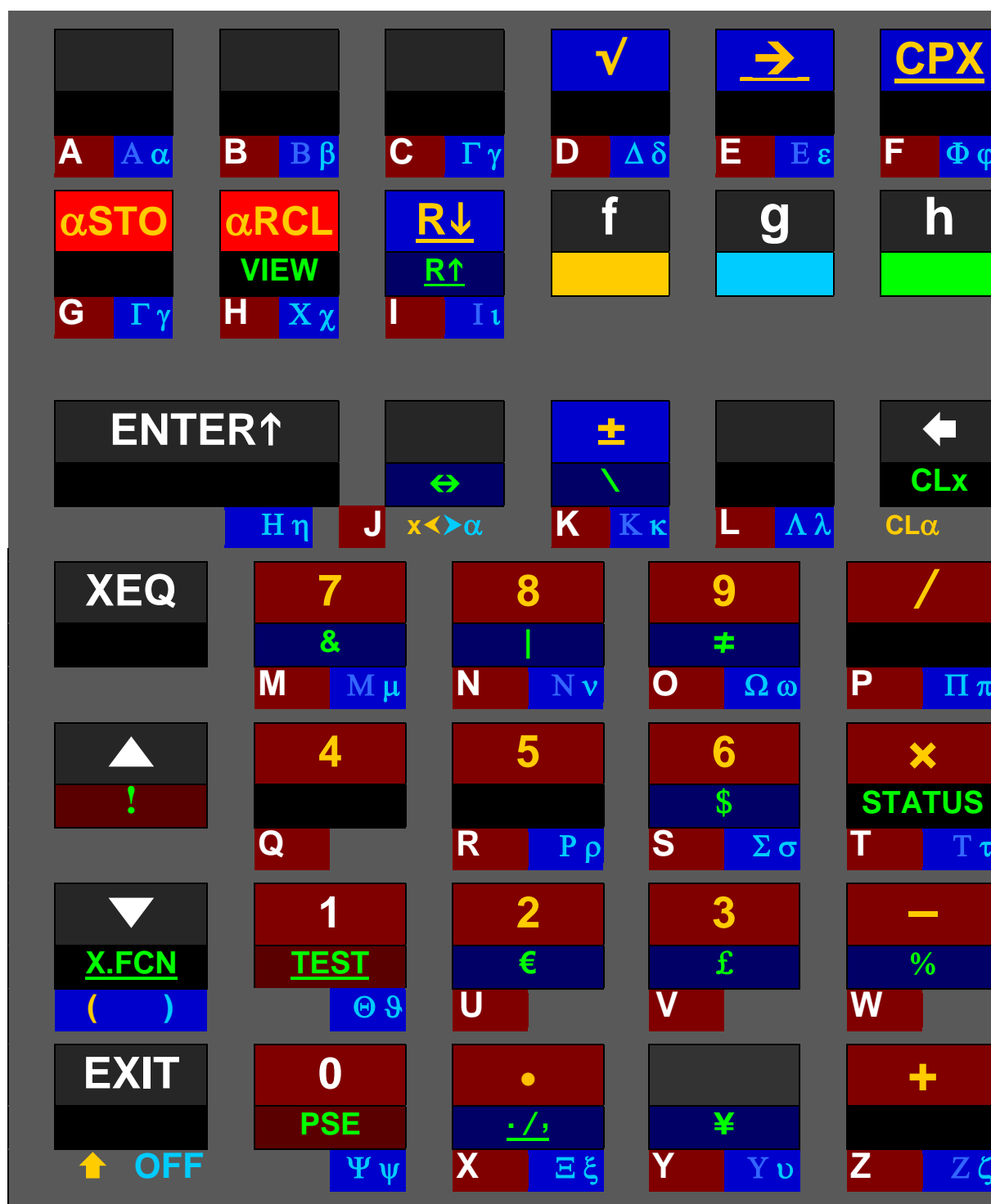
Virtual active keyboard in **hexadecimal** mode:

y^x $\sqrt{x}$

A	B	C	D	→

MODE

A   B   C   D   E   F

STO	RCL	R↓	f	g	h
CAT	VIEW	R↑			

.d   RAN#

ENTER↑	x‹›y	+/−		←
	x‹›	NOT	CLP	CLx

α   FILL   x‹›α   2   8   10   16   CLα

XEQ	7	8	9	/
GTO	AND	OR	XOR	RMDR

2^x LB y^x

▲	4	5	6	×
				STATUS

$\sqrt{x}$ x^2

▼	1	2	3	−
X.FCN	TEST	P.FCN		

‹   ›   x = ≠   ?

EXIT	0		R/S	+
	PSE		P/R	

OFF |x| LBL RTN

Primary functions of the top six keys will be numeric input, so their *default* primary functions are accessed using f . The key → is exclusively for addressing and temporary display in other bases (see *addressing tables* and *index of operations* below).

For smaller integer bases than 16, the active keyboard will look similar, but those top keys not needed for numeric input will keep their default primary functions, except Σ+ and CPX . Attempts to enter an illegal digit will be blocked.

Virtual active keyboard in **alpha** mode:

| √ | → | CPX |

| A | A α | B | B β | C | Γ γ | D | Δ δ | E | E ε | F | Φ φ |

| αSTO | αRCL VIEW | R↓ R↑ | f | g | h |

| G | Γ γ | H | X χ | I | I ι |

| ENTER↑ | | ± \ | | ← CLx |

| H η | J | x◄►α | K | K κ | L | Λ λ | CLα |

| XEQ | 7 & | 8 | | 9 ≠ | / |

| M | M μ | N | N ν | O | Ω ω | P | Π π |

| ▲ ! | 4 | 5 | 6 $ | ✖ STATUS |

| Q | R | P ρ | S | Σ σ | T | T τ |

| ▼ X.FCN | 1 TEST | 2 € | 3 £ | − % |

| ( ) | Θ ϑ | U | V | W |

| EXIT | 0 PSE | • ·/, | ¥ | + |

| ⬆ OFF | Ψ ψ | X | Ξ ξ | Y | Υ υ | Z | Z ζ |

In this mode, the alpha register is displayed in the dot matrix, and the numeric line is accessible by commands only. All labels printed on dark red or blue background in this picture append characters to *alpha* immediately or via alpha catalogs; those on blue deviate from the labels printed on the WP 34S keyboard at these locations.

Alpha mode starts in upper case, and ⬆ toggles upper and lower case. PSE appends a space. Primary function of most keys is appending the letter printed bottom left of this key – dark red on the key plate. Prefix f will access the default primary

functions there [2]. **g** leads to homonymic Greek letters where applicable [3]. And **h** gives access to logic symbols via the Boolean operations, and four currency symbols located next to the %-command as follows: **$** at the letter S, **€** at U for Euro, **£** at π, **¥** at Y for Yen or Yuan – and % at ⊟. The catalogs **f** →, **f** CPX, **f** R↓, **h** R↑, **h** TEST, and **h** ./, feature even more characters (see *below*). See the *index of operations* for αSTO, αRCL, and more alpha commands.

When *alpha* exceeds 31 characters, the leftmost character(s) are discarded.

A **temporary alpha** **mode** is entered during input processing in comparisons and in memory addressing, e.g. during storing. See the respective virtual keyboard here:
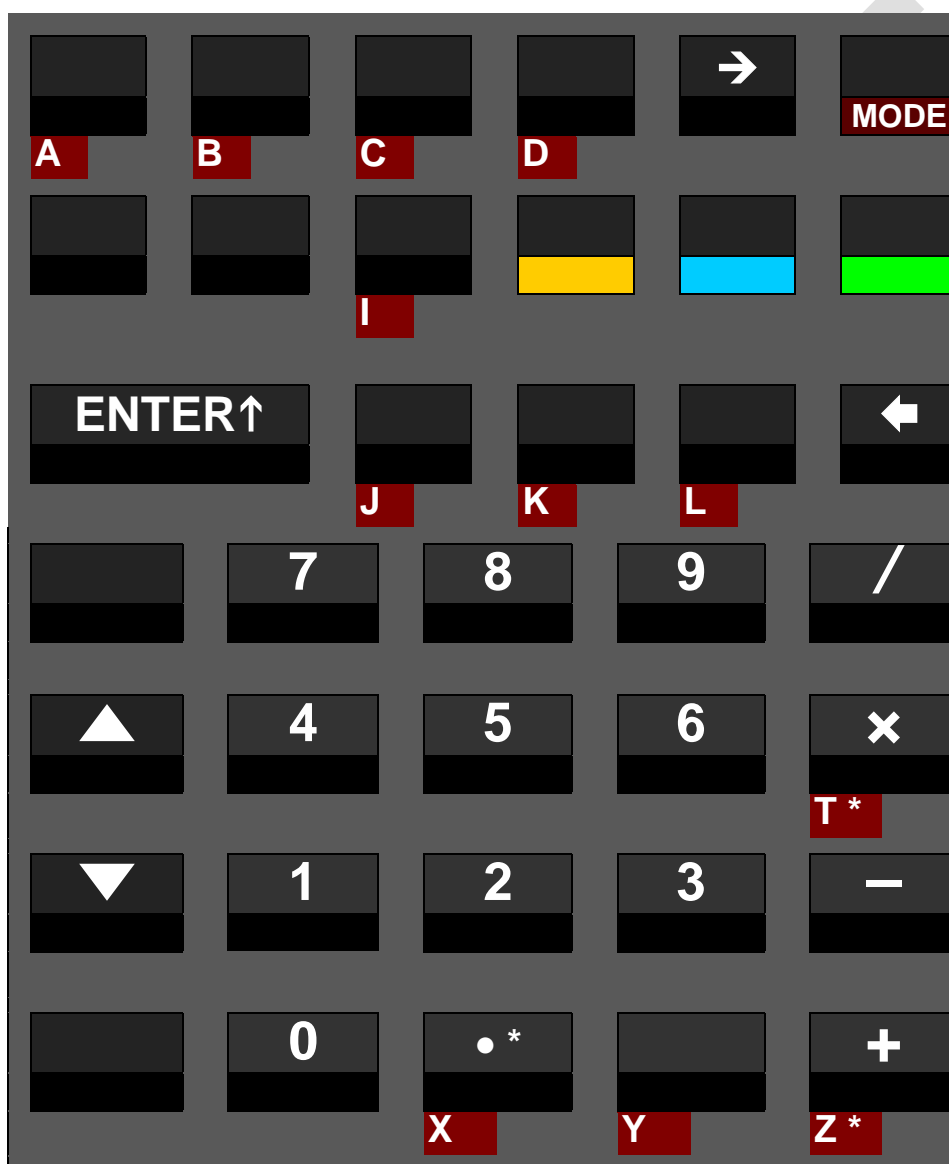
This mode is left automatically when sufficient characters are put in for the respective command. Examples are shown *below*. Special rules apply for ⊡ , T and Z – see *below*.

---

[2] The digits 0 and 1 may also be called using **f** 0 or **f** 1 , respectively.

[3] "Homonymic" according to ancient Greek pronunciation. And we assigned **Gamma** also to **C** due to the alphabet, and **Chi** to **H** since this letter comes next in pronunciation. Three Greek letters require special handling: **Psi** is accessed via **g** 0 (below PSE), **Theta** via **g** 1 (below TEST and following **T**), and **Eta** via **g** ENTER↑ . **Omicron** is not featured since looking exactly like the Latin letter **O** in either case. – Where we printed Greek capitals with lower contrast on page 7, they look like the respective Latin letters in our fonts. Greek professors, we count on your understanding.

REAL AND INTEGER OPERATIONS

Most of the commands your WP 34S provides are mathematical operations or functions in real domain. "Real domain" means these functions use and work with real numbers like 1 or 2.34 or π or 5.6E-7. Please note integer numbers like 8, 9, 10, or -1 are just a subset of real numbers.

Most real number functions provided operate on one number only. Examples are $1/_x$, $\sqrt{x}$, or SIN. Such functions replace x (i.e. the contents of the displayed stack level $\mathbf{X}$) by the result $\mathbf{f(x)}$ stored in $\mathbf{X}$ again. Everything else stays unchanged as is.

Some of the most popular mathematical functions, however, operate on two numbers. Think of + and – , for example. On the WP 34S, such a two-number real function replaces x by the result $\mathbf{f(x, y)}$, i.e. it eats up the lowest two stack levels but needs only one to put its result in. Thus, level $\mathbf{Y}$ is then filled with the content of the next higher level, i.e. z . This goes on for higher levels, as shown *below*. Please note the top stack level content is repeated then since there is nothing else available. You may use this top level repetition for some nice tricks.

There are also a few three-number real functions included – e.g. Iβ and %MRR – replacing x by the result $\mathbf{f(x, y, z)}$. Then $\mathbf{Y}$ is filled with t and so on, and the content of the top level is repeated twice. And there is SLVQ with three input and two output levels, thus treating the higher levels as two-number functions do.

Some real functions (e.g. DECOMP) operate on one number but return two. Other operations (like RCL or SUM) do not consume any stack input at all but just return one or two numbers. Then these extra number(s) will be pushed on the stack, taking one level per real number.

COMPLEX OPERATIONS

Mathematicians know more complicated items than real numbers. The next step are complex numbers. If you do not know them nor want to learn about them, leave them aside – you can use your WP 34S perfectly without them.

Else please note the WP 34S supports many operations in complex domain as well. The key $\boxed{\text{CPX}}$ is employed as a prefix for calling complex functions. E.g. $\boxed{\text{CPX}}$ $\boxed{\text{f}}$ $\boxed{\text{COS}}$ calls the complex cosine, and it is displayed and listed as ^{C}COS (the elevated C is the signature for complex functions on the WP 34S). All such functions operate on Cartesian coordinates exclusively. Generally, if an arbitrary real function $\mathbf{f}$ operates on …

- one real number x only, then its complex sibling $^C\mathbf{f}$ will operate on the complex number $x_c = x + i\,y$.

- one register, e.g. $\mathbf{R12}$, then $^C\mathbf{f}$ will operate on $\mathbf{R12}$ and $\mathbf{R13}$.

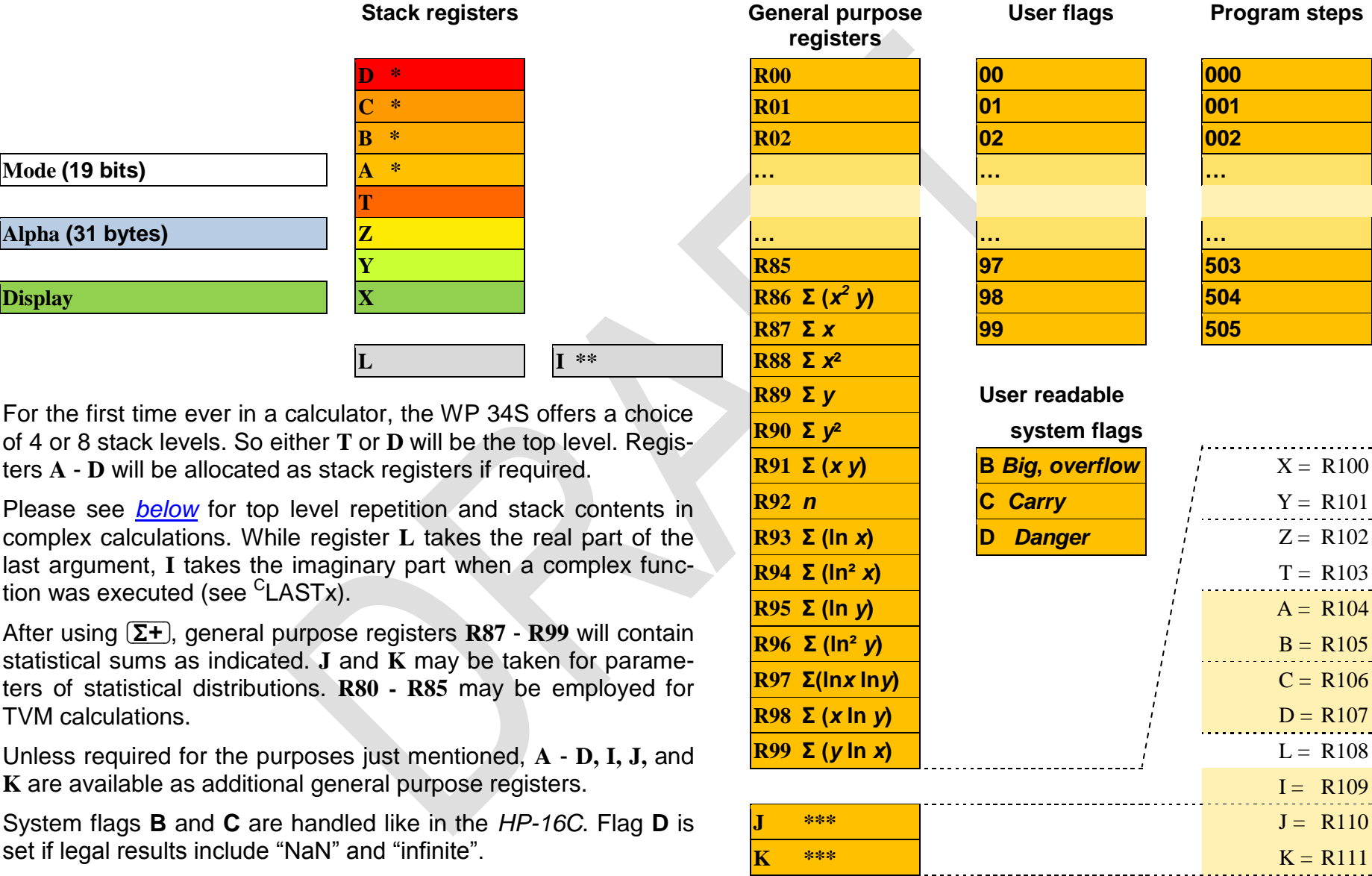- x and y, then $^C\mathbf{f}$ will operate on x, y, z and t .

Where one-number real functions replace x by the result $\mathbf{f(x)}$, one-argument complex functions replace x by the real part and y by the imaginary part of the complex result $^C\mathbf{f(x_c)}$. Higher stack levels remain unchanged. Such functions are ^{C}1/x, ^{C}ABS,

^{C}ANGLE, ^{C}CUBE, ^{C}CUBERT, ^{C}FIB, ^{C}FP, ^{C}IP, ^{C}RND, ^{C}SIGN, ^{C}W, ^{C}W^{-1}, ^{C}x!, ^{C}x^2, $^C\sqrt{}$, ^{C}**+/−**, ^{C}Γ(x), the logarithmic and exponential functions with bases 10, 2 and **e**, as well as hyperbolic, trigonometric, and their inverses.

Two-number real functions replace x by the result $f(x, y)$. Analogously, two-argument complex functions replace x by the real part and y by the imaginary part of the complex result $^Cf(x_c, y_c)$. The next stack levels are filled with the complex contents of higher levels, and the complex number contained in the top two stack levels is repeated as shown . Such complex functions are ^{C}LOG$_X$, ^{C}y^x, ^{C}β(x,y), ^{C}// , and the basic arithmetic operations in complex domain.

Where complex operations (like ^{C}RCL) do not consume any stack input at all but just return a complex number, this will be pushed on the stack taking two levels.

MEMORY

	Stack registers	General purpose registers	User flags	Program steps
	D *	**R00**	**00**	**000**
	C *	**R01**	**01**	**001**
	B *	**R02**	**02**	**002**
Mode (19 bits)	**A** *	**…**	**…**	**…**
	T			
Alpha (31 bytes)	**Z**	**…**	**…**	**…**
	Y	**R85**	**97**	**503**
Display	**X**	**R86 Σ (x^2 y)**	**98**	**504**
		R87 Σ x	**99**	**505**

L	**I** **

R88 Σ x^2
R89 Σ y
R90 Σ y^2

User readable system flags

| **B** *Big, overflow* |
| **C** *Carry* |
| **D** *Danger* |

For the first time ever in a calculator, the WP 34S offers a choice of 4 or 8 stack levels. So either **T** or **D** will be the top level. Registers **A** - **D** will be allocated as stack registers if required.

Please see *below* for top level repetition and stack contents in complex calculations. While register **L** takes the real part of the last argument, **I** takes the imaginary part when a complex function was executed (see ^{C}LASTx).

After using $\boxed{\text{Σ+}}$, general purpose registers **R87** - **R99** will contain statistical sums as indicated. **J** and **K** may be taken for parameters of statistical distributions. **R80 - R85** may be employed for TVM calculations.

Unless required for the purposes just mentioned, **A** - **D, I, J,** and **K** are available as additional general purpose registers.

System flags **B** and **C** are handled like in the *HP-16C*. Flag **D** is set if legal results include "NaN" and "infinite".

R91 Σ (x y)
R92 n
R93 Σ (ln x)
R94 Σ (ln² x)
R95 Σ (ln y)
R96 Σ (ln² y)
R97 Σ(ln$x$ ln$y$)
R98 Σ (x ln y)
R99 Σ (y ln x)

| X = R100 |
| Y = R101 |
| Z = R102 |
| T = R103 |
| A = R104 |
| B = R105 |
| C = R106 |
| D = R107 |
| L = R108 |
| I = R109 |

| **J** *** |
| **K** *** |

| J = R110 |
| K = R111 |

Furthermore, the WP34S allows users accessing flash memory for voltage-fail safe storage of user program and data. Flash memory features three segments accessible. Segment 0 is the backup region.

