



OWNER'S MANUAL

TABLE OF CONTENTS

| | |
|--|----|
| Welcome..... | 2 |
| Print Conventions | 4 |
| Getting Started | 4 |
| Keyboard | 5 |
| Real and Integer Operations..... | 10 |
| Complex Operations..... | 10 |
| Memory..... | 12 |
| Stack Mechanics | 14 |
| Comparing and Addressing Real Numbers | 16 |
| Comparing and Addressing Complex Numbers..... | 17 |
| Addressing Labels | 18 |
| Statistical Distributions, Probabilities etc. | 19 |
| Display and Modes | 20 |
| Fonts..... | 26 |
| Index of Operations | 27 |
| Non-programmable Control, Clearing and Information Commands..... | 58 |
| Catalogs | 60 |
| Addressing Catalog Items..... | 66 |
| Constants | 67 |
| Unit Conversions | 72 |
| Predefined Global Alpha Labels | 76 |
| Messages | 77 |
| Programmed Input and Output | 79 |
| Appendix A: Support Commands | 80 |
| Appendix B: Release Notes..... | 85 |

JUST IN CASE ...

... you still have your *HP-20b Business Consultant* or your *HP-30b Business Professional* sitting on your desk unchanged as produced by HP, please turn to [Appendix A](#) for some instructions how to convert it into a full fledged WP 34S yourself. Alternatively, if you don't want to bother with connecting it to your computer, flashing the calculator firmware and attaching a sticky overlay, you may purchase e.g. a *HP-30b*-based WP 34S readily in the internet:

http://www.thecalculatorstore.com/epages/eb9376.sf/en_GB/?ObjectPath=/Shops/eb9376/Products/%22WP34s%20Pack%22

(We apologize for the small font – it allowed this hyperlink fitting into one print line).

The first way may just cost your time, the second will cost you some money at the store. If you choose buying your WP 34S at the address mentioned, we (the developers) will get a modest fraction of the price. Both ways, however, are proven to work – it is your choice.

For the following, we assume the repurposing is done and you hold a WP 34S in your hands.

WELCOME

Dear user, now you have got it: 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 benefit from their unexcelled processor speed. And with the *HP-30b* you get the famous rotate-and-click keys in addition, giving the tactile feedback appreciated in vintage *Hewlett-Packard* calculators for decades.

On the other hand, the firmware and user interface of the WP 34S were thoroughly thought through and discussed by us, newly designed and written from scratch, loaded with functions, pressed into the little memory provided, and tested over and over again to give you **a fast and compact scientific calculator like you have never had before.**

The WP 34S function set is based on the one of the famous *HP-42S RPN Scientific*, the most powerful programmable RPN calculator built so far¹. We expanded this set, incorporating the functionality of the renowned 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 numbers, Lambert's W (all of these in 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,

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.

- + financial operations like mean rate of return and margin calculations,
- + 80 conversions, mainly between universal SI and old Imperial units,
- + 50 fundamental physical constants as precise as known today by national standards institutes like NIST or PTB, plus some more out of mathematics, astronomy, and surveying,
- + complete Greek and extended Latin letter fonts covering many languages of this planet (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 four stack levels for HP compatibility, eight levels for convenient calculations in complex domain, advanced real calculus, vector algebra, or for whatever application you have in your mind. The WP 34S provides a full set of commands for navigation in either size.

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

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). From its beginning, our project was discussed on the open forum in the Museum of HP Calculators (www.hpmuseum.org), so we want to express our gratitude to all the international 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 the WP 34S to life, starting with an emulator for v1.14, allowing widespread use and convenient testing. From v1.17 on, it runs on the real hardware as well.

We baptized our baby WP 34S in honor of one of the most powerful LED pocket calculators, the *HP-34C* of 1979, 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* once. May the WP 34S help in convincing those having access to more resources than us: covering the market of serious scientific instruments is worthwhile.

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 free of bugs. We cannot guarantee this, however, nor can we bear any liability for errors in calculations nor their possible consequences. Nevertheless, we promise we will continue improving 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 got one ourselves. In the project so far, we did show short response times.

Enjoy!