Flash memory is ideal for backups or other long-living data like program libraries, but shall not be used for repeated transient storage like in programmed loops (since it will not survive more than approximately 10,000 flashs). Registers and standard user program memory, residing in RAM on the opposite, are designed for frequent data changes – but will not hold data with the batteries removed. So both kinds of memory have specific advantages and disadvantages you shall take into account for optimum benefit and long lasting joy with your WP 34S.

STACK MECHANICS

The following assumes you are familiar with RPN – else please turn to the *HP-42S Owner's Manual* first.

The fate of particular stack contents depends on the operation executed, its domain (real or complex) and the stack size chosen. Real functions in a 4-level stack work as known for decades. In a larger stack, everything works alike on the WP 34S – just with more levels for intermediate results. Calculating formulas from inside out stays a wise strategy in either stack. With more levels, however, stack overflow will hardly ever happen, even with the most advanced formulas you compute in your life as a scientist or engineer.

Calculating with complex numbers uses two registers or stack levels for each such number as explained above and shown here:

	Level	Assumed stack contents at the beginning:	Stack contents after executing …				… complex functions of		… real functions of two numbers like /	
			… the complex stack register operations				… one number like $^{C}x^2$	… two numbers like $^{C}/$		
			^{C}ENTER, ^{C}FILL	^{C}DROP	^{C}x↔y, ^{C}R↓, ^{C}R↑	^{C}LASTx			Before	After
With **4** stack levels	**T**	$\mathbf{Im(y_c) = Im(t_c)}$	$Im(x_c)$	$y_c = t_c$	$Im(x_c)$ / x_c	x_c	$y_c = t_c$	$y_c = t_c$	t	t
	Z	$\mathbf{Re(y_c) = Re(t_c)}$	$Re(x_c)$		$Re(x_c)$				z	t
	Y	$\mathbf{Im(x_c)}$	$Im(x_c)$	y_c	$Im(y_c)$ / y_c	$lastx_c$	$Im((x_c)^2)$	$Im(y_c / x_c)$	y	z
	X	$\mathbf{Re(x_c)}$	$Re(x_c)$		$Re(y_c)$		$Re((x_c)^2)$	$Re(y_c / x_c)$	x	y / x

	Level	Assumed stack contents at the beginning:	^{C}ENTER, ^{C}FILL	^{C}DROP	^{C}x↔y, ^{C}R↓, ^{C}R↑		^{C}LASTx	… one number like $^{C}x^2$	… two numbers like $^{C}/$	Before	After
With **8** stack levels	**D**	$\mathbf{Im(t_c)}$	z_c / x_c	t_c	t_c / x_c	z_c	z_c	t_c	t_c	d	d
	C	$\mathbf{Re(t_c)}$								c	d
	B	$\mathbf{Im(z_c)}$	y_c / x_c	t_c	z_c / t_c	y_c	y_c	z_c	t_c	b	c
	A	$\mathbf{Re(z_c)}$								a	b
	T	$\mathbf{Im(y_c)}$	x_c / x_c	z_c	x_c / z_c	x_c	x_c	y_c	z_c	t	a
	Z	$\mathbf{Re(y_c)}$								z	t
	Y	$\mathbf{Im(x_c)}$	x_c / x_c	y_c	y_c / y_c	t_c	$lastx_c$	$(x_c)^2$	y_c / x_c	y	z
	X	$\mathbf{Re(x_c)}$								x	y / x

So, an 8-level stack gives you the same flexibility in complex domain you are used to with a 4-level stack in real domain.

COMPARING AND ADDRESSING REAL NUMBERS

1	User input	x= ?, x≠ ?, x< ?, x≤ ?, x≈ ?, x≥ ?, or x> ?				RCL, STO, RCLS, STOS, αRCL, αSTO, VIEW, x≷, DSE, ISG, DSZ, ISZ, FIX, SCI, ENG, DISP, BASE, KEY?, CB and many more bit commands, or CF and the other flag commands etc.			
	Dot matrix display	**OP _** (with temporary alpha mode set), e.g. x> _				**OP _** (with temporary alpha mode set), e.g. RCL _ [4]			
2	User input	0 or 1	Stack level or named reg. X, Y, ...	ENTER↑ [5] leaves temp. alpha mode.	→ opens indirect addressing.	Stack level or named register X, Y, Z, .. , K [6]	Number of register or flag or bit(s) or decimals [7]	→ opens indirect addressing.
	Dot matrix display	**OP n** e.g. x≤ 0 ?	**OP x** e.g. x≥ y ?	**OP r_**	**OP →_**	**OP x** e.g. SCI Z	**OP nn** e.g. SF 15	**OP →_**
3	User input	Compares x with the number 0.	Compares x with the number on stack level Y.	Register no. 00 … 99	Look right for more about indirect addressing.	Sets scientific display with the number of decimals specified in stack level Z.	Stack level etc. X, Y, Z, ... , K	Register number 00 … 99
	Dot matrix display			**OP r nn** e.g. x≠ r23?			**OP → x** e.g. VIEW →L	**OP → nn** e.g. STO →45
				Compares x with the number stored in R23.			Shows the content of the register where L is pointing to.	Stores x into the location where R45 is pointing to.

---

[4] For RCL and STO, any of +, −, ×, /, ▲, or ▼ may precede step 2, except in RCLM and STOM. ENG . calls ENGOVR, SCI . calls SCIOVR. See the index of operations.

[5] You may skip this for register numbers >19.

[6] Exceptions: RCL T, RCL× T, RCL Z, RCL+ Z require an ENTER↑ preceding T or Z, e.g. RCL + ENTER↑ Z for the latter. This holds for STO as well.

[7] Register and flag numbers may be 00 … 99, number of decimals 0 … 11, integer bases 2 … 16, bit numbers 0 to 63, and integer word size up to 64 bits. For numbers <10, you may key in e.g. 5 ENTER↑ instead of 0 5 . There are three additional flags addressed via B, C, and D . – Take into account some registers may be allocated to special applications.

COMPARING AND ADDRESSING COMPLEX NUMBERS

1	**User input**	CPX, x= ? or x≠ ?				CPX RCL, STO, or x⇄		
	Dot matrix display	OP _ (with temporary alpha mode set) e.g. ▫x= _				OP _ (with temporary alpha mode set) e.g. ▫RCL _ [8]		
2	**User input**	0 or 1	Stack level or named register X, Z, A, C, L, or J	ENTER↑ [9] leaves temp. alpha mode	→ opens indirect addressing.	Stack level or named register Z [10], A, C, L, or J	Register number 0 0 .. 9 8 [11]	→ opens indirect addressing.
	Dot matrix display	OP n e.g. ▫x= 0 ?	OP x e.g. ▫x≠ z ?	OP r_	OP →_	OP x e.g. ▫RCL L	OP nn e.g. ▫STO 18	OP →_
3	**User input**	Compares $x + i\,y$ with the real number **0**.	Compares $x + i\,y$ with $z + i\,t$.	Register number 0 0 … 9 8	Look right for more about indirect addressing.	This is ᶜLASTx.	Stack level or named register X, Y, ... , K	Register number 0 0 … 9 9
	Dot matrix display			OP r nn e.g. ▫x≠ r26?			OP → x e.g. ▫x⟨⟩ →Z	OP → nn e.g. ▫STO →45
				Compares $x + i\,y$ with $r26 + i\,r27$.			Swaps **x** with the contents of the register where **Z** is pointing to, and **y** with the contents of the next one.	Stores $x + i\,y$ into 2 consecutive registers, starting with the one where **R45** is pointing to.

---

[8] For RCL and STO, any of +, −, ×, or / may precede step 2. See the index of operations.

[9] You may skip this keystroke for register numbers >19.

[10] Exceptions: ᶜRCL Z, ᶜRCL + Z, ᶜSTO Z, and ᶜSTO + Z require an ENTER↑ preceding Z, e.g. CPX STO + ENTER↑ Z for the latter.

[11] You may key in e.g. 8 ENTER↑ instead of 0 8 . Take care of pairs, since a complex operation will always affect two registers: the one specified and the one following this. We strongly recommend storing complex numbers with their real parts at even register numbers. – Take into account some registers may be allocated to special applications.

ADDRESSING LABELS

			XEQ, GTO, LBL, LBL?, SLV, ∫, π or Σ			
1 User input	A, B, C, or D					
Dot matrix display	**XEQ *label*** e.g. `XEQ C`		**OP _** e.g. `GTO _`			
2 User input	Calls the function labeled **C**.	A, B, C, or D	**ENTER↑** sets alpha mode.	**→** [12] opens indirect addressing and sets temporary alpha mode.		*2-digit numeric label* 0 0 … 9 9
Dot matrix display		**OP *label*** e.g. `Σ B`	**OP '_**	**OP →_**		**OP *nn*** e.g. `LBL 07`
3 User input	Sums up the function labeled **B**.		*Alphanumeric label* (≤ 3 characters [13])	*Stack level or named register* X, Y, Z, ... , K	*Register number* 0 0 … 9 9 [14]	
Dot matrix display			**OP '*label*'** e.g. `SLV'F1µ'`	**OP → *x*** e.g. `∫ →T`	**OP → *nn*** e.g. `XEQ →44`	
			Solves the function **F1µ** (with **F1µ** keyed in as explained in footer).	Integrates the function whose label is on stack level **T**.	Executes the routine whose label is in **R44**.	

Additionally, see GTO in the *index of operations* for some special cases applying exclusively there.

---

[12] Works with all these operations except LBL .

[13] The 3rd character terminates entry and closes alpha mode – shorter labels need a closing **ENTER↑** . For the example given here you just key in
f 2 ENTER↑ CPX 1 f EXIT g 7 and you are done. Statements including labels being 2 or 3 characters long will take two program steps.
   WARNING: LBL A and LBL'A' are different animals! The latter is entered in alpha mode, the first via the hotkey directly.

[14] Some registers may be allocated to special applications. Please check the memory table above.

STATISTICAL DISTRIBUTIONS, PROBALITIES ETC.

You find a lot of statistics in the WP 34S. Many preprogrammed functions are implemented here for the first time in an RPN scientific calculator – we packed all in what we always had missed. All of these functions, however, have a few features in common:

- Whenever we sum up a probability mass function (pmf [15]) $p(n)$ to get a cumulated distribution function (cdf) $F(m)$ we start at $n = 0$. Thus,

$$F(m) = \sum_{i=0}^{m} p(i) \ .$$

- Whenever we integrate a function, we start at the left end of the integration interval. Thus, integrating a probability density function (pdf) $f(x)$ to get a cdf $F(x)$ works as

$$F(x) = \int_{-\infty}^{x} f(\xi)d\xi = P(x) \ .$$

- Typically, F starts with a very shallow slope, becomes steeper then, and runs out with a decreasing slope while slowly approaching 100%. Obviously you get the most precise results on the left side of the cdf using $P(x)$. On its right side, however, the "error probability" $Q(x) = 1 - P(x)$ is more precise since $P(x)$ comes very close to 1 there. The digits available shall be sufficient in any case.

- On the WP 34S, with an arbitrary cdf named **XYZ** you find **XYZ$_P$** for the pdf or pmf, and **XYZ^{-1}** for the inverse cdf, unless stated otherwise explicitly.

- For calculating confidence limits for the "true value" based on a sample evaluation, employing a particular confidence level (e.g. 95%), you must know your objective:

  o Do you want to know the upper limit, under which the „true value" will lie with a 95% probability? Then take 0.95 as the argument of the inverse cdf to get said limit, and remember there is an inevitable chance of 5% for the "true value" being greater than it.

  o Do you want an upper <u>and</u> a lower limit confining the "true value"? Then there is an inevitable 2.5% chance for said value being less than the lower limit and an equal chance for it being greater than the upper limit. So you shall use 0.025 and 0.975 as arguments in two subsequent calculations using the inverse cdf to get both limits.

  o If you cannot live with these chances, inevitable as they are, employ an higher confidence level.

Turn to a good statistics textbook for more information, also about the particular distributions provided.

---

[15] pmf translates to German „Dichtefunktion", pdf to „Wahrscheinlichkeitsdichte", cdf to „Verteilungsfunktion" or „Wahrscheinlichkeitsverteilung".

DISPLAY AND MODES

The display features three sections: numeric, dot matrix and fixed symbols. The numeric section features a minus sign and 12 digits for the mantissa, as well as a minus sign and 3 digits for the exponent. The dot matrix is 6 dots high and 43 dots wide, allowing for some 7 to 12 characters, depending on their widths. The fixed symbols (except the big "=") are called *annunciators*, and are for indicating modes.

The dot matrix section above is used for

1. indicating some more modes than the annunciators allow,
2. passing additional information to the user.

The numeric section in the lower part of the LCD is used for displaying numbers in different formats, status, or messages.

If two or more requests concur for display space, the items will be shown according to their priorities as follows:

1. error messages as described in a *paragraph further below*,
2. special information as explained below,
3. information about the modes the calculator is running in.

The *annunciators* or specific characters in the LCD signal the modes:

Integer base or mode name	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	DECM
Signaled by …	b	3	4	5	6	7	o	9	d	-1	-2	-3	-4	-5	h	
Set by …	2	BASE3, BASE4, …					8	…	10			…, BASE15			16	.d
Cleared by …	any other BASE setting, FRACT, a b/c , d/c . FIX , SCI , ENG , H.MS , TIME , and ➡ H.MS will set DECM															

Mode name	PRG	α				FRC
Signaled by …	*STO*	*INPUT*	*360*	*RAD*	G	
Set by …	P/R	α αON	DEG	RAD	GRAD	d/c , a b/c 2nd . in input BASE1, FRACT
Cleared by …	P/R	ENTER αOFF	GRAD RAD	DEG GRAD	DEG RAD	BASE ≠ 1 H.MS , TIME , ➡ H.MS FIX , SCI , ENG

BEG indicates the program pointer standing at step 000 of program memory. A running program is signaled by a flashing **RCL** annunciator. The small equal sign **=** is lit while the WP 34S is in hibernation. **RPN** may be lit permanently. Time modes (12h / 24h) are seen in the time string directly. The numeric format of fraction mode is unambiguous as well. Further settings are signaled in the dot matrix section, like the different date modes being indicated there by **Y.MD** or **M.DY**. Defaults D.MY and DECM are not indicated. Please check the examples below.

Some mode and display settings may be stored and recalled collectively by STOM and RCLM. The command RCLM recalls a 19-bit word containing mode data packed as follows, starting with the least significant bit:

Bits	Meaning	Values and corresponding settings		
0, 1	Display format for real numbers	0 = ALL 2 = SCI	1 = FIX 3 = ENG	
2	Overflow for ALL	0 = SCIOVR	1 = ENGOVR	
3 … 6	Number of decimals	0 … 12		
7, 8	Angular mode	0 = DEG	1 = RAD	2 = GRAD
9, 10	Date display format	0 = D.MY	1 = Y.MD	2 = M.DY
11	Time display format	0 = 24h	1 = 12h	
12	Radix mark	0 = point	1 = comma	
13 … 15	Curve fit model	0 = LinF 2 = PowerF	1 = ExpF 3 = LogF	4 = BestF
16, 17	Integer sign mode	0 = 2COMPL 2 = UNSIGN	1 = 1COMPL 3 = SIGNMT	
18	Stack depth	0 = 4 levels	1 = 8 levels	

So the start-up default with 4 stack levels, ALL, SCIOVR, DEG, D.MY, 24h, decimal point, LinF, and 2COMPL equals zero in this mode word. On the other hand, settings for e.g. 8 stack levels, SCI 2, RAD, Y.MD, 12h, decimal comma, BestF, UNSIGN correspond to

$$1101001101010010010_2 = 69A92_{16} = 432786_{10} .$$

STOM takes such a number and sets the calculator modes accordingly. Please see the *index of operations* for more information about changing modes and the individual commands employed.

Some regional combinations may be set at once using a single command:

- SETCHN sets 24h, Y.MD, decimal point, and E3OFF;

- SETIND sets 24h, D.MY, decimal point, E3OFF, and JG1752;

- SETEUR sets 24h, D.MY, decimal comma, E3ON, and JG1582 (these settings apply also to South America);

- SETUSA sets 12h, M.DY, decimal point, E3ON, and JG1752;

- SETUK sets 12h, D.MY, decimal point, E3ON, and JG1752.

Please note the people living in the area of the former Soviet Union, in South Africa, Indonesia, and Vietnam use the decimal comma as well, but have different settings for dates and times.

Especially the angular modes deserve a closer look: there are three of them, DEG, RAD, and GRAD. And degrees (DEG) may be displayed in decimal numbers as well as in hours, minutes, seconds and hundredth of seconds (H.MS). Conversions are provided for going from one to the other:

to From	degrees H.MS	decimal degrees	radians	gon (grad)	current angular mode
degrees H.MS	—	→H.MS	—	—	—
decimal degrees	→H .d	—	rad→°	G→°	→DEG
radians	—	°→rad	—	G→rad	→RAD
gon/grad	—	°→G	rad→G	—	→GRAD
current angular mode	—	DEG→	RAD→	GRAD→	—

Please see the *index of operations* for the commands printed on white background, and the *catalog of conversions* for those printed on yellow.

Some commands and modes use the display in a special way. They are listed below in order of falling priority:

1. **VERS** generates a display similar to the one shown on the title page of this manual. Pressing any key will delete this message and return to previous state.

2. **STATUS** shows the status of 30 user flags very concisely in one display, allowing an immediate status overview after some training. If e.g. flags 2, 3, 5, 7, 11, 13, 14, 17, 19, and 23 are set, and labels B, C, and D are defined in program memory, STATUS will display this:

Within the numeric section, each row of horizontal bars in the mantissa shows the status of 10 flags. When a flag is set, the respective bar turns black. So here the top

row of bars indicates flags 0 and 1 are clear, 2 and 3 set, and flag 4 clear. Then, the divider II separates the first group of five flags from the next. Top row bars on its right side indicate flags 5 and 7 are set. Next row of bars shows flags 11, 13, 14, 17, 19 are set, and in the lowest row only flag 23 is set. All other flags in the range from 10 to 29 are clear.

Scrolling down by ▼ will display flags 10 - 39, then 20 - 49 etc. until 90 - D. Scrolling up by ▲ reverts this. Alternatively, pressing a digit, e.g. 5, will show up to 30 flags starting with 10 times this digit, e.g. flags 50 - 79. The numeric exponent always indicates the status of the hotkeys top left on the keyboard – if all four labels are used in program memory then **ALL** will be displayed there.

The status will be displayed until any key is pressed but ▼, ▲, or a digit.

3. During **command input**, the dot matrix displays the command chosen until input is completed, i.e. until all required trailing parameters are entered. The prefixes **f**, **g**, and **h** are shown until they are resolved. If you pressed any of **f**, **g**, or **h** erroneously, recovery is as easy as follows:

   o **f** **f** = NOP = **g** **g** = **h** **h** = CPX CPX = ➡ ➡

   o **g** **f** = **h** **f** = **f**
   f **g** = **h** **g** = **g**
   f **h** = **g** **h** = **h**

   In addressing, progress is recorded as explained in the *tables above* in detail.

4. In **programming mode**, the numeric display indicates the program step (000 – 505) in the mantissa and the number of free steps in the exponent, while the dot matrix shows the command contained in the respective step, e.g.:

$$\text{ʰRCL++→37} \quad _{RAD} \, ^{STO} \, _{RPN}$$
$$\text{StEP 195}^{267}$$

5. For **floating point decimal numbers**, the mantissa will be displayed adjusted to the right, the exponent to the left. Within the mantissa, either points or commas may be selected as radix marks [16], and additional marks may be chosen to separate thousands. Assume the display set to FIX 4, then 12.345678901 millions may look like:

$$12,345.67\,89.0\,10 \quad _{360}\,_{RPN}$$ or $$12.345.67\,89,0\,10 \quad _{360}\,_{RPN}$$

with thousands separators on, and without them like:

$$1234567\,89.0\,10 \quad _{360}\,_{RPN}$$ or $$1234567\,89,0\,10 \quad _{360}\,_{RPN}$$

---

[16] Starting here, decimal input is written using a point as radix mark throughout this manual, although significantly less visible, unless specified otherwise explicitly. By experience, the „comma people" are more capable to read radix points and interpret them correctly than vice versa.

These separators may also be beneficial in integer or fraction modes described below. – With ENG 3 and after changing the sign, the same number will look like this:

$$- 12.35^{6} \qquad \text{or} \qquad - 12{,}35^{6}$$
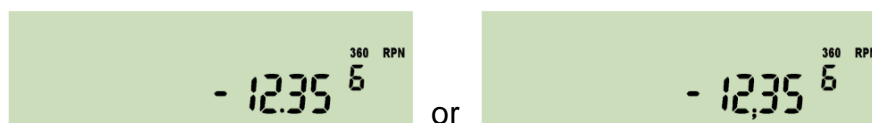
If the last operation executed was a complex one, a capital **C** is displayed top left in the dot matrix pointing to the fact that you find the result of this function in **X** and **Y**.

Floating point decimal numbers within $10^{-383} < x < 10^{+385}$ may be entered easily. Using a decimal mantissa, even numbers down to 10^{-394} can be keyed in. The calculator works with numbers down to 10^{-398} correctly. Smaller values are set to zero. For results $x \geq 10^{+385}$, error 4 or 5 will appear (see *below*).

6. In **integer modes**, numbers are displayed adjusted to the right as well. Word size and complement setting are indicated in the dot matrix using a format **xx.ww**, with **xx** being **1c** or **2c** for 1's or 2's complement, respectively, **un** for unsigned, or **sm** for sign-and-mantissa mode. Sign and first digit of the exponent show the base, a "c" in the second digit signals a carry bit set, an "o" in the third an overflow. Integer bases are indicated as follows:

Base	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sign and 1st digit of exponent displayed	b	3	4	5	6	7	o	9	d	-1	-2	-3	-4	-5	h

The example shows the WP 34S an arbitrary number in unsigned hexadecimal mode with word size 64 and carry set:

After changing to binary mode, this number will need 28 digits, being 1001001110100001010010110110. The 12 least significant digits will be displayed initially together with an indication that there are three display windows in total with the rightmost shown:
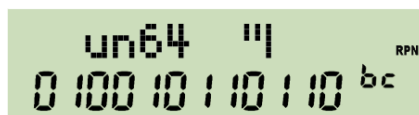
Now press ◄ and you will get the next 12 digits in the middle window:

Press ◄ again to show the most significant digits:

If leading zeros were turned on, there will be six display windows in this case, with the three "most significant" containing only zeros.

Please note numeric input is limited to 12 digits in any integer base.

7. **Fraction mode** works similar to HP-35S. In particular, DENMAX sets the maximum allowable denominator (see the *index of operations*). Display will look like in the examples below. If the fraction is exactly equal, slightly less, or greater 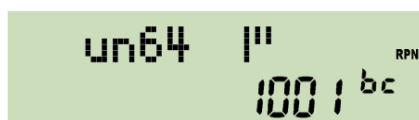than the floating point number converted, "=", "Lt", or "Gt" is indicated in the exponent, respectively. This mode can handle numbers with absolute values < 100,000 and > 0.0001. Maximum denominator is 9999. Underflows as well as overflows will be displayed in the format set before fraction mode was entered.

Now assume the WP 34S reset. Key in -47.40625 `a b/c` and you will see:

$$-47\ \ 13\ulcorner 32\ \ =$$

or after `d/c`:

$$-\ 15\ 17\ulcorner 32\ \ =$$

.

Please note integers like 123 will be displayed as "123 0/1" or "123/1" in fraction mode, respectively.

Squaring the improper fraction shown above results in

$$230\ 1289\ulcorner 1024\ \ =$$
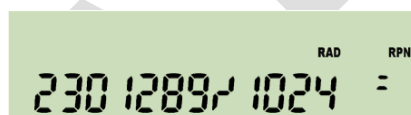
Now, enter `a b/c` for converting this result into a proper fraction. You will get

$$\llcorner 47\ \ 361\ulcorner 1024\ \ =$$
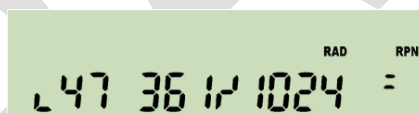
with a little hook left of the first digit shown. This indicates the leading number is displayed incompletely – there are at least two digits preceding 47 but no more display space. Press `SHOW` to unveil the integer part of this proper fraction is 2247.

Input in fraction mode is straightforward and logically coherent:

Key in:	and get in proper fraction mode:
①②．③．④ `ENTER↑`	12 $^{3}/_{4}$
①．② `ENTER↑`	1 $^{1}/_{5}$
．①．② `ENTER↑`	$^{1}/_{2}$
．①② `ENTER↑`	$^{3}/_{25}$ (= 0.12)
①．．② `ENTER↑`	1 $^{0}/_{1}$ (= 1 $^{0}/_{2}$!)

For comparison, please note the HP-32SII reads the last input here as ½ – which is, however, not consistent with its other input interpretations in fraction mode.

8. In **H.MS display mode**, format is `hhhh°mm'ss.dd"` with the number of hours or degrees limited to 9000. Output may look like this:

$$268°43'5 \text{l.}73^{"} \qquad \text{or} \qquad 268°43'5 \text{l,}73^{"}$$
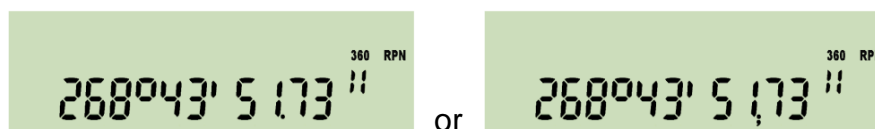
depending on the radix setting. For decimal times less than 5ms or 0.005 angular seconds but greater than zero, an "u" for underflow will be lit in the exponent section. For times or angles exceeding the upper limit, an "o" will be shown there signaling an overflow, and the value is displayed modulo 9000.

9. Output of the function **DAY** will look as follows for an input of 1.13201 in M.DY mode (equivalent to inputs of 13.01201 in D.MY or 2010.0113 in Y.MD). Expect similar displays after DAYS+.

$$\text{Wednesday} \ = \ \\ 3$$

10. In **alpha mode**, the alpha register is displayed in the dot matrix, showing the last characters it is containing, while the numeric section keeps the result of the last numeric operation, e.g.:

$$\text{Answer?}\\ -4,242^{-42}$$
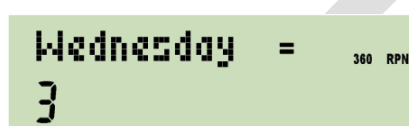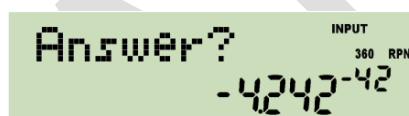
Different information may be appended to *alpha*. See the commands starting with "α" in the index of operations below. E.g. αTIME allows creating texts like

$$\text{It's 7:15pm}\\ -4,242^{-42} \qquad \text{or} \qquad \text{Um 19:15 Uhr}\\ -4,242^{-42}$$

depending on time mode setting (12h / 24h). And αDATE will append – depending on date format setting – either 2011-04-16 or 16.04.2011 or 04/16/2011 to *alpha*.

Please note *alpha* may contain up to 31 characters. And the WP 34S features a rich set of special letters. So you may store a message like

$$\text{ʿΟι μέλλοντε}\\ 0^{-398} \qquad \text{Ϙαιροι σε ϰα}\\ 0^{-398} \qquad \text{Ĺρέτοῦν.}\\ 0^{-398}$$
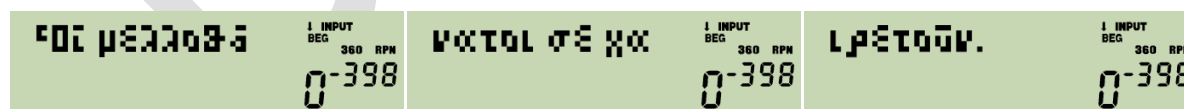
easily. Use [▲] and [▼] for browsing it in steps of 6 characters. Browsing to the left will stop with the very first characters shown, browsing to the right stops showing the right end completely, i.e.

$$\text{ϰαιρέτοῦν.}\\ 0^{-398}$$
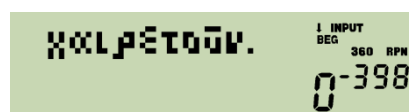
in this very special case.

All keyboard input will be interpreted according to the mode set at input time.

FONTS

The WP 34S features a large and a small font. Both are based on Luiz Viera's fonts as distributed in 2004. Some letters were added and some modified for better legibility, since the dot matrix is only 6 pixels high here. The following tables show the characters directly accessible through the keyboard. More are in the alpha catalogs (see *below*).

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ABCDEFGHIJKLM NOPQRSTUVWXYZ
ABCDEFGHIJKLM NOPQRSTUVWXYZ

a b c d e f g h i j k l m n o p q r s t u v w x y z
abcdefghijklm nopqrstuvwxyz
abcdefghijklm nopqrstuvwxyz

A B Γ Δ E Z H Θ I K Λ M N Ξ O Π P Σ T Y Φ X Ψ Ω
ABΓΔEZHΘIKΛM NΞOΠPΣTYΦXΨΩ
ABΓΔEZHΘIKΛM NΞOΠPΣTYΦXΨΩ

α β γ δ ε ζ η ϑ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω
αβγδεζηϑικλμ νξοπρστυφχψω
αβγδεζηϑικλμ νξοπρστυφχψω

0 1 2 3 4 5 6 7 8 9	( ) + - × / ± . !	↔ % √ \ & | ≠ $ € £ ¥
0123456789	()+−×/±.!	↔%√\&|≠$€£¥
0123456789	()+−×/±.!	↔%√\&|≠$€£¥

INDEX OF OPERATIONS

All commands available are found below with their *names* and *keystrokes* necessary. Names printed in **bold** face in this list belong to functions directly accessible on the keyboard, the other commands may be picked from catalogs. The command names will show up in program listings as well. Sorting in index and catalogs is case insensitive and works in the following order:

␣, 0…9, A…Z, α…ω, ( ) + − × / ± , . ! ? : ; ' " # * @ _ ~
→ ← ↑ ↓ ↔ < ≤ = ≠ ≥ > % $ € £ ¥ √ ∫ ∞ & \ ^ | G [ ] { }

Super- and subscripts are handled like normal characters in sorting. The fifth last item in the sorting order list above is the indicator for the angular mode GRAD.

Generally, functions and keystroke programming will work as on *HP-42S*, bit and integer functions as on *HP-16C*, unless stated otherwise under remarks. Especially, all **tests** will return "Yes" or "No" in the dot matrix if called from the keyboard; if called in a program, they will skip the next program line if the test is false. Please refer to the manuals of the vintage calculators mentioned for additional information about traditional commands.

Functions available on the WP 34S for the first time on an RPN calculator are highlighted yellow under remarks, while operations carrying a familiar name but deviating in their functionality here are marked light red.

Parameters will be taken from the lowest stack level(s) unless mentioned explicitly in the 2nd column – then they must follow the command. If ***underlined***, they may also be specified using indirect addressing, as shown in the *tables* above. Some parameters of statistical distributions must be given in registers **J** and **K** if specified.

In the following, each function is listed stating the mode(s) it will work in, abbreviated by their *indicators*. In this column an "&" stands for a Boolean AND, a comma for an OR, and a backslash for "not". So e.g. 2^x works in all modes but alpha. All operations may also be entered in mode PRG unless stated otherwise explicitly.

Name	Keys to press	in modes	Remarks
c…	CPX …	DECM	Indicates an operation allowing complex input(s) and/or complex results (see *above*). The prefix CPX may be heading all functions whose *names are printed in italics in this list*.
10ˣ	f 10ˣ	DECM	
12h	h MODE 12h	\α	Sets 12h time display mode meaning 1:23 becomes 1:23 AM and 13:45 becomes 1:45 PM. This makes a difference in αTIME only.
1COMPL	h MODE 1COMPL	\α	Sets 1's complement mode like in *HP-16C*.
1/x	f 1/x	DECM	
	B	DECM	Shortcut as long as label B is not defined yet.

Name	Keys to press	in modes	Remarks
24h	**h** **MODE** 24h	\α	Sets 24h time display mode meaning 1:23 AM becomes 1:23, and 1:45 PM becomes 13:45.
2COMPL	**h** **MODE** 2COMPL	\α	Sets 2's complement mode like in *HP-16C*.
2^x	**f** **2^x**	\α	
ABS	**f** **│x│**	\α	Returns the absolute value.
	CPX **f** **│x│**	DECM	Returns $r = \sqrt{x^2 + y^2}$ in **X** and clears **Y**.
ACOS	**g** **COS^{-1}**	DECM	Inverse cosine, also known as *arccos*.
ACOSH	**g** **HYP^{-1}** **COS**	DECM	Inverse hyperbolic cosine, known as *arcosh*. Note there is no need for pressing **f** here.
AGM	**h** **X.FCN** AGM	DECM	Returns the arithmetic-geometric mean of x and y.
ALL	**h** **ALL**	\α	Selects the format displaying "all" digits. With x falling out of the range (1E-12, 1E13) the display will change to SCI or ENG with the maximum number of digits displayable (see SCIOVR and ENGOVR below).
AND	**h** **AND**	Integer	Works bitwise as in *HP-16C*.
		DECM	Works like AND in *HP-28S*, i.e. x and y are interpreted before executing this operation. 0 is "false", any other real number is "true".
ANGLE	**h** **X.FCN** ANGLE	DECM	Returns the angle between positive x-axis and the straight line from the origin to the point (x, y), i.e. $arctan(^y/_x)$. This is a real two-number function.
	CPX **X.FCN** ANGLE	DECM	Calculates the angle as above, and returns it in **X** while clearing **Y**. This is a complex one-number function. Note there is no need for pressing **h** here.
ASIN	**g** **SIN^{-1}**	DECM	Inverse sine, also known as *arcsin*.
ASINH	**g** **HYP^{-1}** **SIN**	DECM	Inverse hyperbolic sine, known as *arsinh*.
ASR	**h** **X.FCN** ASR ***n***	Integer	Works like *n* (up to 63) consecutive ASRs in *HP-16C*. ASR 0 executes as NOP, but loads **L**.
ATAN	**g** **TAN^{-1}**	DECM	Inverse tangent, also known as *arctan*.
ATANH	**g** **HYP^{-1}** **TAN**	DECM	Inverse hyperbolic tangent, known as *artanh*.

Name	Keys to press	in modes	Remarks
BACK	$\boxed{\text{h}}\,\boxed{\text{P.FCN}}$ BACK **_n_**	PRG	Jumps **_n_** program steps backwards ($1 \le n \le 99$). So e.g. BACK 01 goes to the previous step. Reaching step 000 stops program execution. **ATTENTION:** The BACK instruction <u>must not</u> be the last step of the program.
BASE	$\boxed{\text{h}}\,\boxed{\text{MODE}}$ BASE **_n_**	\α	Sets the base for integer calculations, with $2 \le n \le 16$. Popular bases are directly accessible on the keyboard. Current integer base setting is indicated in the exponent as explained *above*. Furthermore, BASE0 equals DECM, and BASE1 calls FRACT. See below.
BASE10	$\boxed{\text{f}}\,\boxed{10}$		
BASE16	$\boxed{\text{g}}\,\boxed{16}$		
BASE2	$\boxed{\text{f}}\,\boxed{2}$		
BASE8	$\boxed{\text{g}}\,\boxed{8}$		
BATT	$\boxed{\text{h}}\,\boxed{\text{X.FCN}}$ BATT	DECM	Measures the battery voltage in the range between 1.9V and 3.4V and returns this value.
		Integer	As above but returns the voltage in 0.1V units.
BC?	$\boxed{\text{h}}\,\boxed{\text{TEST}}$ BC? **_n_**	Integer	Tests the specified bit in x.
BestF	$\boxed{\text{h}}\,\boxed{\text{STAT}}$ BestF	DECM	Selects the best curve fit model, maximizing the correlation like BEST does in *HP-42S*.
Binom Binom$_P$ Binom^{-1}	$\boxed{\text{h}}\,\boxed{\text{PROB}}$ Binom etc.	DECM	Binomial distribution with the number of successes g in $\mathbf{X}$, the probability of a success p_0 in $\mathbf{J}$ and the sample size n in $\mathbf{K}$: pmf: $p_B(g;n;p_0) = \binom{n}{g} \cdot p_0^{\,g} \cdot (1-p_0)^{n-g}$. cdf: $F_B(m;n;p_0) = \sum_{g=0}^{m} p_B(g;n;p_0)$, with the maximum number of successes m in $\mathbf{X}$. The pdf equals BINOMDIST(**_g; n; p₀;_** **0**) and the cdf BINOMDIST(**_m; n; p₀;_** **1**) in MS Excel. Binom^{-1} returns m for given probabilities F_B in $\mathbf{X}$ and p in $\mathbf{J}$ with sample size n in $\mathbf{K}$.
B$_n$	$\boxed{\text{h}}\,\boxed{\text{X.FCN}}$ B$_n$	DECM	Returns the Bernoulli number for an integer $n > 0$ given in $\mathbf{X}$: $B_n = (-1)^{n+1} n \cdot \zeta(1-n)$. See below for ζ.
B$_n$*	$\boxed{\text{h}}\,\boxed{\text{X.FCN}}$ B$_n$*	DECM	Returns the Bernoulli number according to its old definition for integer $n > 0$ given in $\mathbf{X}$: $B_n^* = \dfrac{2 \cdot (2n)!}{(2\pi)^{2n}} \cdot \zeta(2n)$. See below for ζ.

Name	Keys to press	in modes	Remarks
BS?	**h** TEST BS? _**n**_	Integer	Tests the specified bit in x .
Cauch Cauch$_P$ Cauch $^{-1}$	**h** PROB Cauch etc.	DECM	Cauchy-Lorentz distribution with the location x_0 specified in $\mathbf{J}$ and the shape γ in $\mathbf{K}$, also known as Lorentz or Breit-Wigner distribution: pdf: $f_{Ca}(x) = \dfrac{1}{\pi\gamma} \cdot \dfrac{1}{1 + \left(\dfrac{x - x_0}{\gamma}\right)^2}$ cdf: $F_{Ca}(x) = \dfrac{1}{2} + \dfrac{1}{\pi}\arctan\left(\dfrac{x - x_0}{\gamma}\right)$. Cauch^{-1} returns x for a given probability $\boldsymbol{F_{Ca}}$ in $\mathbf{X}$, with location $\boldsymbol{x_0}$ in $\mathbf{J}$ and shape $\boldsymbol{\gamma}$ in $\mathbf{K}$.
CB	**h** X.FCN CB _**n**_	Integer	Clears the specified bit in x .
CEIL	**h** X.FCN CEIL	DECM	Returns the smallest integer $\geq x$.
CF	**h** P.FCN CF _**n**_	\α	Clears the flag specified.
CLFLAG	**h** P.FCN CLFLAG	\α	Clears all user flags.
CLREG	**h** X.FCN CLREG	All	Clears all general purpose registers.
CLSTK	**0** **g** FILL **h** P.FCN CLSTK	\α	Clears all stack registers.
CLx	**h** CLx	All	Clears the lowest stack register and disables stack lift as usual.
CLα	**f** CLα	All	Clears the alpha register like CLA in *HP-42S*.
CLΣ	**g** CLΣ	DECM	Clears all statistical sums in the respective general purpose registers.
COMB	**f** Cy.x	DECM	Returns the number of possible <u>sets</u> of y items taken x at a time. No item occurs more than once in a set, and different orders of the same x items are <u>not</u> counted separately. Formula: $C_{y,x} = \begin{pmatrix} y \\ x \end{pmatrix} = \dfrac{y!}{x! \cdot (y - x)!}$
CONJ	CPX X.FCN CONJ	DECM	Changes the sign of y .

Name	Keys to press	in modes	Remarks
CORR	g r	DECM	Returns the correlation coefficient for the current statistical data and curve fitting model.
COS	f COS	DECM	Returns the cosine of the angle in X.
COSH	f HYP COS	DECM	Returns the hyperbolic cosine of x.
COV	h STAT COV	DECM	Returns the population covariance for two data sets. It depends on the fit model selected. For LinF, it calculates $$COV_{xy} = \frac{1}{n^2}\left(n\sum x_i y_i - \sum x_i \sum y_i\right)$$ See s_{xy} for the sample covariance.
CUBE	h X.FCN CUBE	\α	Returns x^3.
CUBERT	h X.FCN CUBERT	\α	Returns $\sqrt[3]{x}$.
DATE	h X.FCN DATE	DECM	Recalls the date from the real time clock and displays it in the numeric section in the format selected. See D.MY, M.DY, and Y.MD. The function DATE of *HP-12C* corresponds to DAYS+ in WP 34S (see below).
DAY	h X.FCN DAY	DECM	Takes x as a date in the format selected and returns the name of the day in the dot matrix and a corresponding integer in the numeric display (Monday = 1, Sunday = 7).
DAYS+	h X.FCN DAYS+	DECM	Works like DATE in *HP-12C*, adding x days on a date in Y in the format selected and displaying the resulting date including the day of week in the same format as DAY does.
DBLR	h X.FCN DBLR	Integer	Double precision commands for remainder, multiplication and division like in *HP-16C*.
DBL ×	h X.FCN DBL×	Integer	
DBL /	h X.FCN DBL/	Integer	
DEC	h P.FCN DEC *r*	\α	Decrements *r* by one, equivalent to 1 STO– *r* , but without modifying the stack.
DECM	f H.d	\α	Sets default decimal mode for calculations.