Paul Dale and Walter Bonin

PRINT CONVENTIONS

Please note:

- Throughout this manual, standard font is Arial. *Specific terms, names or titles* are printed in italics. [Hyperlinks](#) are underlined. Bold italic letters like ***n*** are used for variables. Calculator commands – e.g. ENTER – are generally called by their names, printed in capitals for easy recognition. Each and every command featured is listed in the [Index of Operations](#) below.
- This **CPX** font is taken for explicit references to keys.
- Register addresses are printed using **bold** Times New Roman, while lower case italic letters of this font are employed for register contents. So, for example, *y* lives in stack level **Y**, *r45* in general purpose register **R45**, and *alpha* in the alpha register, respectively.

All this holds unless stated otherwise explicitly.

GETTING STARTED

If you know how to deal with a good old HP RPN scientific calculator, you can start with your WP 34S right away.

Else we recommend you get an *HP-42S Owner's Manual*. It is available at low cost on the DVD distributed by the *Museum of Hewlett-Packard Calculators* (www.hpmuseum.org). There are also other sources in the internet.

Please read Part 1 of said manual as a starter. This part includes an excellent introduction to RPN. This RPN is a very effective method making **(I)**, **(J)**, **(L)**, **(J)**, **(f)**, **(j)** and **(=)** keys obsolete in calculations. Once you got used to it you will most probably never employ a calculator featuring **(=)** again.

Part 2 of said manual will support you when you are heading for programming your WP 34S for easy handling of repeated or iterative computations. Further documentation, also about the other calculators mentioned above and in the following text, will add valuable information – it is all readily accessible on a single DVD from said source.

Most commands on your WP 34S will work as they did on the *HP-42S*. This little manual here is meant as a supplement showing you all the new features. It contains some formulas and technical explanations but is not intended to replace textbooks about mathematics, statistics, physics, programming, or the like.

The following text starts presenting the 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, addressing items therein, the display and indicators used to give you feedback what is 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.

KEYBOARD



Generally, white labels execute the *default primary function* of the respective key. There are further (secondary) functions provided for almost all keys. Their labels are printed next to the white ones in golden, blue, green or grey color.

Green labels are placed on the slanted faces of most keys. Golden and blue labels are printed below of the respective key on the top face of the WP 34S, i.e. on the *key plate*. Grey letters are put bottom left of most keys. Labels underlined open *catalogs*.

To access a golden, blue, or green label, use the *prefix* **f**, **g**, or **h**, respectively. For example, the key **5** preceded by

- **f** will calculate the arithmetic mean values of the data accumulated in the statistic registers via **\bar{x}** ,
- **g** will return the standard deviations for the same data via **s**,
- **h** will open a catalog of supplementary statistic functions via **STAT**.
- The grey letter **R** will become relevant in *alpha mode*.

These prefixes allow for easily accessing a multiple of the 37 primary functions the keyboard can take. You may keep the respective prefix pressed if you want to call several functions in sequence showing the same label color. Any numeric entry will just fill the display and is interpreted when completed, not earlier.

Time for an example. Please take your WP 34S and press the following keys:

- ON** (i.e. the bottom left key) to turn your calculator on. Depending on the last work you did, your WP 34S will show an arbitrary value in display.
- ←** resets this displayed value to zero.
- ALL** (i.e. **h** plus the top left key) returns **ALL__** prompting for your input,
- 0 0** sets your WP 34S to show results with maximum precision. It will display a plain 0 and the radix mark now, nothing more.

Now let us assume you want to fence a little patch of land 40 feet long and 30 feet wide. You set the first corner post (A) already, and also the second (B) in a distance of 30 feet from A. Where do you place the third post (C) to be sure setting up the fence forming a proper rectangle? Simply enter:

- 4 0 ENTER 3 0 →P** The latter (reached via **g** and **→**) will return 50 exactly. So, just take a 90 feet rope, nail its one end on post A and the other one on B, fetch the loose loop and walk 40 feet away. As soon as both parts of the rope are tightly stretched, stop and place post C there. You may set the fourth post the same way.

Further remarks (you will find more about these topics in dedicated sections below):

- The *hotkeys* **A**, **B**, **C**, and **D** immediately call the user programs carrying these labels if defined, else they act as **$\Sigma+$** , **$1/x$** , **y^x** , or **\sqrt{x}** , respectively.
- **→** trailed by **H.MS**, **H.d**, **DEG**, **RAD**, **GRAD**, **2**, **8**, **10**, or **16**, converts x , i.e. the value currently displayed.
- If **.** is used twice in numeric input, the WP 34S enters *fraction mode*.
- **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 of all the commands provided and the necessary explanations.