Name	Keys to press	in modes	Remarks
DECOMP	**h** X.FCN DECOMP	FRC	Decomposes *x* (after converting it into an improper fraction, if applicable), resulting in a stack [*numerator(x)*, *denominator(x)*, *y*, *z*] or [*num(x)*, *den(x)*, *y*, *z*, *t*, *a*, *b*, *c*] , respectively. Reversible by division.
DEG	**g** DEG	DECM	Sets angular mode to degrees.
DEG→	**h** X.FCN DEG→	DECM	Takes *x* as degrees and converts them to the angular mode currently set.
DENANY	**h** MODE DENANY	\α	Sets default fraction format like in *HP-35S*, allowing maximum precision. Any denominator up to the value set by DENMAX may appear.
DENFAC	**h** MODE DENFAC	\α	Sets "factors of the maximum denominator". With e.g. DENMAX = 60, possible denominators are 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, and 60.
DENFIX	**h** MODE DENFIX	\α	Sets fixed denominator format, i.e. the denominator equaling DENMAX always.
DENMAX	**h** MODE DENMAX	\α	Works like /c in *HP-35S*, but maximum denominator settable is 9999. It will be set to this value if $x < 1$ or $x > 9999$ at execution time. For $x = 1$ the current setting is recalled.
DISP	**h** MODE DISP	DECM	Changes the number of decimals while keeping the display format (FIX, SCI, ENG) as is.
DROP	**h** P.FCN DROP	\α	Drops *x* , changing stack contents to [*y*, *z*, *t*, *t*] or [*y*, *z*, *t*, *a*, *b*, *c*, *d*, *d*] , respectively. See *above* for ^CDROP.
DSE	**f** DSE *r*	PRG & DECM	Given cccccc.fffii in *r*, this function decrements *r* by ii, skipping next program line if then cccccc ≤ fff .
DSZ	**h** P.FCN DSZ *r*	PRG	Decrements *r* by one, skipping next program line if then $|r| < 1$.
D.MY	**h** MODE D.MY	\α	Sets the format for date display.
D→J	**h** X.FCN D→J	DECM	Takes *x* as a date in the format selected and converts it to a Julian day number according to JG…
D→R		DECM	Please see the *catalog of conversions below* for conversions from degrees to radians.

Name	Keys to press	in modes	Remarks
E3OFF	**h** **MODE** E3OFF	\α	Toggle the thousands separator (either a point or a comma depending on the radix setting).
E3ON	**h** **MODE** E3ON	\α	
ENG	**h** **ENG** _**n**_	\α	Sets engineering display format.
ENGOVR	**h** **ENG** **.**	\α	Numbers exceeding the range displayable in ALL or FIX will be shown in engineering format. See SCIOVR.
ENTER↑	**ENTER↑**	\α	See _above_ for CENTER.
ENTRY?	**h** **TEST** ENTRY?	All	Checks if at least one character was entered in response to a programmed STOP.
erf	**h** **X.FCN** erf	DECM	Returns the error function and its complementary, respectively:
erfc	**h** **X.FCN** erfc	DECM	$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\tau^2} d\tau \text{ and } erfc(x) = 1 - erf(x)$$
ERR	**h** **P.FCN** ERR _**n**_	PRG	Raises the error specified. See _below_ for the respective error codes.
EVEN?	**h** **TEST** EVEN?	\α	Checks if x is integer and even.
e^x	**f** e^x	DECM	
ExpF	**h** **STAT** ExpF	DECM	Selects the exponential curve fit model.
Expon Expon$_P$ Expon^{-1}	**h** **PROB** Expon etc.	DECM	Exponential distribution with the rate λ specified in **J**: pdf: $f_{Ex}(x) = \lambda \cdot e^{-\lambda x} = $ EXPONDIST(*x; λ; 0*), cdf: $F_{Ex}(x) = 1 - e^{-\lambda x} = $ EXPONDIST(*x; λ; 1*) in MS Excel. Expon^{-1} returns the survival time t_s for a given probability F_{Ex} in **X** and rate λ in **J**.
e^x -1	**h** **X.FCN** e^X-1	DECM	Returns more accurate results for the fractional part of e^X with $x \approx 0$.
FB	**h** **X.FCN** FB _**n**_	Integer	Inverts ("flips") the specified bit in x.
FC? FC?C FC?F FC?S	**h** **TEST** FC? _**n**_ etc.	\α	Tests if the flag specified is clear. Clears, flips, or sets this flag after testing, if applicable.

Name	Keys to press	in modes	Remarks
FF	**h** **P.FCN** FF *__n__*	\α	Flips the flag specified.
FIB	**h** **X.FCN** FIB	\α	Returns the Fibonacci number F_x.
FILL	**g** **FILL**	\α	Copies x to all stack levels. See *above* for ^{C}FILL.
FIX	**h** **FIX** *__n__*	\α	Sets fixed point display format.
FLOOR	**h** **X.FCN** FLOOR	DECM	Returns the largest integer $\leq x$.
FP	**g** **FP**	DECM	Returns the fractional part of x.
FP?	**h** **TEST** FP?	\α	Tests x for having a nonzero fractional part.
FRACT	**h** **MODE** FRACT	DECM	Sets fraction mode like in HP-35S, but keeps display format as set by PROFRC or IMPFRC.
FS?			
FS?C	**h** **TEST** FS? *__n__*	\α	Tests if the flag specified is set. Clears, flips, or sets this flag after testing, if applicable.
FS?F	etc.		
FS?S			
F(x)	**h** **PROB** F(x)	DECM	F-distribution. The cdf F(x) equals 1 - Q(F) in *HP-21S*. The degrees of freedom are specified in **J** and **K**.
F $^{-1}$(p)	etc.		
f'(x)	**h** **P.FCN** f'(x) *label*	DECM	Return the first or second derivative of *f(x)*, respectively, with the function *f(x)* being specified in a routine starting with LBL *label*. The return stack will have *y*, *z*, and *t* cleared and the position *x* in **L**.
f"(x)	**h** **P.FCN** f"(x) *label*		Either command will attempt to call a user routine labeled 'δx' to provide a fixed step size *dx*. If that routine is not defined, a step size of 0.1 is employed instead.
GCD	**h** **X.FCN** GCD	\α	Returns the Greatest Common Divisor of x and y.

Name	Keys to press	in modes	Remarks
Geom Geom$_P$ etc. Geom^{-1}	**h** `PROB` Geom	DECM	Geometric distribution: pdf: $f_{Ge}(m) = p_0(1-p_0)^m$, cdf: $F_{Ge}(m) = 1 - (1-p_0)^{m+1}$ is the probability for a first success after $m = x$ Bernoulli experiments. The probability p_0 for a success in each such experiment must be specified in **J**. Geom^{-1} returns the number of failures f before the first success for given probabilities F_{Ge} in **X** and p_0 in **J**.
GRAD	**g** `GRAD`	DECM	Sets angular mode to gon or grads.
GRAD→	**h** `X.FCN` GRAD→	DECM	Takes x as gon or grads and converts them to the angular mode currently set.
GTO	**h** `GTO` *label*	PRG	Inserts an unconditional branch to *label*.
		\PRG, \α	Positions the program pointer to *label*.
	h `GTO` `.` `A`, `B` , `C` , or `D`	\α	Positions the program pointer to one of these labels if defined (not programmable).
	h `GTO` `.` *nnn*	\α	Positions the program pointer to line *nnn* (not programmable).
	h `GTO` `.` `.`	\α	Positions the program pointer to line 000 (not progr.) and lights the annunciator *BEG*.
GTOα	**h** `P.FCN` GTOα	\α	Takes the first three characters of *alpha* (or less if there are less) as a label and positions the program pointer to it.
H$_n$	**h** `X.FCN` H$_n$	DECM	Hermite's polynomials for probability: $H_n(x) = (-1)^n \cdot e^{x^2/2} \cdot \dfrac{d^n}{dx^n}\left(e^{-x^2/2}\right)$ with n in **Y**, solving the differential equation $$f''(x) - 2x \cdot f'(x) + 2n \cdot f(x) = 0 .$$
H$_{np}$	**h** `X.FCN` H$_{np}$	DECM	Hermite's polynomials for physics: $H_{np}(x) = (-1)^n \cdot e^{x^2} \cdot \dfrac{d^n}{dx^n}\left(e^{-x^2}\right)$ with n in **Y**.

Name	Keys to press	in modes	Remarks
H.MS	**f** **H.MS**	DECM	Assumes $\mathbf{X}$ containing *decimal* hours or degrees, and displays them converted in the format `hhhh°mm′ss.dd″` as shown *above*. Will return to the previous decimal display with the next keystroke thereafter.
H.MS+	**h** **P.FCN** H.MS+	DECM	Assumes $\mathbf{X}$ and $\mathbf{Y}$ containing times or degrees in the format `hhhh.mmssdd`, and adds or subtracts them, respectively.
H.MS–	**h** **P.FCN** H.MS–		
IMPFRC	**g** **d/c**	\a	Sets fraction mode allowing improper fractions in display (i.e. $^5/_3$ instead of $1\ ^2/_3$). Converts x according to the settings by DEN… Absolute decimal equivalents of x must not exceed 100,000. Compare PROFRC.
		FRC	Allows displaying improper fractions. Thus converts a proper fraction in $\mathbf{X}$ into the equivalent improper fraction, if applicable.
INC	**h** **P.FCN** INC *r*	\a	Increments r by one, equivalent to 1 STO+ *r*, but without modifying the stack.
INT?	**h** **TEST** INT?	\a	Tests x for being an integer, i.e. having a fractional part equal to zero. Compare FP?.
IP	**f** **IP**	DECM	Returns the integer part of x.
ISG	**g** **ISG** *r*	PRG & DECM	Given `cccccc.fffii` in *r*, this function increments *r* by `ii`, skipping next program line if then `ccccccc > fff`.
ISZ	**h** **P.FCN** ISZ *r*	PRG	Increments *r* by one, skipping next program line if then $|r| < 1$.
I β	**h** **X.FCN** Iβ	DECM	Returns the regularized incomplete beta function $$\frac{\beta_x(x,y,z)}{\beta(y,z)} = \frac{1}{\beta(y,z)} \cdot \int_0^x t^{y-1}(1-t)^{z-1}\,dt \qquad \text{with} \quad \beta_x$$ being the incomplete beta function and β being Euler's beta (see below).
I Γ	**h** **X.FCN** IΓ	DECM	Returns the regularized incomplete gamma function $\dfrac{\gamma(x,y)}{\Gamma(x)}$ with $\gamma(x,y) = \int_0^y t^{x-1}e^{-t}\,dt$ being the lower incomplete gamma function. For Γ see below.

Name	Keys to press	in modes	Remarks
JG1582	**h** (X.FCN) JG1582	DECM	These two commands reflect different dates the Gregorian calendar was introduced in different large areas of the world. D→J and J→D will be calculated accordingly.
JG1752	**h** (X.FCN) JG1752		
J→D	**h** (X.FCN) J→D	DECM	Takes x as a Julian day number and converts it to a date according to JG... in the format selected
KEY?	**h** (TEST) KEY? **_a_**	DECM	Tests if a key was pressed while a program was running or paused. If <u>no</u> key was pressed, the next program step after KEY? will be executed, else it will be skipped and the code of said key will be found in address **_a_**. Key codes start top left ((A) is 11, (CPX) is 16, (STO) is 21, (+) is 75).
LASTx	(RCL) (L)	\α	See *above* for ^{C}LASTx .
LBL	**f** (LBL) *label*	PRG	Identifies programs and routines for execution and branching. See opportunities for specifying **_label_** in the table *above*.
LBL?	**h** (TEST) LBL? **_label_**	All	Tests for the existence of the label specified, anywhere in program memory. See opportunities for specifying **_label_** in the table *above*.
LCM	**h** (X.FCN) LCM	\α	Returns the Least Common Multiple of x and y.
LEAP?	**h** (TEST) LEAP?	DECM	Takes x as a date in the format selected, extracts the year, and tests for a leap year.
LgNrm	**h** (PROB) LgNrm etc.	DECM	Lognormal distribution with $\mu = \ln \bar{x}_g$ specified in **J** and $\sigma = \ln \varepsilon$ in **K**. See x̄g and ε below. pdf: $f_{Ln}(x) = \dfrac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}$, cdf: $F_{Ln}(x) = \Phi\left(\dfrac{\ln x - \mu}{\sigma}\right)$ with $\Phi(z)$ denoting the standard normal cdf. LgNrm $^{-1}$ returns x for a given probability **_F_Ln** in **X**, **_μ_** in **J**, and **_σ_** in **K**.
LgNrm_P			
LgNrm $^{-1}$			
LinF	**h** (STAT) LinF	DECM	Selects the linear curve fit model.
LJ	**h** (X.FCN) LJ	Integer	Left adjust as in *HP-16C*.
LN	**g** (LN)	DECM	Returns the (natural) logarithm of x for base e.

Name	Keys to press	in modes	Remarks
L_n	**h** X.FCN L_n	DECM	Laguerre's polynomials (compare $L_n\alpha$ below): $$L_n(x) = \frac{e^x}{n!} \cdot \frac{d^n}{dx^n}\left(x^n e^{-x}\right) = L_n^{(0)}(x) \quad \text{with} \quad \textbf{n} \text{ in } \textbf{Y},$$ solving the differential equation $$x \cdot y'' + (1-x)y' + ny = 0 \, .$$
LN1+x	**h** X.FCN LN1+x	DECM	Natural logarithm of values close to zero. Returns $\ln(1+x)$, providing a much higher accuracy in the fractional part of the result.
$L_n\alpha$	**h** X.FCN $L_n\alpha$	DECM	Laguerre's generalized polynomials with **n** in **Y** and α in **Z**: $$L_n^{(\alpha)}(x) = \frac{x^{-\alpha}e^x}{n!} \cdot \frac{d^n}{dx^n}\left(x^{n+\alpha}e^{-x}\right) \, .$$
LNβ	**h** STAT LNβ	DECM	Returns the natural logarithm of Euler's β function. See there. Also contained in X.FCN.
LNΓ	**h** STAT LNΓ	DECM	Returns the natural logarithm of $\Gamma(x)$. See there. Also contained in the catalog X.FCN.
LOAD	**h** X.FCN LOAD	\α	Restore the entire backup. See SAVE.
LOG_{10}	**g** LG	DECM	Returns the logarithm of x for base 10.
LOG_2	**g** LB	\α	Returns the logarithm of x for base 2.
LogF	**h** STAT LogF	DECM	Selects the logarithmic curve fit model.
Logis	**h** PROB Logis etc.	DECM	Logistic distribution with μ given in **J** and s in **K**
Logis$_P$			pdf: $f_{Lg}(x) = e^{-\frac{x-\mu}{s}} \bigg/ s \cdot \left(1 + e^{-\frac{x-\mu}{s}}\right)^2$,
			cdf: $F_{Lg}(x) = \left(1 + e^{-\frac{x-\mu}{s}}\right)^{-1}$
Logis $^{-1}$			Logis $^{-1}$ returns $F_{Lg}^{-1}(p) = \mu + s \cdot \ln\left(\frac{p}{1-p}\right)$ for a probability p given in **X**, μ in **J**, and s in **K**.
LOGx	**g** LOGx	DECM	Returns the logarithm of y for base x.
	CPX **g** LOGx	DECM	Returns the complex logarithm of $z + i\,t$ for the complex base $x + i\,y$.

Name	Keys to press	in modes	Remarks
LZOFF	**h** `MODE` LZOFF	Integer	Toggles leading zeros like flag 3 does in *HP-16C*.
LZON	**h** `MODE` LZON		
L.R.	**h** `L.R.`	DECM	Returns the parameters *a1* and *a0* of the fit curve through the data points accumulated, according to the model selected, and pushes them on the stack. For a straight regression line, *a0* is the y-intercept and *a1* the slope.
MASKL	**h** `X.FCN` MASKL ___n___ etc.	Integer	Work like MASKL and MASKR on *HP-16C*, but with the mask length following the command instead of taken from **X**.
MASKR			
MAX	**h** `X.FCN` MAX	\α	Returns the maximum of x and y.
MIN	**h** `X.FCN` MIN	\α	Returns the minimum of x and y.
MIRROR	**h** `X.FCN` MIRROR	Integer	Reflects the bit pattern in x (e.g. 000101 becomes 101000 for word size 6).
M.DY	**h** `MODE` M.DY	\α	Sets the format for date display.
NAND	**h** `X.FCN` NAND	\α	Works in analogy to AND.
NaN?	**h** `TEST` NaN?	\α	Tests x for "Not a Number".
nBITS	**h** `X.FCN` nBITS	Integer	Counts bits set in x like #B does on *HP-16C*.
NOP	**h** `P.FCN` NOP	PRG	"Empty" step FWIW.
NOR	**h** `X.FCN` NOR	\α	Works in analogy to AND.
Norml	**h** `PROB` Norml etc.	DECM	Normal distribution with an arbitrary mean $\boldsymbol{\mu}$ specified in **J** and standard deviation $\boldsymbol{\sigma}$ in **K** : pdf: $f_N(x) = \dfrac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, cdf: $F_N(x) = \Phi\left(\dfrac{x-\mu}{\sigma}\right)$ with Φ denoting the standard normal distribution. The pdf equals NORMDIST($x; \boldsymbol{\mu}; \boldsymbol{\sigma}; 0$) and the cdf NORMDIST($x; \boldsymbol{\mu}; \boldsymbol{\sigma}; 1$) in MS Excel. Norml^{-1} returns x for a given probability F_N in **X**, mean $\boldsymbol{\mu}$ in **J**, and standard deviation $\boldsymbol{\sigma}$ in **K**. Equals NORMINV($F_N; \boldsymbol{\mu}; \boldsymbol{\sigma}$) in MS Excel.
Norml$_P$			
Norml^{-1}			
NOT	**h** `NOT`	\α	Works in analogy to AND.

Name	Keys to press	in modes	Remarks
nΣ	**h** **STAT** nΣ	DECM	Recalls the number of accumulated data points. Necessary for basic statistics.
ODD?	**h** **TEST** ODD?	\α	Checks if x is integer and odd.
OFF	**h** **P.FCN** OFF	PRG	Turns the WP 34S off under program control.
OR	**h** **OR**	\α	Works in analogy to AND.
PERM	**g** **Py,x**	DECM	Returns the number of possible <u>arrangements</u> of y items taken x at a time. No item occurs more than once in an arrangement, and different orders of the same x items <u>are</u> counted separately. Formula: $P_{y,x} = x! \cdot C_{y,x}$, see COMB.
P_n	**h** **X.FCN** P_n	DECM	Legendre's polynomials: $P_n(x) = \dfrac{1}{2^n n!} \cdot \dfrac{d^n}{dx^n}\left[(x^2-1)^n\right]$ with n in **Y**, solving the differential equation $$\dfrac{d}{dx}\left[(1-x^2)\cdot\dfrac{d}{dx}f(x)\right] + n(n+1)f(x) = 0 .$$
Poiss Poiss$_P$ Poiss^{-1}	**h** **PROB** Poiss etc.	DECM	Poisson distribution with the number of successes g in **X**, gross error probability p_0 in **J**, and sample size n in **K**. Alternatively, Poisson's $\lambda = n \cdot p_0$ may be in **J** if $k = 1$: pmf: $P_P(g;\lambda) = \dfrac{\lambda^g}{g!}e^{-\lambda}$, cdf: $F_P(m;\lambda) = \sum_{g=0}^{m} P_P(g;\lambda)$, with the maximum number of successes m in **X**. The pdf equals POISSON($g; \lambda; 0$) and the cdf POISSON($g; \lambda; 1$) in MS Excel. Poiss^{-1} returns m for given probabilities F_P in **X** and p in **J** with sample size n in **K**.
PowerF	**h** **STAT** PowerF	DECM	Selects the power curve fit model.
PRCL	**h** **X.FCN** PRCL n	\α	Recall the user program space from flash segment n (see *above*).
PRIME?	**h** **TEST** PRIME?	\α	Checks if the absolute value of the integer part of x is a prime.

Name	Keys to press	in modes	Remarks
PROFRC	**f** `a b/c`	DECM	Sets fraction mode like in *HP-35S*, allowing only proper fractions or mixed numbers in display. Converts x according to the settings by DEN… Absolute decimal equivalents of x must not exceed 100,000. Compare IMPFRC.
		FRC	Allows displaying only proper fractions. Thus converts an improper fraction in **X**, if applicable, e.g. 5/3 into 1 2/3.
PROMPT	**h** `P.FCN` PROMPT	PRG	Displays *alpha* and stops program execution (equaling αVIEW followed by STOP actually). If alpha input is requested for a program, use the sequence αON PROMPT αOFF. With a program running, enter the value or text requested and press `R/S` to continue.
PSE	**h** `PSE` **nn**	PRG	Refreshes the display and pauses program execution for **nn** times 0.1s, with $0 \leq nn \leq 99$. The pause will be terminated early as soon as a key is pressed.
PSTO	**h** `X.FCN` PSTO	\α	Store the user program space in flash segment **n** or exchange it with the contents of flash segment **n** , respectively (see *above*).
P↔	**h** `X.FCN` P↔		
RAD	**g** `RAD`	DECM	Sets angular mode to radians.
RAD→	**h** `X.FCN` RAD→	DECM	Takes x as radians and converts them to the angular mode currently set.
RAN#	**f** `RAN#`	DECM	Returns a random number between 0 and 1 like RAN in *HP-42S*.
		Integer	Returns a random bit pattern for the word size set.
RCF	**h** `X.FCN` RCF	\α	xxx
RCL	`RCL` **s**	\α	See the *addressing table above* for ^CRCL.
RCLM	`RCL` `MODE`	\α	Recalls selected mode settings into **X**. See the paragraph about *indicators* above for details about the mode word.
RCLS	**h** `P.FCN` RCLS **s**	\α	Recalls 4 or 8 values from a set of registers starting at address **s**, and pushes them on the stack. This is the converse command of STOS.

Name	Keys to press	in modes	Remarks
RCL+	RCL + *s*	\α	Recalls the content of address *s*, executes the specified operation on it and pushes the result on the stack. E.g. RCL–12 recalls *r12*, subtracts *x* from it and displays the result. RCL↑ (↓) recalls the maximum (minimum) of the values in *s* and **X**. See the *addressing table above* for ᶜRCL.
RCL–	RCL – *s*		
RCL×	RCL × *s*		
RCL/	RCL / *s*		
RCL↑	RCL ▲ *s*		
RCL↓	RCL ▼ *s*		
RDX,	h MODE RDX,	DECM	Sets the decimal mark to a comma. Also contained in catalog P.FCN.
RDX.	h ./.		Toggle the radix mark.
	h MODE RDX.	DECM	Sets the decimal mark to a point. Also contained in catalog P.FCN.
RJ	h X.FCN RJ	Integer	Right adjusts, in analogy to LJ on *HP-16C*.
RL	h X.FCN RL *n*	Integer	Works like *n* consecutive RLs / RLCs on *HP-16C*. For RL, 1 ≤ *n* ≤ 63. For RLC, 1 ≤ *n* ≤ 64. RL 0 and RLC 0 execute as NOP.
RLC	h X.FCN RLC *n*		
RMDR	h RMDR	\α	MOD of *HP-42S* equals RMD of *HP-16C*.
ROUND	g RND	DECM	Rounds *x* using the current display format, like RND in *HP-42S*.
		FRC	Rounds *x* using the current denominator, like RND in *HP-35S*.
ROUNDI	h X.FCN ROUNDI	DECM	Rounds *x* to next integer. ½ rounds to 1.
RR	h X.FCN RR *n*	Integer	Works like *n* consecutive RRs / RRCs on *HP-16C*. See RL / RLC for more.
RRC	h X.FCN RRC *n*		
RRCL	h X.FCN RRCL	\α	Recover the general purpose registers from the backup region (see SAVE and *above*).
RTN	g RTN	\PRG	Moves the program pointer to step 000.
		PRG	Last command in a routine. Returns control to the calling routine in program execution, i.e. moves the program pointer one step behind the most recent XEQ instruction encountered. If there is none, program execution halts.

Name	Keys to press	in modes	Remarks
RTN+1	**h** **P.FCN** RTN+1	PRG	Returns control to the calling routine like RTN does, but moves the program pointer to the <u>second</u> line following the most recent XEQ instruction encountered. If there is no matching XEQ, program execution halts.
R-CLR	**h** **P.FCN** R-CLR	DECM	Interprets x in the form `ss.nn`. Clears **nn** registers starting with number **ss**. E.g. for x = 34.56, R-CLR will clear **R34** through **R89**.
R-COPY	**h** **P.FCN** R-COPY	DECM	Interprets x in the form `ss.nndd`. Takes **nn** registers starting with number **ss** and copies their contents to **dd**. E.g. for x = 7.0345678, *r07, r08, r09* will be moved into **R45, R46, R47**, respectively.
R-SORT	**h** **P.FCN** R-SORT	DECM	Interprets x in the form `ss.nn`. Sorts the contents of **nn** registers starting with number **ss**. Assume x = 49.026, *r49* = 1.2, *r50* = –3.4; then R-SORT returns *r49* = –3.4, *r50* = 1.2.
R-SWAP	**h** **P.FCN** R-SWAP	DECM	Works like R-COPY but swaps the register contents of source and destination.
R→D		DECM	Please see the *catalog of conversions below* for conversions of radians to degrees.
R↑	**h** **R↑**	\α	Rotates the stack contents one level up or down, respectively. See *above* for complex rotations.
R↓	**R↓**		
s	**g** **s**	DECM	Takes the statistical sums, calculates the sample standard deviations s_y and s_x and pushes them on the stack.
SAVE	**h** **X.FCN** SAVE	\α	Saves user program space, registers and system state to flash memory. Program space is stored in segment 0. Registers and system state are in their own special region.
SB	**h** **X.FCN** SB **_n_**	Integer	Sets the specified bit in x.
SCI	**h** **SCI** **_n_**	\α	Sets scientific display format.
SCIOVR	**h** **SCI** **.**	\α	Numbers exceeding the range displayable in ALL or FIX will be shown in scientific format (default as in vintage HP calculators). Compare ENGOVR.

Name	Keys to press	in modes	Remarks
SEED	h STAT SEED	DECM	Stores a seed for random number generation.
SERR	h STAT SERR	DECM	Works like s but pushes the standard errors $s/\sqrt{n}$ on the stack (i.e. the standard deviations of x̄ and y).
SERRw	h STAT SERRw	DECM	Works like sw but returns the standard error $s/\sqrt{\sum y_i}$ (i.e. the standard deviation of x̄w).
SETCHN	h MODE SETCHN	All	Sets some regional preferences (see *above*).
SETDAT	h X.FCN SETDAT	DECM	Sets the date for the real time clock (doesn't work with the emulator, since it takes this information from the PC clock).
SETEUR	h MODE SETEUR etc.	All	Set some regional preferences (see *above*).
SETIND			
SETTIM	h X.FCN SETTIM	DECM	Sets the time for the real time clock (doesn't work with the emulator, since it takes this information from the PC clock).
SETUK	h MODE SETUK etc.	All	Set some regional preferences (see *above*).
SETUSA			
SF	h P.FCN SF *n*	\α	Sets the flag specified.
SIGN	h X.FCN SIGN	\α	Returns 1 for $x > 0$, −1 for $x < 0$, and 0 for $x = 0$ or non-numbers.
	CPX X.FCN SIGN	DECM	Returns the unit vector of $x + i\,y$ in **X** and **Y**.
SIGNMT	h MODE SIGNMT	\α	Sets sign-and-mantissa mode for integers.
SIN	f SIN	DECM	Returns the sine of the angle in **X**.
SINC	h X.FCN SINC	DECM	Returns $\dfrac{\sin(x)}{x}$.
SINH	f HYP SIN	DECM	Returns the hyperbolic sine of x.
SKIP	h P.FCN SKIP *n*	PRG	Skips *n* program steps forwards ($1 \le n \le 99$). So e.g. SKIP 02 skips over the next two steps, going e.g. from step 123 to step 126. If the skip would land beyond the end of occupied program memory, the same will happen as if a RTN had been encountered.

Name	Keys to press	in modes	Remarks
SL	**h** [X.FCN] SL __n__	Integer	Works like __n__ (up to 63) consecutive SLs on *HP-16C*. SL 0 executes as NOP.
SLV	**f** [SLV] __label__	DECM	Solves the equation $f(x) = 0$, with $f(x)$ calculated by the routine specified. Two initial estimates of the root must be supplied in $\mathbf{X}$ and $\mathbf{Y}$ when calling SLV. For the rest, the user interface is as in *HP-15C*.
SLVQ	**h** [X.FCN] SLVQ	DECM	Assumes the stack containing the parameters [__c, b, a__, …] of an equation $ax^2 + bx + c = 0$, and returns its two roots $-\dfrac{b}{2a} \pm \sqrt{\left(\dfrac{b}{2a}\right)^2 - \dfrac{c}{a}}$ in $\mathbf{Y}$ and $\mathbf{X}$. Then $\mathbf{Z}$ will contain what was in $\mathbf{T}$ when SLVQ was called, etc. Please note SLVQ works for real numbers only.
SPEC?	**h** [TEST] SPEC?	\α	True if x is special, i.e. infinity or NaN.
SR	**h** [X.FCN] SR __n__	Integer	Works like __n__ consecutive SRs on *HP-16C*. SR 0 executes as NOP.
SRCL	**h** [X.FCN] SRCL	\α	Recover the system state from the backup region (see SAVE and *above*).
SSIZE4	**h** [MODE] SSIZE4	\α	Set the stack size to 4 or 8 levels, respectively. If stack size grows, the top level contents will be copied into the new levels. If the stack shrinks, previous top levels will be lost.
SSIZE8	**h** [MODE] SSIZE8		The same will happen if stack size is changed via STOM.
SSIZE?	**h** [TEST] SSIZE?	\α	Returns the number of stack levels accessible.
STO	[STO] __d__	\α	See the *addressing table above* for ^{C}STO.
STOM	[STO] [MODE]	\α	Sets selected modes as encoded in x. See the paragraph about *indicators* above for details about the mode word.
STOP	[R/S]	PRG	Stops program execution. May be used to wait for an input, for example.
STOS	**h** [P.FCN] STOS __d__	\α	Stores all stack levels in a set of 4 or 8 registers, starting at destination __d__.

Name	Keys to press	in modes	Remarks
STO+	[STO] [+] *d*	\α	Executes the specified operation on the content of address *d* and stores the result into said address.

E.g. STO–12 subtracts x from *r12*, and stores the result in **R12** again. STO↑ (↓) takes the maximum (minimum) of the values in *d* and **X** and stores it.

See the *addressing table above* for ^{C}STO. |
STO–	[STO] [–] *d*		
STO×	[STO] [×] *d*		
STO/	[STO] [/] *d*		
STO↑	[STO] [▲] *d*		
STO↓	[STO] [▼] *d*		
SUM	[h] [STAT] SUM	DECM	Recalls the linear sums *Σy* and *Σx*. Useful for elementary vector algebra.
sw	[h] [STAT] sw	DECM	Returns the standard deviation for weighted data $$s_w = +\sqrt{\frac{\sum y_i \cdot \sum (y_i \cdot x_i^2) - \left[\sum (y_i \cdot x_i)\right]^2}{\left(\sum y_i\right)^2 - \sum y_i^2}}$$ with the weights in y.
sxy	[h] [STAT] sxy	DECM	Returns the sample covariance for two data sets. It depends on the fit model selected. For LinF, it returns $$s_{xy} = \frac{1}{n \cdot (n-1)}\left(n\sum x_i y_i - \sum x_i \sum y_i\right).$$ See COV for the population covariance.
TAN	[f] [TAN]	DECM	Returns the tangent of the angle in **X**.
TANH	[f] [HYP] [TAN]	DECM	Returns the hyperbolic tangent of x.
TICKS	[h] [P.FCN] TICKS	\α	Returns the number of ticks from the real time clock at execution time.
TIME	[h] [TIME]	DECM, α	Recalls the time from the real time clock at execution, displaying it in the format hh.mmssdd in 24h-mode. Chose FIX 6 for best results.
T_n	[h] [X.FCN] T_n	DECM	Chebychev's (a.k.a. Tschebyschow) polynomials of first kind $T_n(x)$ with *n* in **Y**, solving the differential equation $$\left(1-x^2\right)y'' - x \cdot y' + n^2 y = 0.$$
t(x)

$t^{-1}(p)$ | [h] [PROB] t(x)

etc. | DECM | Student's t distribution. t(x) equals $1 - Q(t)$ in *HP-21S*. The degree of freedom is stored in **J**. |

Name	Keys to press	in modes	Remarks
U_n	**h** X.FCN U_n	DECM	Chebychev's polynomials of second kind $U_n(x)$ with n in **Y**, solving the differential equation $$\left(1-x^2\right)y''-3x\cdot y'+n(n+2)y=0 .$$
UNSIGN	**h** MODE UNSIGN	\a	Sets unsigned mode for integers.
VIEW	**h** VIEW **s**	All	Displays the content of address **s** until the next key is pressed.
W	**h** X.FCN W	DECM	W returns Lambert's W for given $x \geq -1/e$, while W^{-1} returns x for given W ($\geq$ -1).
W⁻¹	**h** X.FCN W⁻¹		
Weibl	**h** PROB Weibl etc.	DECM	Weibull distribution with the shape parameter *b* in **J** and the characteristic lifetime *T* in **K**: pdf: $f_W(t)=\dfrac{b}{T}\left(\dfrac{t}{T}\right)^{b-1}e^{-\left(\frac{t}{T}\right)^b}$, cdf: $F_W(t)=1-e^{-\left(\frac{t}{T}\right)^b}$. The pdf equals WEIBULL(*x; b; T;* **0**) and the cdf WEIBULL(*x; b; T;* **1**) in MS Excel. Weibl $^{-1}$ returns the survival time *t_s* for given probability F_W , *b* in **J** and *T* in **K**.
Weibl_P			
Weibl ⁻¹			
WSIZE	**h** MODE WSIZE **n**	\a	Works like on *HP-16C*, but with the parameter following the command instead of taken from **X**. Reducing the word size truncates the values in the stack registers employed, including **L**. WSIZE 0 sets the word size to maximum, i.e. 64 bits.
WSIZE?	**h** TEST WSIZE?	\a	Recalls the word size set.
x²	**g** x²	\a	
XEQ	XEQ *label*	PRG	Calls the respective subroutine.
		\PRG, \a	Executes the respective program.
	B , C , or D (you may need **f** for accessing these hotkeys in integer bases >10.)	PRG	Calls the respective subroutine, so e.g. XEQ C will be inserted when C is pressed.
		\PRG, \a	Executes the respective program if defined.
XEQα	**h** P.FCN XEQα	\a	Takes the first three characters of *alpha* (or less if there are less) as a label and calls or executes the respective routine.

Name	Keys to press	in modes	Remarks
XNOR	**h** X.FCN XNOR	\α	Works in analogy to AND.
XOR	**h** XOR	\α	Works in analogy to AND.
x̄	**f** x̄	DECM	Returns the arithmetic means, pushing $\bar{y} = \frac{1}{n}\sum y$ and $\bar{x} = \frac{1}{n}\sum x$ on the stack. See also s, SERR, and σ.
x̄g	**h** STAT x̄g	DECM	Returns the geometric means, pushing $\bar{y}_g = \sqrt[n]{\prod y} = e^{\frac{1}{n}\sum \ln y}$ and $\bar{x}_g = \sqrt[n]{\prod x}$ on the stack. See also ε, εg, and εP.
x̄w	**h** STAT x̄w	DECM	Returns the weighted arithmetic mean $\sum xy / \sum y$. See also sw and SERRw.
x̂	**h** STAT x̂	DECM	Returns a forecast *x* for a given *y* (in **X**) following the fit model chosen. See L.R. for more.
x !	**h** !	DECM	Return the factorial, equaling $\Gamma(x + 1)$.
x → α	**g** x◀▶a	All	Interprets *x* as a character code. Appends the respective character to *alpha*, similar to XTOA in *HP-42S*.
x↔	**h** x⋛ *r*	\α	Swaps the contents of **X** and *r* . See *above* for complex x↔ .
x↔y	x⋛y	\α	Swaps *x* and *y* , performing Re↔Im if a complex operation was executed immediately before. See *above* for ^Cx↔y .
x < … ?	**h** TEST x < ? *a*	\α	Compare *x* with *a*. The three dots will be replaced in the listing by *a* according to the examples given in the *addressing table above*. x ≈ ? will be true if the rounded values of *x* and *a* are equal (see ROUND). CPX **f** x = ? *a* and CPX **g** x ≠ ? *a* compare the complex number *x + i y* as explained in the *addressing table above*.
x ≤ … ?	**h** TEST x ≤ ? *a*		
x = … ?	**f** x = ? *a*		
x ≈ … ?	**h** TEST x ≈ ? *a*		
x ≠ … ?	**g** x ≠ ? *a*		
x ≥ … ?	**h** TEST x ≥ ? *a*		
x > … ?	**h** TEST x > ? *a*		

Name	Keys to press	in modes	Remarks
y^*x*	**f** y^x	\a	In integer modes *x* must be ≥ 0.
	C	\a, \13, \14, \15, \h	Shortcut working as long as label C is not defined yet.
ŷ	**f** ŷ	DECM	Returns a forecast *y* (in **X**) for a given *x* following the fit model chosen. See L.R. for more.
Y.MD	**h** MODE Y.MD	\a	Sets the format for date display.
αDATE	**h** X.FCN αDATE	\integer	Takes *x* as a date and appends it to *alpha* in the format set. See DATE. – To append a date stamp to *alpha*, call DATE αDATE.
αDAY	**h** X.FCN αDAY	\integer	Takes *x* as a date, recalls the name of the respective day and appends its first 3 letters to *alpha*.
αGTO	**h** P.FCN αGTO *nn*	\a	Takes the contents of **R**nn as character code. Takes the first three characters of the converted code (or less if there is only less) as an alpha label and positions the program pointer to it.
αIP	**h** X.FCN αIP	All	Appends the integer part of *x* to *alpha*, similar to AIP in *HP-42S*.
αLENG	**h** X.FCN αLENG	All	Returns the number of characters found in *alpha*, like ALENG in *HP-42S*.
αMONTH	**h** X.FCN αMONTH	\integer	Works like αDAY, but processing the month.
αOFF	**h** P.FCN αOFF	PRG & α	Work like AOFF and AON in *HP-42S*, turning alpha mode off and on.
αON	**h** P.FCN αON	PRG & \a	
αRCL	**f** RCL *s*	α	Interprets the content of the source *s* as characters and appends them to *alpha*.
	h X.FCN αRCL *s*	\a	
αRC#	**h** X.FCN αRC# *s*	All	Takes the content of *s* as a number, converts it to a string in the format set, and appends this to *alpha*. If e.g. *s* = 1234 and ENG 2 and RDX. are set, then ˍ1.23E3 will be appended.
αRL	**h** X.FCN αRL *n*	All	Rotates *alpha* by *n* characters like AROT in *HP-42S*, but with *n ≥ 0* and the parameter trailing the command instead of taken from **X**. αRL 0 executes as NOP.

Name	Keys to press	in modes	Remarks
αRR	**h** X.FCN αRR **_n_**	All	Works like αRL but rotates to the right.
αSL	**h** X.FCN αSL **_n_**	All	Shifts the **_n_** leftmost characters out of **_alpha_**, like ASHF in *HP-42S*. αSL 0 equals NOP.
αSR	**h** X.FCN αSR **_n_**	All	Works like αSL but takes the **_n_** rightmost characters instead.
αSTO	**f** STO **_d_**	α	Stores the first (i.e. leftmost) 6 characters in the alpha register into destination **_d_** .
	h X.FCN αSTO **_d_**	\α	
αTIME	**h** X.FCN αTIME	\integer	Takes **_x_** as a decimal time and appends it to **_alpha_** in the format hh:mm:ss according to the time mode selected. See TIME. – To append a time stamp to **_alpha_**, call TIME αTIME.
αVIEW	**h** P.FCN αVIEW	\α	Displays **_alpha_**. Also contained in X.FCN. Use αVIEW followed by PSE for message output in programs.
αXEQ	**h** P.FCN αXEQ **nn**	\α	Takes the contents of **R**_nn_ as character code. Interprets the first three characters (or less if there are only less) of the converted code as an alpha label and calls or executes the respective routine.
α → x	**f** x◄►a	All	Returns the character code of the leftmost character in **_alpha_** and deletes this character, like ATOX in *HP-42S*.
β	**h** STAT β	DECM	Returns Euler's Beta $B(x, y) = \dfrac{\Gamma(x) \cdot \Gamma(y)}{\Gamma(x + y)}$ with $\mathrm{Re}(x) > 0$, $\mathrm{Re}(y) > 0$. Called β here for avoiding ambiguities. Also contained in X.FCN.
Γ	**h** STAT Γ	DECM	Returns $\Gamma(x)$. This function is also contained in X.FCN. Additionally, **h** ! calls $\Gamma(x + 1)$.
ΔDAYS	**h** X.FCN ΔDAYS	DECM	Assumes **X** and **Y** containing dates in the format chosen and calculates the number of days between them. Works like in *HP-12C*.
Δ%	**f** Δ%	DECM	Returns $100 \cdot \dfrac{x - y}{y}$ like %CH in *HP-42S*.

Name	Keys to press	in modes	Remarks
ε	**h** **STAT** ε	DECM	Calculates the scattering factors (or geometric standard deviations) $$\ln(\varepsilon_y) = \sqrt{\frac{\sum \ln^2(y) - 2n \cdot \ln(\bar{y}_G)}{n-1}} \quad \text{and} \quad \ln(\varepsilon_x)$$ and pushes them on the stack. ε works for the geometric mean in analogy to s for the arithmetic mean but <u>multiplicative</u>.
ε$_m$	**h** **STAT** ε$_m$	DECM	Works like ε but pushes the scattering factors of the geometric means $\varepsilon_m = \varepsilon^{1/\sqrt{n}}$ on the stack.
ε$_p$	**h** **STAT** ε	DECM	Works like ε but with a denominator **n** instead of **n – 1**, returning the scattering factors of the populations.
ζ	**h** **X.FCN** ζ	DECM	Returns Riemann's Zeta function for real arguments, with $\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x}$ for **x > 1** and its analytical continuation for **x < 1**: $$\zeta(x) = 2^x \pi^{x-1} \sin\left(\frac{\pi}{2} x\right) \cdot \Gamma(1-x) \cdot \zeta(1-x).$$
π	**h** **π**	DECM	Complex version copies π in **X** and clears **Y**.
Π	**f** **π** *label*	DECM	Computes a product with the routine specified by *label*. 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 computes a factor. At its end, this factor is multiplied with said product; the operation then decrements `ccccccc` by `ii` and runs said routine again if then `ccccccc > fff`, else returns the resulting product in **X**.
Σ	**g** **Σ** *label*	DECM	Computes a sum with the routine specified by *label*. 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 computes a summand; at its end, this is added to said sum; the operation then decrements `ccccccc` by `ii` and runs said routine again if then `ccccccc > fff`, else returns the resulting sum in **X**.
σ	**h** **STAT** σ	DECM	Works like s but returns the standard deviations of the populations instead.