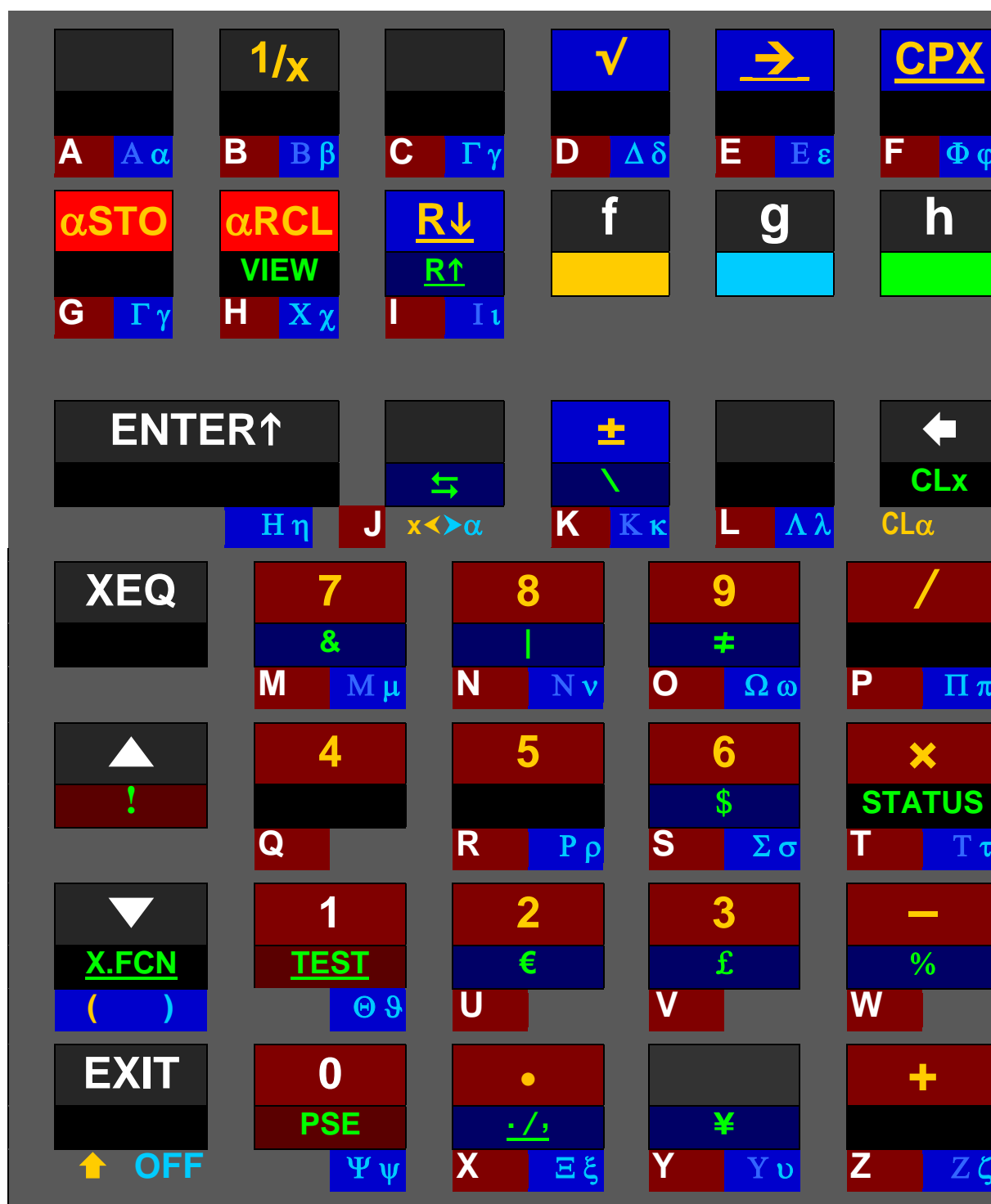
Virtual active keyboard in hexadecimal mode:



Primary functions of the top six keys will be numeric input, so **f** is used for accessing their *default* primary functions. **f** → is exclusively for addressing and temporary display in other bases here (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**:



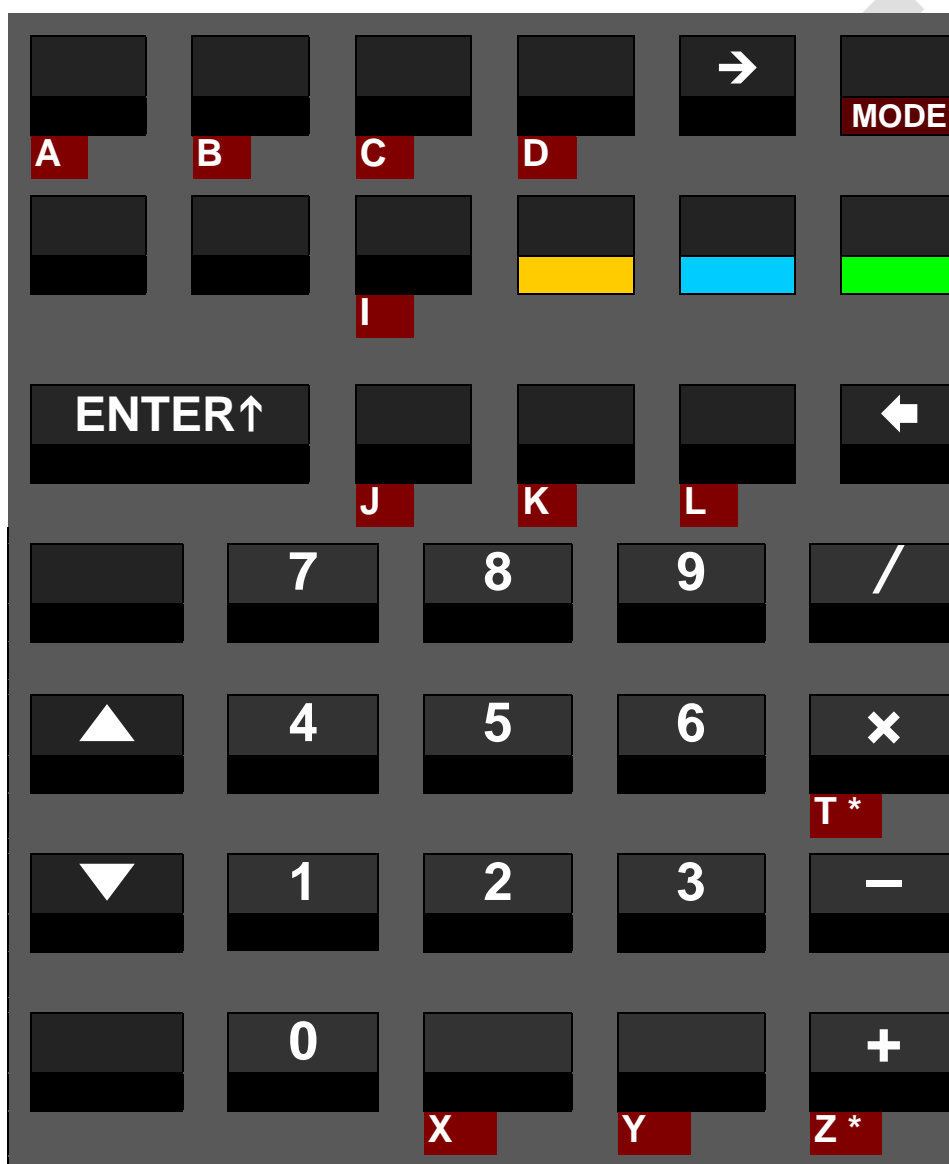
In this mode, the alpha register is displayed in the dot matrix, and the numeric line is accessible by commands only. All labels printed on dark red or blue background in this picture append characters to *alpha* immediately or via alpha catalogs; those on blue deviate from the labels printed on the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

Alpha mode starts with capitals, 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 – grey on the key plate. Prefix  will access the default primary functions

there ². Prefix **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 8, 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 features 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 **X**) by the result **f(x)** stored in **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 your WP 34S, such a two-number real function replaces x by the result **f(x, y)**, eating up the lowest two stack levels but needing only one to put its result in. Thus, level **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 for filling). You may use this top level repetition for some nice tricks.

There are also a few three-number real functions included – e.g. $\text{I}\beta$ and %MRR – replacing x by the result **f(x, y, z)**. Then **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 **CPX** is employed as a prefix for calling complex functions. E.g. **CPX** **f** **COS** calls the complex cosine, and it is displayed and listed as ^CCOS (the elevated C is the signature for complex functions on the WP 34S). All such functions operating on complex numbers do so in Cartesian coordinates exclusively. Each complex number occupies two adjacent registers: the lower one for its real part and the higher one for its imaginary part. Generally, if an arbitrary real function **f** operates on ...

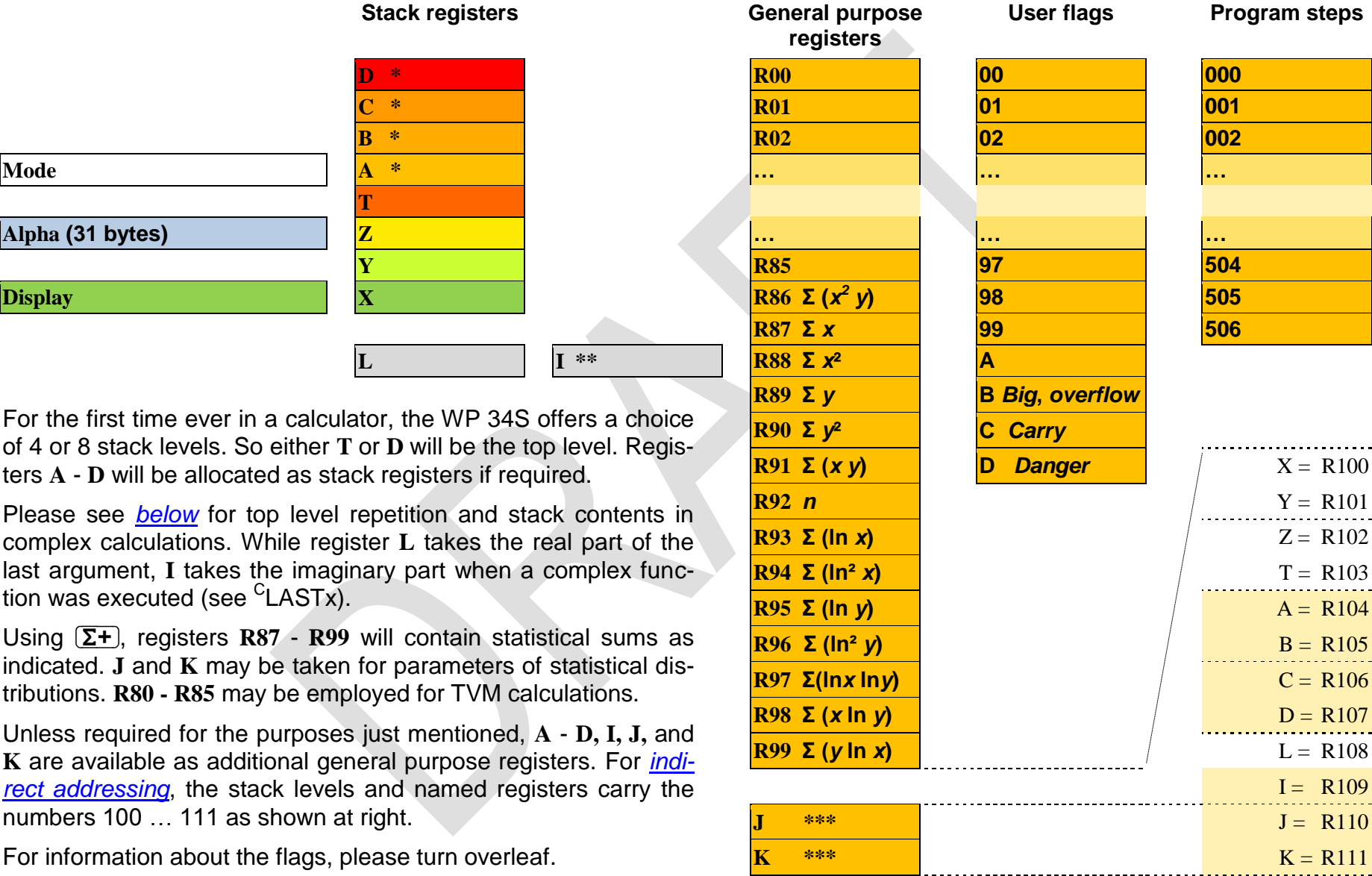
- ... one real number x only, then its complex sibling ^C**f** will operate on the complex number $x_c = x + i y$.
- ... one register, e.g. **R12**, then ^C**f** will operate on **R12** and **R13**.
- ... x and y , then ^C**f** will operate on x , y , z and t .