Name	Keys to press	in modes	Remarks
$\Sigma\ln^2 x$ $\Sigma\ln^2 y$ $\Sigma\ln x$ $\Sigma\ln xy$ $\Sigma\ln y$ $\Sigma x\ln y$ $\Sigma y\ln x$	**h** **STAT** $\Sigma\ln^2 x$ etc.	DECM	Recall the respective statistical sums. These sums are necessary for curve fitting models beyond pure linear. Calling them by name enhances readability of programs significantly.
σw	**h** **STAT** σw	DECM	Works like sw 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 Σx^2 $\Sigma x^2 y$ Σxy Σy Σy^2	**h** **STAT** Σx etc.	DECM	Recall the respective statistical sums. These sums are necessary for basic statistics and linear curve fitting. Calling them by name enhances readability of programs significantly.
Σ+	**h** **Σ+**	DECM	Adds a data point to the statistical sums.
	A	DECM	Shortcut as long as label A is not defined yet.
Σ−	**h** **Σ−**	DECM	Subtracts a data point from the statistical sums.
$\varphi(x)$	**h** **PROB** $\varphi(x)$	DECM	Standard normal pdf: $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.
$\Phi(x)$	**f** **Φ**	DECM	Standard normal cdf $\Phi(z) = \int_{-\infty}^{z} \varphi(x)dx$, equals $1 - Q$ in *HP-32E* and $1 - Q(z)$ in *HP-21S* with $z = x$.
$\Phi^{-1}(p)$	**g** **Φ^{-1}**		
χ^2 χ^2INV	**h** **PROB** χ^2 etc.	DECM	Chisquare distribution. The cdf χ^2 (with the degrees of freedom given in **J**) equals $1 - Q(\chi^2)$ in *HP-21S*.
(-1)^X	**h** **X.FCN** (-1)^X	\α	For x not being a natural number, this function will return $\cos(\pi \cdot x)$.

Name	Keys to press	in modes	Remarks
+	⊞	\α	Returns $y + x$.
−	⊟	\α	Returns $y - x$.
×	⊠	\α	Returns $y \cdot x$.
/	⟋	\α	Returns y / x .
+/−	+/−	\α	Unary minus like CHS in *HP-35*.
→DEG	→ (g) DEG	DECM	Takes x as an angle in the angular mode currently set and converts it to degrees. Prefix g may be omitted.
→GRAD	→ (g) GRAD	DECM	Like →DEG, but converts to gon or grads.
→H	→ f H.d	DECM	Takes x as hours or degrees in the format `hhhh.mmssdd` and converts them into a decimal time or angle.
→H.MS	→ f H.MS	DECM	Takes x as decimal hours or degrees and converts them into `hhhh.mmssdd` as in vintage HPs. For calculations, use H.MS+ or H.MS– then or reconvert to decimal values before.
→POL	g R◄►P	DECM	Assumes **X** and **Y** containing 2D Cartesian coordinates (x, y) and converts them to the respective polar coordinates (r, ϑ) with the radius $r = \sqrt{x^2 + y^2}$
→RAD	→ (g) RAD	DECM	Works like →DEG, but converts to radians.
→REC	f R◄►P	DECM	Assumes **X** and **Y** containing 2D polar coordinates (r, ϑ) and converts them to the respective Cartesian coordinates (x, y) .
%	g %	DECM	Returns $\dfrac{x \cdot y}{100}$, leaving **Y** unchanged.
%MG	h X.FCN h % MG	DECM	Returns the margin [17] $100 \cdot \dfrac{x - y}{x}$ in % for a price x and cost y , like %MU-Price in *HP-17B*.

---

[17] Margin corresponds to „Handelsspanne" in German.

Name	Keys to press	in modes	Remarks
%MRR	**h** (X.FCN) **h** (%) MRR	DECM	Returns the mean rate of return in percent per period, i.e. $100 \cdot \left[\left(\dfrac{x}{y} \right)^{\frac{1}{z}} - 1 \right]$ with x = FV = future value after z periods, y = PV = present value. For $z = 1$, $\Delta\%$ returns the same result easier.
%T	**h** (X.FCN) **h** (%) T	DECM	Returns $100 \cdot \dfrac{x}{y}$, interpreted as % of total.
%Σ	**h** (STAT) **h** (%) Σ	DECM	Returns $100 \cdot \dfrac{x}{\sum x}$. Also contained in X.FCN.
%+MG	**h** (X.FCN) **h** (%) +MG	DECM	Calculates a sales price $y/(1 - 0.01 \cdot x)$ by adding a margin of x % to the cost y, as %MU-Price does in *HP-17B*.
√	**f** (√x̄)	\α	
√	**D**	\α, \14, \15, \h	Shortcut working as long as label D is not defined yet.
∫	**g** (∫) *label*	DECM	Integrates 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 *HP-15C*.
∞?	**h** (TEST) ∞?	\α	Tests x for infinity.
//	**g** (||)	DECM	Returns $\left(\dfrac{1}{x} + \dfrac{1}{y} \right)^{-1}$.

Alphanumeric input:

Character	Keys to press	in modes	Remarks
°	⟨.⟩	DECM	Separates degrees or hours from minutes and seconds, so input format is hhhh.mmssdd. The user has to take care where an arbitrary real number represents such an angle or time.
0 … 9	⟨0⟩ … ⟨9⟩	\α	Standard numeric input. For integer bases <10, input of illegal digits throws an *error message*.
		in addressing	Register input. See the *tables* above for more.
	⟨0⟩, ⟨1⟩, f ⟨2⟩, … , f ⟨9⟩	α	Appends the respective digit to *alpha*.
A … F	⟨A⟩ … ⟨F⟩ (red print)	11, 12, 13, 14, 15, h	Numeric input for digits >10. See page 6 for more information.
A … Z	⟨A⟩ … ⟨Z⟩ (red print)	in addressing	Register input. See the *addressing tables* above for the letters applicable.
		α	Alphabetic input. See page 7 for more.
E	⟨E⟩ (the key)	DECM & \FRACT	Like EEX in the older vintage calculators.
(or)	f ◄() / g ()►	α	Appends a left / right parenthesis to *alpha*.
/	Second ⟨.⟩	DECM	A persistent 2nd ⟨.⟩ in input switches to fraction mode and will be interpreted as explained below. Please note you cannot enter ⟨E⟩ after you entered ⟨.⟩ twice – but you may delete the 2nd dot while editing the input line.
		FRC	First ⟨.⟩ is interpreted as a space, 2nd as a fraction mark. E.g. input of ⟨2⟩⟨.⟩⟨3⟩⟨.⟩⟨4⟩ results in 2 ¾ in the display. Improper fractions may be entered starting with a ⟨.⟩ , e.g. ⟨.⟩⟨3⟩⟨.⟩⟨2⟩ .
,	h ⟨./.⟩ ⟨XEQ⟩	α	Appends a comma to *alpha*.
.	f ⟨.⟩	α	Appends a point to *alpha*.
'.' or ','	⟨.⟩	DECM	Inserts a radix mark as selected.

NON-PROGRAMMABLE CONTROL, CLEARING AND INFORMATION COMMANDS

Keys to press	in modes	Remarks
▲ / ▼	All	These two navigation keys will repeat with 5Hz when held down for longer than 0.5s.
	Status open	Goes to previous / next set of flags.
	Catalog open	Goes to previous / next item in this catalog.
	α	Scrolls the display window six characters to the left / right in *alpha* if possible. If less than six characters are beyond the limits of the display window on the left / right side, the window will be positioned to the beginning / end of string. Useful for longer strings.
	Else	Acts like BST / SST in *HP-42S*.
⬅	Input pending	Deletes the last digit or character put in.
	α	Deletes the rightmost character in *alpha*.
	PRG	Deletes current step.
	Else	Acts like CLx.
f ◀ / g ▶	Integer	Shifts the display window to the left / right like in *HP-16C*. Helpful while working with small bases.
f ⬆	α	Toggles upper and lower case.
h X.FCN CLALL	\PRG	Clears all registers and programs if confirmed.
h CLP	\α	Clears the program memory after confirmation.
ENTER↑	Catalog open	Selects the current item like XEQ below.
	α	Turns alpha mode off.
EXIT	Catalog open	Leaves the catalog without executing anything.
	Input pending	Cancels the execution of pending operations, returning to the calculator status as it was before.
	\PRG & program running	Stops the running program like R/S . See below.
	PRG	Leaves programming mode like h P/R . See below.
	α	Turns alpha mode off like ENTER↑ . See above.
	Else	Does nothing.
g OFF	All	Turns calculator off.
ON	Calculator off	Turns calculator on.

Keys to press	in modes	Remarks
h **P/R**	\α	Toggles programming mode for keyboard entry.
h **X.FCN** RESET	All	Executes CLALL and resets all modes to start-up default, i.e. 24h, 2COMPL, ALL, DEG, DENANY, DENMAX 9999, DECM, LinF, PROFRC, RDX., SCIOVR, SSIZE4, WSIZE 64, Y.MD.
R/S	\PRG, \α	Stops the running program immediately. If there is none running, runs the current program or resumes its execution starting with the current step. Compare the programmable command STOP.
h **SHOW**	DECM & \PRG	Shows the full mantissa until the next key is pressed.
	PRG	Displays a CRC checksum of program memory contents, allowing validation of program integrity.
h **STATUS**	\PRG	Shows the status of all user flags, similar to STATUS on *HP-16C*. See *above*.
h **X.FCN** VERS	\PRG	Shows the firmware version and build number.
XEQ	Catalog open	Selects the item currently displayed and exits, executing the respective command. See *above*.
f **α**	\α	Turns on alpha mode for keyboard entry. When entering alpha constants in programs, please note there is no concatenation character – added characters are appended to *alpha* always. For starting a new string, use CLα first. Alpha constants will be listed like e.g. 'Test 1'.
→ **f** **2** **→** **f** **10** **→** (**g**) **16** **→** (**g**) **8**	\α	These commands show x in target integer representation until the next key is pressed. Base is kept as set. Prefix **g** may be omitted here. If used in integer bases 15 and 16, prefix **f** must precede the key **→**

CATALOGS

A catalog on the WP 34S is a collection of items, e.g. operations or characters. Such catalogs may be called using the keystrokes listed below:

Keys to press	in modes	Contents of said catalog
h **CAT**	\α	Predefined alpha labels. Some special rules apply here: **▲** and **▼** browse the catalog as usual, but in the numeric line the location of the respective label is indicated (RAM, Lib for XROM, or SEG *n* for flash memory segment n). **0** – **2** go to the alpha label currently displayed, in the flash segment specified. **ENTER** goes to the alpha label as displayed, while **XEQ** or **R/S** execute it. These keystrokes will perform a label search, starting at the current position of the program pointer. Search order is RAM, XROM, flash segments. **.** goes to the first alpha label in XROM. **◄** or **EXIT** leave CAT returning to the state as before.
h **CONST**	DECM	Constants like in HP35s. Picking a constant will push it onto the stack. See the constants listed in a *table below*.
CPX **CONST**	DECM	This catalog contains the same constants. Picking one, however, does a complex recall here. So, if the stack was [x, y, …] before, it will look like [*constant*, **0**, x, y, …] thereafter.
h **CONV**	DECM	Conversions as listed in a *table below*.
f **CPX**	α	"Complex" letters mandatory for many languages. Case is determined by setting (see **f** **↑** above).
h **MODE**	\α	Mode setting functions.
	Integer	Only a small subset of these functions makes sense in integer modes.
h **PROB**	DECM	Extra probability distributions.
h **P.FCN**	\α	Extra programming functions.
f **R↓**	α	Subscripts.
h **R↑**	α	Superscripts.
h **STAT**	DECM	Extra statistical functions.
h **TEST**	\α	All tests except the two on the keyboard.
	α	Comparison symbols and brackets, except **f** **(** and **g** **)**.

Keys to press	in modes	Contents of said catalog
h (X.FCN)	DECM	Extra real functions.
	Integer	Extra integer functions.
	α	Extra alpha functions.
(CPX) (X.FCN)	DECM	Extra complex functions.
h (./,)	α	Punctuation marks and text symbols.
f (➡)	α	Arrows and mathematical symbols.

Opening a catalog will set alpha mode to allow for typing the first character(s) of the item wanted. A subset of the full alpha keyboard shown *above* is sufficient for browsing:
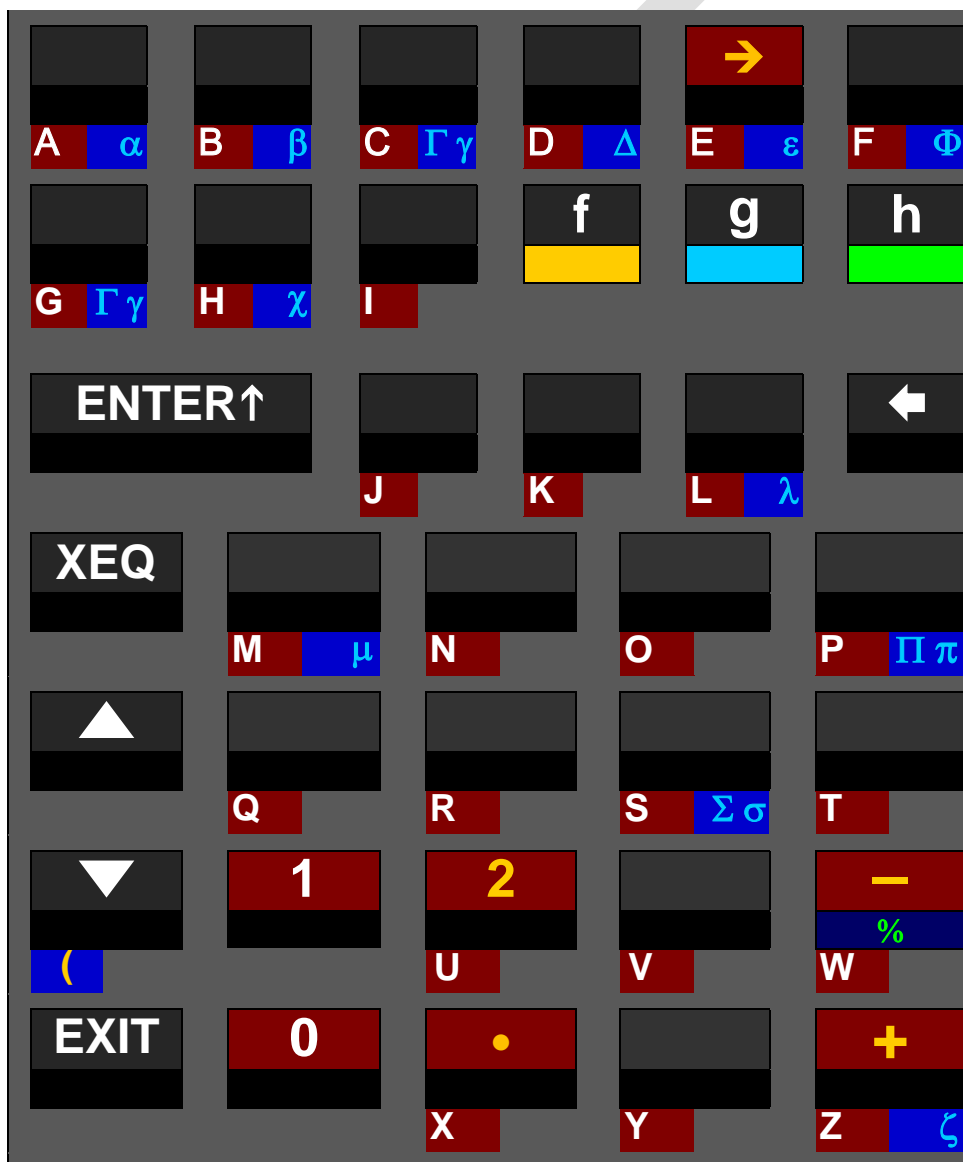
Please note **f** (➡) will just call the character → while browsing a catalog.

(▲) and (▼) browse the catalog, (ENTER↑) or (XEQ) select the item shown, recall or execute it and exit the catalog, while (EXIT) will just leave it without executing anything, returning to the mode as set before.

See *below* for some examples.

Reopening the very last catalog called, the last command selected therein is displayed for easy repetitive use. See the *table below about addressing cataloged items*, and the next pages for detailed item lists.

Within these lists, the characters necessary to access a specific function from an arbitrary position in the respective catalog are printed bold. Where a character is printed **grey** it will be faster employing ▼ to get to this function. E.g. for addressing Logis^{-1} , press ⓛⓞ ▼ ▼.

A single function, e.g. CB, may be contained in more than one catalog.

The alpha catalogs are found three pages below. See also CONST and CONV in separate paragraphs further below.

Catalog contents in detail:

(MODE)	(PROB)	(STAT)		(TEST)	(P.FCN)	
12h	**B**inom	**B**estF	σ	**B**C?	**B**ACK	**R**-CLR
1COMPL	**B**inom$_P$	**C**OV	$\Sigma\ln^2x$	**B**S?	**C**F	**R**-**C**OPY
24h	**B**inom $^{-1}$	**Ex**pF	$\Sigma\ln^2y$	**E**NTRY?	**CL**FLAG	**R**-**S**ORT
2COMPL	**C**auch	**L**inF	Σlnx	**E**VEN?	**CL**STK	**R**-**SW**AP
BASE	**C**auch$_P$	**LN**β	Σlnxy	**F**C?	**D**ATE	**S**F
DENANY	**C**auch $^{-1}$	**LN**Γ	Σlny	**FC**?C	**DE**C	**SK**IP
DENFAC	**E**xpon	**Lo**gF	σw	**FC**?F	**DR**OP	**ST**OM
DENFIX	**Ex**pon $_P$	**n**Σ	Σx	**FC?S**	**DS**Z	**ST**OS
DENMAX	**Ex**pon $^{-1}$	**P**owerF	Σx^2	**F**P?	**E**RR	**T**ICKS
DISP	**F**(x)	**S**EED	Σx^2y	**F**S?	**F**F	**T**IME
D.MY	**F** $^{-1}$(p)	**SE**RR	Σxlny	**FS**?C	**f** '(x)	**X**EQα
E3OFF	**G**eom	**SE**RRw	Σxy	**FS?F**	**f** ''(x)	α**G**TO
E3ON	**Ge**om $_P$	**SU**M	Σy	**FS?S**	**G**TOα	α**O**FF
FRACT	**Ge**om $^{-1}$	**sw**	Σy^2	**I**NT?	**H**.MS+	α**ON**
LZOFF	**L**gnrm	**sx**y	Σylnx	**K**EY?	**H**.MS-	α**X**EQ
LZON	**Lg**nrm $_P$	**x̄**g	%Σ	**L**BL?	**I**NC	α**V**IEW
M.DY	**Lg**nrm $^{-1}$	**x̄**w		**LE**AP?	**IS**Z	
RDX,	**Lo**gis	x̂		**N**aN?	**N**OP	
RDX.	**Lo**gis$_P$	β		**O**DD?	**O**FF	
SETCHN	**Lo**gis $^{-1}$	Γ		**P**RIME?	**P**ROMPT	
SETEUR	**N**orml	ε		**S**PEC?	**R**CLM	
SETIND	**N**orml $_P$	ε$_m$		**SS**IZE?	**R**CLS	
SETUK	**N**orml $^{-1}$	ε$_p$		**W**SIZE?	**RD**X,	
SETUSA	**P**oiss			**x** < ?	**RD**X.	
SIGNMT	**P**oiss$_P$			**x** ≤ ?	**RT**N+1	
SSIZE4	**P**oiss $^{-1}$			**x** ≈ ?		
SSIZE8	**t**(x)			**x** ≥ ?		
UNSIGN	**t** $^{-1}$(p)			**x** > ?		
WSIZE	**W**eibl			∞?		
Y.MD	**W**eibl $_P$					
	Weibl $^{-1}$					
	φ(x)					
	χ^2					
	χ^2 INV					

[X.FCN] varies with the mode set; it contains in … **[CPX][X.FCN]**

… alpha mode:	… decimal mode:		… integer modes:		^{C}AGM column (CPX)
CLALL	**A**GM	**M**AX	**A**SR	**R**CF	^{C}**A**GM
CLREG	**AN**GLE	**MI**N	**B**ATT	**RE**SET	^{C}**C**ONJ
RESET	**B**ATT	**N**AND	**C**B	**R**J	^{C}**CU**BE
VERS	**B**$_n$	**N**OR	**CL**ALL	**R**L	^{C}**CU**BERT
α**D**ATE	**B**$_n$*	**P**$_n$	**CLF**LAG	**RL**C	^{C}**D**ROP
α**D**AY	**C**EIL	**PR**CL	**CL**REG	**RR**	^{c}**e**^{x}-1
α**I**P	**CL**ALL	**PS**TO	**CU**BE	**RRC**	^{C}**FIB**
α**L**ENG	**CL**REG	**P**↔	**CU**BERT	**RR**CL	^{C}**LN**1+x
α**M**ONTH	**CU**BE	**R**AD→	**D**BLR	**S**AVE	^{C}**LN**β
α**R**C#	**CU**BERT	**R**CF	**D**BL*	**S**B	^{C}**LN**Γ
α**R**L	**D**ATE	**RE**SET	**D**BL/	**SE**ED	^{C}**R**CF
α**R**R	**D**AY	**RO**UNDI	**F**B	**SI**GN	^{C}**S**IGN
α**S**L	**D**AYS+	**RR**CL	**F**IB	**S**L	^{C}**SI**NC
α**S**R	**DE**COMP	**S**AVE	**G**CD	**S**R	^{C}**W**
α**T**IME	**DE**G→	**SE**TDAT	**L**CM	**SR**CL	^{C}**W**$^{-1}$
	D→J	**SE**TTIM	**L**J	**V**ERS	^{c}**β**
	erf	**SI**GN	**LO**AD	**X**NOR	^{c}**Γ**
	erfc	**SI**NC	**M**ASKL	α**I**P	^{c}**(-1)**^{x}
	e^x-1	**SL**VQ	**M**ASKR	α**L**ENG	
	FIB	**SR**CL	**M**AX	α**R**CL	
	FLOOR	**T**$_n$	**MI**N	α**R**C#	
	GCD	**TV**M	**MIR**ROR	α**R**L	
	GRAD→	**U**$_n$	**N**AND	α**R**R	
	H$_n$	**V**ERS	**nB**ITS	α**S**L	
	H$_{np}$	**W**	**N**OR	α**S**R	
	Iβ	**W**$^{-1}$	**P**RCL	α**ST**O	
	IΓ	**X**NOR	**PS**TO	α**V**IEW	
	JG1582	α**D**ATE	**P**↔	**(-1)**^{x}	
	JG1752	α**D**AY			
	J→D	α**I**P			
	LCM	α**L**ENG	α**S**R	ζ	
	L$_n$	α**M**ONTH	α**ST**O	**(-1)**^{x}	
	LN1+x	α**R**CL	α**T**IME	**%**MG	
	L$_n$α	α**R**C#	α**V**IEW	**%M**RR	
	LNβ	α**R**L	β	**%T**	
	LNΓ	α**R**R	Γ	**%Σ**	
	LOAD	α**S**L	ΔDAYS	**%+**MG	

CPX					
À			à		
Á			á		
Â Ã Ā Ă			â ã ā ă		
Ä			ä (ă)		
Å			å		
Ć			ć		
Č			č		
Ç			ç		
È			è		
É			é		
Ê Ē Ĕ Ě			ê ē ĕ ě		
Ë			ë (ĕ)		
			ħ		
Ì			ì		
Í			í		
Î Ĩ Ī Ĭ			î ĩ ī ĭ		
Ï			ï (ĭ)		
Ñ Ň			ñ ň		
Ò			ò		
Ó			ó		
Ô Õ Ō Ŏ			ô õ ō ŏ		
Ö			ö (ŏ)		
Ø			ø		
Ř			ř		
Š			š		
			ß		
Ù			ù		
Ú			ú		
Û Ũ Ū Ŭ			û ũ ū ŭ		
Ü			ü (ŭ)		
Ů			ů		
Ý			ý		
Ÿ			ÿ		
Ž			ž		

Here are the contents of the alpha catalogs making the WP 34S the most versatile global calculator known. Large font is printed in left column or upper row, small font in right column or lower row. Accented letters show the same width as plain ones wherever possible.