Where one-number real functions replace x by the result $f(x)$, one-argument complex functions replace x by the real part and y by the imaginary part of the complex result ${}^c f(x_c)$. Higher stack levels remain unchanged. Such functions are ${}^c 1/x$, ${}^c \text{ABS}$, ${}^c \text{ANGLE}$, ${}^c \text{CUBE}$, ${}^c \text{CUBERT}$, ${}^c \text{FIB}$, ${}^c \text{FP}$, ${}^c \text{IP}$, ${}^c \text{RND}$, ${}^c \text{SIGN}$, ${}^c W$, ${}^c W^{-1}$, ${}^c x!$, ${}^c x^2$, ${}^c \sqrt{}$, ${}^c +/ -$, ${}^c \Gamma(x)$, 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 ${}^c f(x_c, y_c)$. 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 [below](#). Such complex functions are ${}^c \text{LOG}_x$, ${}^c y^x$, ${}^c \beta(x, y)$, ${}^c //$, and the basic arithmetic operations in complex domain.

Where complex operations (like ${}^c \text{RCL}$) 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



Flags 00 ... 99 are free to use for whatever purpose you like. Flags A, B, C and D may be used the same way, but the system checks them, too. Flag A lights the big '=' symbol in display. In integer modes, flags B and C will be set by the system in analogy to the overflow and carry bits of the *HP-16C*. Some integer operations (like shift and rotate) also read flag C. Flag D may be set by the user to allow special results (infinities and non-numeric results) without getting an error. The system only reads D. – For [indirect addressing](#), flags A ... D carry the numbers 100 ... 104.

In addition to the RAM provided, the WP 34S allows you accessing **flash memory** for voltage-fail safe storage of user programs and data. Flash memory features four segments (regions, banks) of 1 kB each. Segment 0 is the backup region, holding the image of the entire program memory, registers and calculator state as soon as you completed a SAVE. Segments 1 ... 3 hold programs only. Alphanumeric labels (see below) in flash can be called via XEQ like in RAM. This allows creating program libraries in flash. Use CAT to see the labels defined already.

Flash memory is ideal for backups or other long-living data, but shall not be used for repeated transient storage like in programmed loops (since it will not survive more than approximately 10,000 flashes). 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.

Structuring program memory and jumping around in it is eased by **labels** you may tag to any program steps – as known from previous programmable pocket calculators. The WP 34S features a full set of alphanumeric labels as described [below](#).

When a command like e.g. GTO **xy** is encountered, with **xy** representing one, two or three characters (like A, BC, 12, Tst, Pg3, x1μ, etc.), the WP 34S will search this label **xy** using the following method:

1. If **xy** is purely numeric, it will be searched forward from the current position of the program pointer. When the end of the program space is reached without finding **xy**, the quest will continue at the start of the current segment. No other segments will be searched. This is as known from vintage HP calculators.
2. Else, i.e. if **xy** is an alpha label of up to three characters of arbitrary case, searching will start at program step 000 and cover the entire memory in the order RAM, flash segments 3, 2, 1, 0, and XROM, independent of the position of the program pointer.

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 register 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. Please note only the contents of **X** are displayed in any case. See below for details of the stack mechanics:

| | Level | Assumed stack contents at the beginning: | Stack contents <u>after</u> executing the <u>real</u> stack register operations | | | | | | | ... <u>real</u> functions of | |
|---------------------|-------|--|--|----------|----------|-----------------------|----------|----------|--------------|------------------------------|--------------------------|
| | | | ENTER | FILL | DROP | $x \leftrightarrow y$ | R↓ | R↑ | LASTx | ... one number like x^2 | ... two numbers like $/$ |
| With 4 stack levels | T | <i>t</i> | <i>z</i> | <i>x</i> | <i>t</i> | <i>t</i> | <i>x</i> | <i>z</i> | <i>z</i> | <i>t</i> | <i>t</i> |
| | Z | <i>z</i> | <i>y</i> | <i>x</i> | <i>t</i> | <i>z</i> | <i>t</i> | <i>y</i> | <i>y</i> | <i>z</i> | <i>t</i> |
| | Y | <i>y</i> | <i>x</i> | <i>x</i> | <i>z</i> | <i>x</i> | <i>z</i> | <i>x</i> | <i>x</i> | <i>y</i> | <i>z</i> |
| | X | <i>x</i> | <i>x</i> | <i>x</i> | <i>y</i> | <i>y</i> | <i>y</i> | <i>t</i> | <i>lastx</i> | x^2 | y / x |
| With 8 stack levels | D | <i>d</i> | <i>c</i> | <i>x</i> | <i>d</i> | <i>d</i> | <i>x</i> | <i>c</i> | <i>c</i> | <i>d</i> | <i>d</i> |
| | C | <i>c</i> | <i>b</i> | <i>x</i> | <i>d</i> | <i>c</i> | <i>d</i> | <i>b</i> | <i>b</i> | <i>c</i> | <i>d</i> |
| | B | <i>b</i> | <i>a</i> | <i>x</i> | <i>c</i> | <i>b</i> | <i>c</i> | <i>a</i> | <i>a</i> | <i>b</i> | <i>c</i> |
| | A | <i>a</i> | <i>t</i> | <i>x</i> | <i>b</i> | <i>a</i> | <i>b</i> | <i>t</i> | <i>t</i> | <i>a</i> | <i>b</i> |
| | T | <i>t</i> | <i>z</i> | <i>x</i> | <i>a</i> | <i>t</i> | <i>a</i> | <i>z</i> | <i>z</i> | <i>t</i> | <i>a</i> |
| | Z | <i>z</i> | <i>y</i> | <i>x</i> | <i>t</i> | <i>z</i> | <i>t</i> | <i>y</i> | <i>y</i> | <i>z</i> | <i>t</i> |
| | Y | <i>y</i> | <i>x</i> | <i>x</i> | <i>z</i> | <i>x</i> | <i>z</i> | <i>x</i> | <i>x</i> | <i>y</i> | <i>z</i> |
| | X | <i>x</i> | <i>x</i> | <i>x</i> | <i>y</i> | <i>y</i> | <i>y</i> | <i>d</i> | <i>lastx</i> | x^2 | y / x |

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 begin- ning: | Stack contents <u>after</u> executing the <u>complex</u> stack register operations | | | | | | | ... <u>complex</u> functions of | |
|---------------------------|-------|---|---|-------------------|-------------------|------------------------------------|-----------------|-----------------|--------------------|---|---|
| | | | ^c ENTER | ^c FILL | ^c DROP | ^c x \leftrightarrow y | ^c R↓ | ^c R↑ | ^c LASTx | ... one number like ^c x ² | ... two numbers like ^c / |
| With 4 stack levels | T | $\text{Im}(y_c) = \text{Im}(t_c)$ | $\text{Im}(x_c)$ | | | $\text{Im}(x_c)$ | x_c | | | $y_c = t_c$ | $y_c = t_c$ |
| | Z | $\text{Re}(y_c) = \text{Re}(t_c)$ | $\text{Re}(x_c)$ | | $y_c = t_c$ | $\text{Re}(x_c)$ | x_c | | | $y_c = t_c$ | $y_c = t_c$ |
| | Y | $\text{Im}(x_c)$ | $\text{Im}(x_c)$ | | | $\text{Im}(y_c)$ | y_c | | | $\text{Im}(x_c)^2$ | $\text{Im}(y_c / x_c)$ |
| | X | $\text{Re}(x_c)$ | $\text{Re}(x_c)$ | | y_c | $\text{Re}(y_c)$ | y_c | | | $\text{Re}(x_c)^2$ | $\text{Re}(y_c / x_c)$ |
| With 8 stack levels | D | $\text{Im}(t_c)$ | z_c | x_c | t_c | t_c | x_c | z_c | z_c | t_c | t_c |
| | C | $\text{Re}(t_c)$ | | | | | | | | | |
| | B | $\text{Im}(z_c)$ | y_c | x_c | t_c | z_c | t_c | y_c | y_c | z_c | t_c |
| | A | $\text{Re}(z_c)$ | | | | | | | | | |
| | T | $\text{Im}(y_c)$ | x_c | x_c | z_c | x_c | z_c | x_c | x_c | y_c | z_c |
| | Z | $\text{Re}(y_c)$ | | | | | | | | | |
| | Y | $\text{Im}(x_c)$ | x_c | x_c | y_c | y_c | y_c | t_c | $lastx_c$ | $(x_c)^2$ | y_c / x_c |
| | X | $\text{Re}(x_c)$ | | | | | | | | | |