⌐./.	
, ? : ; ' " # * @ _ ~ `	

TEST	
< ≤ = ≈ ≥ > [ ] { }	

➡	
→ ← ↑ ↓ ∫ ∞ ^	

R↓ (subscripts)	
0 1 2 A B c e k m n p u μ ∞	

R↑ (superscripts)	
c ° 2 X x̄ x̂ ȳ ŷ ⁻¹	

The letters provided in the WP 34S allow for correct writing the languages of more than $3 \cdot 10^9$ people (still only half of mankind yet), i.e.:

Afrikaans, Català, Cebuano, Česky, Cymraeg, Deutsch, Eesti, English, Español, Euskara, Français, Gaeilge, Galego, Bahasa Indonesia, Italiano, Basa Jawa, Kiswahili, Kreyòl ayisyen, Magyar, Bahasa Melayu, Nederlands, Português, Quechua, Shqip, Slovenčina, Slovenščina, Basa Sunda, Suomeksi, Svenska, Tagalog, Winaray, Zhōngwén (with a little trick explained below), and almost Dansk and Norsk (sorry, no æ) as well as Hrvatski and Srpski (no đ). If you know further living languages covered, please tell us.

Mandarin Chinese (Zhōngwén) features four tones, usually transcribed like e.g. mā, má, mǎ, and mà. So you need different letters for ā and ă here, and for e, i, o, and u as well. With 6 pixels total character height we found no way to display these in both fonts nicely, keeping letters and accents separated for easy reading. For an unambiguous solution, we suggest using a dieresis (else not employed in Hànyǔ pīnyīn) representing the third tone here. Pinyin writers, we ask for your understanding.

ADDRESSING CATALOG ITEMS

1	User input	`CONST`, `CONV`, `MODE`, `PROB`, `P.FCN`, `STAT`, `TEST`, or `X.FCN`	`CPX`, `R↓`, or `R↑` in alpha mode	`→`, `TEST`, or `./,` in alpha mode
	Dot matrix display	**Shows 1ˢᵗ item in selected catalog.** (e.g. `BC?` in `P.FCN`) Alpha mode is set.	(e.g. `Á` in `CPX`)	(e.g. `,` in `./,`)
2	User input	`XEQ`, `▼`, `▲`, `EXIT`, or 1ˢᵗ character (e.g. `F`)	`XEQ`, `▼`, `▲`, `EXIT`, or character (e.g. `O`)	⬇
	Dot matrix display	**Shows 1ˢᵗ item starting with this character *)** (e.g. `FB`)	**Shows 1ˢᵗ item starting with this letter *)** (e.g. `Ó`)	
3	User input	`XEQ`, `▼`, `▲`, `EXIT`, or 2ⁿᵈ character (e.g. `S`)	⬇	⬇
	Dot matrix display	**Shows 1ˢᵗ item starting with this sequence *)** (e.g. `FS?`)		
4	User input	`XEQ`, `▼`, `▲`, or `EXIT` (e.g. `▼`)		
	Dot matrix display	**Shows next item in this catalog** (e.g. `FS?C`)	(e.g. `Ò`)	(e.g. `?`)
…		Continue browsing this way until reaching the item desired		
		(e.g. `FS?F`).	(e.g. `Ö`).	(e.g. `:`).
n	User input	`XEQ`		
		Calculator leaves the catalog returning to the mode set before		
	Dot matrix display	… and executes or inserts the command chosen, or recalls the constant selected. **Result**	… and appends the selected character to *alpha*. **Contents of alpha register** (e.g. `Östl. Seite:`)	

*) If a character or sequence specified is not found in this catalog then the first item following alphabetically will be shown. If there is no such item, then the last item in this catalog is displayed. You may key in even more than two characters – after 3 seconds, however, or after `▼` or `▲`, the search string will be reset and you may start with a first character again.

CONSTANTS

Below you find the contents of the catalog CONST. Navigation works as in the catalogs mentioned before. Values of physical constants (*incl. their relative standard deviations given in parentheses below*) are from CODATA 2006, copied in August 2010. Green background denotes exact or almost exact values. The more the color turns to red, the less precise the respective constant is known [18].

For the units, remember Tesla with $1T = 1\dfrac{Wb}{m^2} = 1\dfrac{V \cdot s}{m^2}$, Joule with $1J = 1N \cdot m = 1\dfrac{kg \cdot m^2}{s^2}$

and on the other hand $1J = 1W \cdot s = 1V \cdot A \cdot s = \dfrac{1}{e}eV \approx 6.24 \cdot 10^6 TeV$. Thus $1\dfrac{J}{T} = 1A \cdot m^2$.

	Numeric value	Unit	Remarks
a	365.2425	d	Gregorian year (per definition)
a$_0$	5.2917720859E-11 *(6.8E-10)*	m	Bohr radius $= \dfrac{\alpha}{4\pi \cdot R_\infty}$
c	2.99792458E8	m/s	Vacuum speed of light (per definition)
c$_1$	3.74177118E-16 *(5.0E-8)*	$m^2 \cdot W$	First radiation constant $= 2\pi \cdot h \cdot c^2$
c$_2$	0.014387752 *(1.7E-6)*	$m \cdot K$	Second radiation constant $= hc/k$
e	1.602176487E-19 *(2.5E-8)*	C	Electron charge $= \dfrac{2}{K_J R_K} = \Phi_0 G_0$
eE	2.718281828459045…	1	Euler's e. Please note the letter e is used for the electron charge elsewhere in this table.
F	96485.3399 *(2.5E-8)*	$\dfrac{C}{mol}$	Faraday's constant $= e\,N_A$
g	9.80665	m/s^2	Standard earth acceleration (per definition)
G	6.67428E-11 *(1.0E-4)*	$\dfrac{m^3}{kg \cdot s^2}$	Newton's gravitation constant
G$_0$	7.7480917004E-5 *(6.8E-10)*	$1/\Omega$	Conductance quantum $= 2e^2/h = 2/R_K$ with the von Klitzing constant R_K = 25812.807557 Ω
g$_e$	2.0023193043622 *(7.4E-13)*	1	Landé's g-factor

---

[18] The bracketed values printed here for your kind attention allow you to compute the precision of results you may obtain using these constants. The procedure to be employed is called error propagation. It is often ignored, though essential for trustworthy results – not only in science. Please turn to respective texts before you believe in 4 decimals of a calculation result based on yardstick measurements.

	Numeric value	Unit	Remarks
h	6.62606896**E**-34 *(5.0E-8)*	*J s*	Planck constant
ħ	1.054571628**E**-34 *(5.0E-8)*		$= h/2\pi$
k	1.3806504**E**-23 *(1.7E-6)*	J/K	Boltzmann constant $= R/N_A$
l_p	1.616252**E**-35 *(5.0E-5)*	*m*	Planck length $= \sqrt{\hbar G/c^3} = t_p c$
m ...

	Numeric value	Unit	Remarks
h	6.62606896**E**-34 *(5.0E-8)*	*J s*	Planck constant
ħ	1.054571628**E**-34 *(5.0E-8)*		$= h/2\pi$
k	1.3806504**E**-23 *(1.7E-6)*	J/K	Boltzmann constant $= R/N_A$
l_p	1.616252**E**-35 *(5.0E-5)*	*m*	Planck length $= \sqrt{\hbar G/c^3} = t_p c$
m_e	9.10938215**E**-31 *(5.0E-8)*		Electron mass
m_n	1.674927211**E**-27 *(5.0E-8)*		Neutron mass
m_p	1.672621637**E**-27 *(5.0E-8)*	*kg*	Proton mass
M_p	2.17644**E**-8 *(5.0E-5)*		Planck mass $= \sqrt{\hbar c/G} \approx 22\mu g$
m_u	1.660538782**E**-27 *(5.0E-8)*		Atomic unit mass $= 10^{-3}\, kg / N_A$
m_μ	1.88353103**E**-28 *(5.6E-8)*		Muon mass
N_A	6.02214179**E**23 *(5.0E-8)*	$1/mol$	Avogadro's number
NaN			"not a number"
p_o	101325	*Pa*	standard atmospheric pressure (per definition)
q_p	1,8755459**E**-18 *(5.0E-5)*	*As*	Planck charge $= \sqrt{4\pi\varepsilon_0 \hbar c} \approx 11.7e$
R	8.314472 *(1.7E-6)*	$\dfrac{J}{mol\cdot K}$	Molar gas constant
r_e	2.8179402894**E**-15 *(2.1E-9)*	*m*	Classical electron radius $= \alpha^2 \cdot a_0$
R_∞	1.0973731568527**E**7 *(6.6E-12)*	$1/m$	Rydberg constant $= \dfrac{\alpha^2 m_e c}{2h}$
T_o	273.15	*K*	= 0°C, standard temperature (per definition)
t_p	5.39124**E**-44 *(5.0E-5)*	*s*	Planck time $= \sqrt{\hbar G/c^5} = \dfrac{l_p}{c}$
T_p	1.416785**E**32 *(5.0E-5)*	*K*	Planck temperature $= \dfrac{c^2}{k}\sqrt{\dfrac{\hbar c}{G}} = \dfrac{M_p c^2}{k} = \dfrac{E_p}{k}$
V_m	0.022413996 *(1.7E-6)*	m^3/mol	Molar volume of an ideal gas at standard conditions $= \dfrac{RT_0}{p_0}$

	Numeric value	Unit	Remarks
$\mathbf{Z_o}$	376.730313461…	Ω	Characteristic impedance of vacuum $= \sqrt{\dfrac{\mu_0}{\varepsilon_0}} = \mu_0 c$
α	7.2973525376**E**-3 *(6.8E-10)*	1	Fine-structure constant $= \dfrac{e^2}{4\pi\varepsilon_0 \hbar c} \approx \dfrac{1}{137}$
γEM	0.57721566490153286…	1	Euler-Mascheroni constant
γp	2.675222099**E**8 *(2.6E-8)*	$\dfrac{1}{s \cdot T}$	Proton gyromagnetic ratio $= \dfrac{2\mu_P}{\hbar}$
εo	8.854187817…**E**-12	$\dfrac{A \cdot s}{V \cdot m}$ or F/m	Electric constant, vacuum permittivity $= \dfrac{1}{\mu_0 c^2}$
λc	2.4263102175**E**-12 *(1.4E-9)*		Compton wavelength of the electron $= h/{m_e c}$
λcn	1.3195908951**E**-15 *(1.5E-9)*	m	Compton wavelength of the neutron $= h/{m_n c}$
λcp	1.3214098446**E**-15 *(1.9E-9)*		Compton wavelength of the proton $= h/{m_p c}$
μo	1.2566370614…**E**-6	$\dfrac{V \cdot s}{A \cdot m}$	Magnetic constant, also known as vacuum permeability $= 4\pi \cdot 10^{-7} \dfrac{V \cdot s}{A \cdot m}$ (per definition)
μB	9.27400915**E**-24 *(2.5E-8)*		Bohr's magneton $= e\hbar/{2m_e}$
μe	-9.28476377**E**-24 *(2.5E-8)*		Electron magnetic moment
μn	-9.6623641**E**-27 *(2.4E-7)*	J/T or $A \cdot m^2$	Neutron magnetic moment
μp	1.410606662**E**-26 *(2.6E-8)*		Proton magnetic moment
μu	5.05078324**E**-27 *(2.5E-8)*		Nuclear magneton $= e\hbar/{2m_p}$
μμ	-4.49044786**E**-26 *(3.6E-8)*		Muon magnetic moment
π	3.141592653589793…	1	
σB	5.6704**E**-8 *(7.0E-6)*	$\dfrac{W}{m^2 K^4}$	Stefan Boltzmann constant $= \dfrac{2\pi^5 k^4}{15 h^3 c^2}$
Φ	1.61803398874989485…	1	Golden ratio $= \dfrac{1+\sqrt{5}}{2}$

	Numeric value	Unit	Remarks
Φₒ	2.067833667**E**-15 *(2.5**E**-8)*	*V s*	Magnetic flux quantum $= \frac{h}{2e} = \frac{1}{K_J}$ with the Josephson constant $K_J = 4.83597891 \cdot 10^{14} \frac{Hz}{V}$
∞		1	Infinity (may the Lord of Mathematics forgive us calling this a constant)

UNIT CONVERSIONS

These are the contents of the catalog CONV [19]. Navigation works as in the other catalogs. The constant **Tₒ** may be useful for conversions, too; it is found in the *catalog CONST*. The conversion factors or divisors listed in this table for your information are user transparent in executing a conversion. Those printed on light green background apply exactly.

Conversion		Remarks	Class
°C→°F	* 1.8 + 32		Temperature
°F→°C	- 32) / 1.8		Temperature
°→G	/ 0.9	Converts to 'grads' or 'gon'	Angle
°→rad	* π / 180	Equals D→R	Angle
acres→ha	* 0.4046873	1 ha = 10^4 m²	Area
ar.→dB	10 * lg(R)	Amplitude ratio	Ratio
atm→Pa	* 1.01325**E**5		Pressure
AU→km	* 1.495979**E**8	Astronomic units	Length
bhp→W	* 745.6999	British horse power	Power
Btu→J	* 1055.056	British thermal units	Energy
cal→J	* 4.1868		Energy
cft→*l*	* 28.31685	Cubic feet	Volume
cm→inches	/ 2.54		Length
dB→ar.	$10^{R_{dB}/20}$		Ratio

---

[19] For most readers, many of the units appearing here may look obsolete at least. They die hard, however, in some corners of this world. For symmetry reasons, we may also add some traditional Indian and Chinese units. Anyway, this catalog provides the means to convert local to common units.

Conversion		Remarks	Class
dB→pr.	$10^{R_{dB}/10}$	Power ratio	Ratio
fathom→m	* 1.8288		Length
feet→m	* 0.3048		Length
flozUK→*ml*	* 28.41306	1 *l* = $^1/_{1000}$ m^3	Volume
flozUS→*ml*	* 29.57353		
galUK→ *l*	* 4.54609		
galUS→ *l*	* 3.785418		
G→°	* 0.9	Grads or gon	Angle
g→**o**z	/ 28.34952		Mass
G→**r**ad	* π / 200		Angle
g→**t**r.oz	/ 31.10348		Mass
ha→acres	/ 0.4046873	1 ha = 10^4 m²	Area
HP$_e$→W	* 746	Electric horse power	Power
inches→cm	* 2.54		Length
inHg→Pa	* 3386.389		Pressure
J→Btu	/ 1055.056		Energy
J→cal	/ 4.1868		Energy
J→**k**Wh	/ 3.6**E**6		Energy
kg→lb	/ 0.4535924		Mass
km→AU	/ 1.495979**E**8	Astronomic units	Length
km→*l.y.*	/ 9.460730**E**12	Light years	Length
km→**m**iles	/ 1.609344		Length
km→**nm**i	/ 1.852	Nautical miles	Length
km→**pc**	/ 3.085678**E**16	Parsec	Length
kWh→J	* 3.6**E**6		Energy
lbf→N	* 4.448222		Force
lb→kg	* 0.4535924		Mass
l.y.→km	* 9.460730**E**12	Light years	Length

Conversion		Remarks	Class
l →cft	/ 28.31685		Volume
l →**g**alUK	/ 4.54609	1 *l* = $^1/_{1000}$ m^3	Volume
l →**g**alUS	/ 3.785418		
mbar→Pa	* 100		Pressure
miles→km	* 1.609344		Length
m*l*→flozUK	/ 28.41306	1 *ml* = 1 cm^3	Volume
m*l*→flozUS	/ 29.57353		
mmHg→Pa	* 133.3224	1 torr = 1 mm Hg	Pressure
m→fathom	/ 1.8288		Length
m→**f**eet	/ 0.3048		Length
m→**y**ards	/ 0.9144		Length
nmi→km	* 1.852	Nautical miles	Length
N→lbf	/ 4.448222		Force
oz→g	* 28.34952	Ounces	Mass
Pa→atm	/ 1.01325**E**5	1 Pa = 1 N/m^2	Pressure
Pa→inHg	/ 3386.389		Pressure
Pa→mbar	/ 100		Pressure
Pa→mmHg	/ 133.3224		Pressure
Pa→psi	/ 6894.757		Pressure
Pa→torr	/ 133.3224		Pressure
pc→km	* 3.085678**E**16	Parsec	Length
pr.→dB	10 * lg(R)	Power ratio	Ratio
psi→Pa	* 6894.757	Pounds per square inch	Pressure
PS(hp)→W	* 735.4988	Horse power	Power
rad→°	* 180 / π	Equals R→D	Angle
rad→G	* 200 / π		Angle
s.tons→t	* 0.9071847	Short tons	Mass
tons→t	* 1.016047	Imperial tons	Mass
torr→Pa	* 133.3224	1 torr = 1 mm Hg	Pressure

Conversion		Remarks	Class
tr.oz→g	* 31.10348	Troy ounces	Mass
t→s.tons	/ 0.9071847	1 t = 10^3 kg	Mass
t→**t**ons	/ 1.016047		
W→bhp	/ 745.6999		Power
W→HP$_e$	/ 746		Power
W→**P**S(hp)	* 735.4988		Power
yards→m	* 0.9144		Length

In cases of emergency of a particular kind, remember Becquerel equals Hertz, Gray is the unit for deposited or absorbed energy ($1Gy = 1J/kg$), and Sievert is Gray times a radiation dependant dose conversion factor for the damage caused in human bodies.

In this area also some outdated units may be found in older literature: Pour les ami(e)s de Mme. Curie, $1Ci = 3.7 \cdot 10^{10} Bq = 3.7 \cdot 10^{10} decays/s$. And for those admiring the very first Nobel laureate in physics, Mr. Röntgen, for finding the x-rays (ruining his hands in these experiments), the charge generated by radiation in matter was measured by the unit $1R = 2.58 \cdot 10^{-4} As/kg$. A few decades ago, Rem (i.e. Röntgen equivalent men) was measuring what Sievert does today.

PREDEFINED GLOBAL ALPHA LABELS

There are a few labels employed and provided for particular tasks already. They are located in XROM, thus not taking any steps from user program memory. The following global labels are used:

TVM	Time Value of Money almost as known since the *HP-80*. This routine contains the equation

$$PMT - \frac{I}{k} \cdot \left[PV + \frac{PV + FV}{(1+I)^n - 1} \right] = 0 \quad \text{with}$$

PMT = periodic payment = *r80*,
PV = present value = *r81*,
FV = future value = *r82*,
I = interest rate per period = *r83*,
n = number of periods = *r84*,

k = 1 if payment is made at the end of the period = **flag 80** clear,
= 1 + *I* if it is made at the beginning of the period = **flag 80** set.

Store all you know and solve for the unknown. E.g. solving for *PMT* may look like:

```
LBL 'PMT'          ;routine is entered with a first guess in X.
    SLV 'PM1'
    NOP            ;this step must be included since SLV acts as a test.
    RTN

LBL 'PM1'
    STO 80         ;initial or previous guess
    XEQ 'TVM'
    RTN
```

See SLV for more.

WHO	Displays credits.

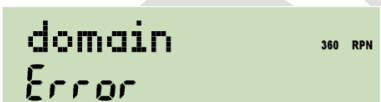| δx	Provides the step size for differentiation. See f '(x) and f "(x) for more information.
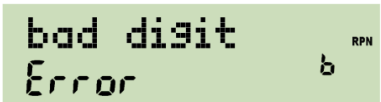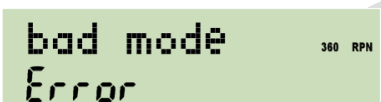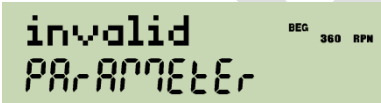
MESSAGES
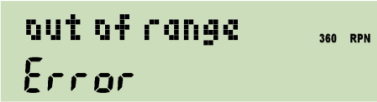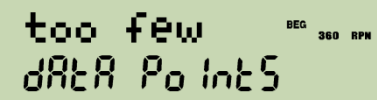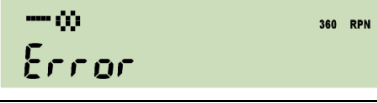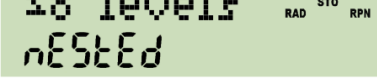
There are some commands generating messages, also in the dot matrix section of the display. Four of them, DAY, DAYS+, STATUS, and VERS, were introduced above in the *paragraph about display* already. Others are PROMPT, αVIEW and many more alpha commands, and the test commands as mentioned *above*.

Also two constants will return a special display when called: **NaN** and ∞ will show

`not nuПЕrI c`   or   `InF In Ity`   , respectively.

Furthermore, there are a number of error messages. Depending on error conditions, the following messages will be displayed in the mode(s) listed:

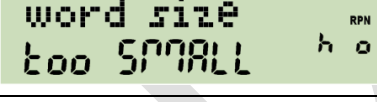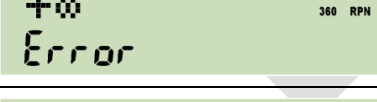| Message	Error Code	Mode(s)	Explanation and Examples
bad date Error	2	DECM	Invalid date format or incorrect date in input, e.g. month >12, day >31 etc.
bad digit Error b	9	Integer	Invalid digit in integer input, e.g. 2 in binary, 9 in octal, or +/- in unsigned mode.
bad mode Error	13	All	Caused by calling an operation in a mode where it is not defined, e.g. SIN in hexadecimal.
domain Error	1	\α	An argument exceeds the domain of the mathematical function called. May be caused by roots or logs of negative numbers (if not preceded by $\boxed{\text{CPX}}$), by $0/0$, LN(0), Γ(0), TAN(90°) and equivalents, ATANH(x) for $\lvert \mathrm{Re}(x) \rvert \ge 1$, ACOSH(x) for $\mathrm{Re}(x) < 1$, etc.
invalid PArAПЕtEr	16	\α	Similar to error 1 but a parameter specified in **J** or **K** is out of supported range for the function called. May appear e.g. if LgNrm is called with $j < 0$.
no such LAbEL	6	All	Attempt to address an undefined label.

Message	Error Code	Mode(s)	Explanation and Examples
out of range Error `360 RPN`	8	All	• A number exceeds the valid range. Caused e.g. by specifying decimals >11, word size >64, negative flag numbers, integers $\geq 2^{64}$, hours or degrees >9000, invalid times, denominators ≥9999 etc. • A register address exceeds the valid range. May also happen in indirect addressing. • An R-operation (e.g. R-COPY) attempts exceeding valid register numbers (0 .. 99).
SLV ∫ Σ Π nEStEd `RAD STO RPN`	7	PRG	Nested use of solve, integrate, sum or product is not allowed.
stack CLASH `BEG 360 RPN`	12	All	STOS or RCLS attempt using registers that would overlap the stack. Will happen with e.g. SSIZE = 8 and STOS 94.
too few dAtA PointS `BEG 360 RPN`	15	DECM	A statistical calculation was started based on too few data points, e.g. regression or standard deviation for < 2 points.
too long Error `360 RPN`	10	All	Keyboard input is too long for the buffer (should never happen, but who knows).
undefined OP-COdE `STO 360 RPN`	3	All	An instruction with an undefined op-code occurred (should never happen, but who knows).
word size too SMALL `RPN h o`	14	Integer, \PRG	Stack or register content is too big for the word size set.
+∞ Error `360 RPN`	4	\α, \PRG	• Division of a number > 0 (or < 0) by zero. • Divergent sum or product or integral. • Positive (or negative) overflow in DECM (see *above*).
−∞ Error `360 RPN`	5		
≥8 levels nEStEd `RAD STO RPN`	11	PRG	Subroutine nesting exceeds 8 levels.

Any key pressed will erase the error message displayed and execute with the stack contents present. Thus, the easiest return to the display shown before the error occurred is pressing a prefix twice.

APPENDIX A: SUPPORT COMMANDS

How to flash your HP-20b or -30b

You need a special cable and a PC featuring a serial interface. For further information, please turn to http://dl.dropbox.com/u/10022608/Flashing%20a%2020b%20Calculator.pdf edited by Tim Wessmann of HP.

Commands to handle the flash memory on the real calculator

With **ON** (i.e. the key [EXIT]) held down, press one of the following keys:

[B] for Backup: This creates a copy of the RAM in flash memory. The whole RAM is saved to address 0x11F800 (relative address 0x1F800). This may be copied to your PC or loaded from it

[R] (i.e. [5]) for Restore: This restores a previously created backup.

[S] (i.e. [6]) for SAM-BA: Clears the GPNVM1 bit and turns the calculator off. ATTENTION: You can now only boot into SAM-BA mode! Without the SAM-BA software and the cable, you're lost!

We recommend you backup before flashing a new release! After flashing, your backup should still be present if you didn't press the ERASE button on the cable but used ON+S instead to get into SAM-BA boot mode.

How to transfer data between the WP 34S calculator and your PC

1. From calculator to PC

   a) Do an ON+B, then ON+S, then RESET with the cable connected.

   b) Press ON once and start SAM-BA on the PC. It should connect.

   c) Set the start address to 0x11F800 and the size to 0x800

   d) Enter a file name in the "receive" field.

   e) You can now receive the file with SAM-BA.

   f) Move it under the name wp34s.dat into your emulator directory.

   g) The emulator should accept the file. Your registers and programs will then be in place.

   h) To get your calculator back in business, start the "Boot from flash" script in SAM-BA, the same procedure you know from flashing the firmware.

   i) Power the calculator up and perform a restore with ON+R.

2. From PC to calculator

   a) Perform steps (a) to (c) from above. No need to set the size.

   b) Point SAM-BA to your wp34s.dat file from the emulator.

   c) You can now send the short file with SAM-BA.

   d) Perform steps (h) and (i) from above.

As an option, you can concatenate the flash image file calc.bin with your wp34s.dat file and send the result to the calculator as a whole.

More keyboard commands employing ON (use at your own risk)

With **ON** (i.e. the key ⌊**EXIT**⌋) held down, press one of the following keys:

⌊**RCL**⌋ (i.e. ⌊**TIME**⌋):   Tells the system a quartz is installed for the real time clock. This is a hardware modification described elsewhere. ATTENTION: If this command is entered though the hardware does not contain said modification, the system will hang!

⌊**.**⌋ for toggling the radix mark as ⌊**./,**⌋ does.

Internal commands (use at your own risk)

Some commands are used in internal routines exclusively and are not accessible from the keyboard. They are listed here for sake of a complete documentation only:

Name	Purpose and remarks
iC **_n_**	Recalls internal constants, selected by the number specified:

0	0	
1	1	
2	5.01402	Kronrod only weight loop initializer (constants 5 - 14 below)
3	15.02903	Gauss-Kronrod weight loop initializer (constants 15 - 29 below)
		Midpoint location is 0.5.
4	0.14944555400291690566493646836389821	Kronrod weight for midpoint k10
5	0.99565716302580808073552728068900	Kronrod location of k0 and k20
6	0.01169463886737187427806439606219	Kronrod weight for k0 and k20
7	0.93015749135570822600120718005950	Kronrod location of k2 and k18
8	0.05475589965743519960313813002445	Kronrod weight for k2 and k18
9	0.78081772658641689706371757834504	Kronrod location of k4 and k16
10	0.09312545458369760553506546508336	Kronrod weight for k4 and k16
11	0.56275713466860468333900009927269	Kronrod location of k6 and k14
12	0.12349197626206585107795810983107	Kronrod weight for k6 and k14
13	0.29439286270146019813112660310386	Kronrod location of k8 and k12
14	0.14277593857706008079709427313871	Kronrod weight for k8 and k12
15	0.97390652851717172007796401208445	Location of g0, g9, k1 and k19
16	0.06667134430868813759356880989333	Gauss weight for g0 and g9
17	0.03255816230796472747881897245939	Kronrod weight for k1 and k19
18	0.86506336668898451073209668842349	Location of g1, g8, k3 and k17
19	0.14945134915058059314577633965769	Gauss weight for g1 and g8
20	0.07503967481091995276704314091619	Kronrod weight for k3 and k17
21	0.67940956829902440623432736511487	Location of g2, g7, k5 and k15
22	0.21908636251598204399553493422816	Gauss weight for g2 and g7
23	0.10938715880229764189921059032580	Kronrod weight for k5 and k15
24	0.43339539412924719079926594316578	Location of g3, g6, k7 and k13
25	0.26926671930999635509122692156946	Gauss weight for g3 and g6
26	0.13470921731147332592805400177170	Kronrod weight for k7 and k13
27	0.14887433898163121088482600112972	Location of g4, g5, k9 and k11
28	0.29552422471475287017389299465133	Gauss weight for g4 and g5
29	0.14773910490013384913748415159720	Kronrod weight for k9 and k11

Constants 2 .. 29 are for the 10 / 21 point Gauss-Kronrod quadrature used by the internal integration command. Locations are in the range (0, 1) which is scaled to match the interval of integration. The quadrature sums the weight times the function value at each location to

Name	Purpose and remarks
	estimate the integral. In Gauss-Kronrod schemes the Gauss points are common to both quadratures although the weights are different. This means two estimates of the integral can be performed without increasing the number of function evaluations which in turn allows an estimate of the error to be made. The cost for this is a reduction in the degree of polynomial function that is always integrated exactly.
	The two solver commands described below may use some hidden registers and flags. The start points of the respective register and flag blocks are passed as one argument *n*.

Registers:
    n+0 .. *n*+1: first two estimates *a* and *b* for the root
    n+2: third estimate *c*
    n+3: function value at first estimate **f(*a*)**
    n+4: function value at second estimate **f(*b*)**

Flags:
    n+0 .. *n*+7: an eight bit iteration counter
    n+8: "bracket flag" – true if we've got an interval with $f(a) * f(b) < 0$
    n+9: true if all function evaluations have been constant so far |
| SLVI **n** | Initializes the solver. SLVI clears the iteration counter, takes *a* and *b* and calculates **f(*a*)** and **f(*b*)**, sets the last 2 flags accordingly, and produces a guess *c*. There is no stack interaction. |
| SLVS **n** | Solver step. Updates the internal solver state based on the last function evaluation. In particular, SLVS takes *a*, *b*, *c*, **f(*a*)**, and **f(*b*)** from the register block plus **f(*c*)** from **X** and updates the register values so that *c* and **f(*c*)** replace one of *a* and **f(*a*)** or *b* and **f(*b*).** It also produces a new guess *c* and returns zero in **X** if the solving should continue and non-zero if not. Otherwise, the stack isn't altered. |
| | The built in solver loop looks like this in principle, assuming $n = 0$:

```
        SLVI           ; calculate f(a) and f(b) and initialize the registers and flags
  LBL 00
        RCL 02         ; recall c
        XEQUSR         ; call the user's subroutine calculating f(c)
        x≈ 0?          ; test if the solution has converged
        GTO 01         ; converged, so exit the routine
        SLVS           ; update estimates
        x= 0?          ; should we continue?
        GTO 00         ; loop back again
  LBL 01
        RCL 02         ; best guess so far
        RTN
```

The actual solver is fairly complex. A combination of quadratic interpolation and a guarded secant method is used. |
| XEQUSR | Calls a user subroutine (used by SLV, ∫, Π and Σ). The subroutine is defined by the argument to the initial command (either numeric of alpha label). |

APPENDIX B: CANDIDATES FOR FURTHER FUNCTIONS

If space allows, the following functions may be implemented easily since they are coded already. None of these are counting the catalog and function table overheads. Two bytes for a catalog entry (one for each catalog it is in) and 12-20 bytes for a function table entry (but only one of these), i.e. not terribly significant. These are all moderately useful functions.

Function name and remarks	Size	Domain

Bessel functions of first and second kinds:
    J_n & I_n: real and complex (argument and order);
    Y_n & K_n: real and complex (argument and order).
 	
    Remember: $J_n(x) = \sum_{r=0}^{\infty} \dfrac{(-1)^r \cdot \left(x/2\right)^{2r+n}}{r! \cdot \Gamma(n+r+1)}$
	4470 B	$\mathbb{R}, \mathbb{C}$

Digamma function (ψ, needed for Bessel functions of second kind of integer order)	1384 B	$\mathbb{R}, \mathbb{C}$

Fused multiply and add
    The real version can be replaced by complex multiply.
    x+y*z can be done via (y, x) * (z, -1) at a pinch.
	96 B	$\mathbb{Z}, \mathbb{R}$

Jacobi elliptic functions S_n, C_n & D_n	1780 B	$\mathbb{R}, \mathbb{C}$
x!!	288 B	$\mathbb{R}, \mathbb{C}$

PRIME? also includes overflow resistant code for *(a * b) mod c* and *(a ^ b) mod c* which could also be exposed if required.

APPENDIX C: RELEASE NOTES

	Date	Release notes
1	9.12.08	Start
1.1	15.12.08	Added the table of indicators; added NAND, NOR, XNOR, RCLWS, STOWS, //, N, SERR, SIGMA, < and >; deleted HR, INPUT, 2 flag commands, and 2 conversions; extended explanations for addressing and COMPLEX & …; put XOR on the keyboard; corrected errors.
1.2	4.1.09	Added ASRN, CBC?, CBS?, CCB, SCB, FLOAT, MIRROR, SLN, SRN, >BIN, >DEC, >HEX, >OCT, BETA, D>R, DATE, DDAYS, D.MY, M.DY, Y.MD, CEIL, FLOOR, DSZ, ISZ, D>R, R>D, EMGAM, GSB, LNBETA, LNGAMMA, MAX, MIN, NOP, REAL, RJ, W and WINV, ZETA, %+ and %-; renamed the top left keys B, C, and D, and bottom left EXIT.
1.3	17.1.09	Added AIP, ALENG, ARCL, AROT, ASHF, ASTO, ATOX, XTOA, AVIEW, CLA, PROMPT (all taken from 42S), CAPP, FC?C, FS?C, SGMNT, and the …# commands; renamed NBITS to BITS and STOWS to WSIZE; specified the bit commands closer; deleted the 4 carry bit operations.
1.4	10.2.09	Added CONST and a table of constants provided, D>J and J>D, LEAP?, %T, RCL and STO ▲ and ▼, and 2 forgotten statistics registers; deleted CHS, EMGAM, GSB, REAL and ZETA; purged and renamed the bit operations; renamed many commands.
1.5	5.3.09	Added RNDINT, CONV and its table, a memory table, the description of XEQ B, C, D to the operation index, and a and g_e to the table of constants; put CLSTK on a key, moved CLΣ and FILL, changed the % and log labels on the keyboard, put CLALL in X.FCN; checked and cleaned alpha mode keyboard and added a temporary alpha keyboard; rearranged the alphabet to put Greek after Latin, symbols after Greek consistently; separated the input and non-programmable commands; cleaned the addressing tables.
1.6	12.8.09	Added BASE, DAYS+, DROP, DROPY, E3OFF, E3ON, FC?F, FC?S, FIB, FS?F, FS?S, GCD, LCM, SETDAT, SETTIM, SET24, SINC, TIME, VERS, αDAY, αMONTH, αRC#; %Σ, as well as F-, t-, and χ^2-distributions and their inverses; reassigned DATE, modified DENMAX, FLOAT, αROT, and αSHIFT; deleted BASE arithmetic, BIN, DEC, HEX, and OCT; updated the alpha keyboards; added flags in the memory table; included indirect addressing for comparisons; added a paragraph about the display; updated the table of indicators; corrected errors.
1.7	9.9.09	Added P.FCN and STAT catalogs, 4 more conversions, 3 more flags, Greek character access, CLFLAG, DECOMP, DENANY, DENFAC, DENFIX, Iβ, IΓ, αDATE, αRL, αRR, αSL, αSR, αTIME, 12h, 24h, fraction mode limits, normal distribution and its inverse for arbitrary μ and σ, and Boolean operations working within FLOAT; deleted αROT, αSHIFT, the timer, and forced radians after inverse hyperbolics; renamed WINV to W^{-1}, and beta and gamma commands to Greek; added tables of catalog contents; modified label addressing; relabeled PRGM to P/R and PAUSE to PSE; swapped SHOW and PSE as well as Δ% and % on the keyboard; relabeled Q; corrected CEIL and FLOOR; updated X.FCN and alpha commands; updated the virtual alpha keyboard.
1.8	29.10.09	Added R-CLR, R-COPY, R-SORT, R-SWAP, RCLM, STOM, alpha catalogs, 1 more constant and some more conversions, a table of error messages, as well as the binomial, Poisson, geometric, Weibull and exponential distributions and their inverses; renamed some commands; put √ instead of π on hotkey D.
1.9	14.12.09	Added two complex comparisons; swapped and changed labels in the top three rows of keys, dropped CLST; completed function descriptions in the index.
1.10	19.1.10	Added IMPFRC, PROFRC, ᶜENTER, αBEG, αEND, and an addressing table for items in catalogs; updated temporary alpha mode, display and indicators, RCLM and STOM, alpha-commands and the message table; renamed the exponential distribution; wrote the introduction.
1.11	21.9.10	Changed keyboard layout to bring Π and Σ to the front, relabeled binary log, swapped the locations of π, CLPR, and STATUS, as well as SF and FS?; created a menu TEST for the comparisons removed and the other programmable tests from P.FCN; added %MG, %+MG, %MRR, RESET, SSIZE4, SSIZE8, SSIZE?, ᶜDROP, ᶜFILL, ᶜR↓, ᶜR↑, registers J and K, a table of contents and tables for stack mechanics and addressing in complex operations; updated memory and real number addressing tables, DECOMP, αOFF, αON, Π, and Σ; renamed ROUNDI, WSIZE?, β(x,y), Γ(x) and the constant p_0; deleted DROPY (use x↔y, DROP instead), αAPP, αBEG, αEND, and the "too long error" message; deleted Josephson and von Klitzing constants (they are just the inverses of other constants included in CONST already); brought more symbols on the alpha keyboard.
1.12	22.12.10	Modified keyboard layout; added catalogs MODE and PROB; changed mode word, catalog contents and handling (XEQ instead of ENTER), as well as some non-programmable info commands; expanded IMPFRC and PROFRC; added a paragraph about the fonts provided and explained alpha catalogs in detail; added PRIME? and some conversions; deleted FRACT, OFF and ON.
1.13	3.2.11	Modified keyboard layout; modified αTIME, radix setting, H.MS+ and H.MS-; added EVEN?, FP?, INT?, LZOFF, LZON, ODD?, RCLS, STOS, returned FRACT; added and renamed some conversions; updated the paragraph about display; added appendices A and B; baptized the device WP 34S.

1.14	18.3.11	Added DEC and INC, renamed FLOAT to DECM; redefined αTIME and H.MS mode; updated appendix A; documented the annunciators BEG and = as well as underflows and overflows in H.MS; corrected some errors showing up with the emulator.
1.15	21.3.11	Modified FIX, removed ALL from MODE, updated CONV.
1.16	27.3.11	Added LBL?, f'(x), and f''(x); modified PSE; upgraded catalog searching.
1.17	9.5.11	Modified keyboard layout for adding a fourth hotkey; added AGM, BATT, B_n, B_n*, Cauch, Lgnrm, Logis and their inverses, all the pdf, COV, CUBE, CUBERT, DEG→, ENGOVR, ENTRY?, erfc, GRAD→, GTO . *hotkey*, KEY?, RAD→, SCIOVR, SERRw, SLVQ, sw, sxy, TICKS, TVM, xg, ε, ε_m, ε_p, ζ, σw, $(-1)^X$, the polynomials, four angular conversions, four Planck constants, the regional settings, global alpha labels, and three messages; renamed most cdf; changed →DEG, →RAD, →GRAD to leaving angular mode as set; altered PSE for early termination by keystroke; made D.MY default instead of Y.MD; moved degrees to radians conversions to CONV; removed ^{C}CLx, H.MS mode, %+ and %-; corrected errors.
1.18	ff	Expanded program memory; modified label addressing (A ≠ 'A') and fraction mode limits, changed ANGLE to work in real and complex domains, renamed MOD to RMDR, changed // to ‖, moved VIEW to RCL and TIME to P.FCN; put BACK, ERR, SKIP, and SPEC? to the main index; added the I/O commands for flash memory; corrected x→α.