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 | <div>$x = ?$, $x \neq ?$, $x < ?$, $x \leq ?$, $x \approx ?$, $x \geq ?$, or $x > ?$</div> | | | | <div>RCL, STO, $RCLS$, $STOS$, αRCL, αSTO, $VIEW$, $VW\alpha+$, $x \geq$, DSE, ISG, DSZ, ISZ, FIX, SCI, ENG, $DISP$, $BASE$, $KEY?$, CB and many more bit commands, or CF and the other flag commands etc.</div> | | | |
| | Dot matrix display | <div>$OP _$ (with temporary alpha mode set), e.g. $x > _$</div> | | | | <div>$OP _$ (with temporary alpha mode set), e.g. $RCL _ ^4$</div> | | | |
| 2 | User input | <div>0 or 1</div> | <div>Stack level or named reg. X, Y, ...</div> | <div>$ENTER \uparrow$ ⁵ leaves temp. alpha mode.</div> | <div>\rightarrow opens indirect addressing.</div> | <div>Stack level or named register X, Y, Z, .. , K ⁶</div> | <div>Number of register or flag or bit(s) or decimals ⁷</div> | <div>\rightarrow opens indirect addressing.</div> | |
| | Dot matrix display | <div>$OP n$ e.g. $x \leq 0 ?$</div> | <div>$OP x$ e.g. $x \geq y ?$</div> | <div>$OP r _$</div> | <div>$OP \rightarrow _$</div> | <div>$OP x$ e.g. $SCI Z$</div> | <div>$OP nn$ e.g. $SF 15$</div> | <div>$OP \rightarrow _$</div> | |
| 3 | User input | <div>Compares x with the number 0.</div> | <div>Compares x with the num- ber on stack level Y.</div> | <div>Register no. $00 \dots 99$</div> | <div>Look right for more about indirect ad- dressing.</div> | <div>Sets scientific display with the number of decimals specified in stack level Z.</div> | <div>Stack level etc. X, Y, Z, ... , K</div> | <div>Register number $00 \dots 99$</div> | |
| | Dot matrix display | | | <div>$OP r nn$ e.g. $x \neq r23?$</div> | | | <div>$OP \rightarrow x$ e.g. $VIEW \rightarrow L$</div> | <div>$OP \rightarrow nn$ e.g. $STO \rightarrow 45$</div> | |
| | | | <div>Compares x with the number stored in $R23$.</div> | | | | <div>Shows the content of the register where L is pointing to.</div> | <div>Stores x into the loca- tion where $R45$ is pointing to.</div> | |

⁴ For **RCL** and **STO**, any of **+**, **-**, **x**, **/**, **▲**, or **▼** may precede step 2, except in RCLM and STOM. **VIEW** **ENTER↑** calls αVIEW, And **ENG** **ENTER↑** calls ENGOVR, **SCI** **ENTER↑** calls SCIOVR. See the index of operations.

⁵ You may skip this for register numbers >19.

⁶ Exceptions: RCL T, RCLx 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 Legal register numbers are 00 ... 111 (00 ... 99 may be specified directly). Valid flag numbers are 00 ... 104, with the four top flags directly addressed via **A**, **B**, **C**, and **D**. Legal numbers of decimals are 0 ... 11, accepted 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**. – Please take into account some registers may be allocated to special applications.

COMPARING AND ADDRESSING COMPLEX NUMBERS

| | | | | | | | | | |
|---|--------------------|---|---|--|--|--|---|---|--|
| 1 | User input | <div>CPX</div> <div>x=?</div> or <div>x≠?</div> | | | | <div>CPX</div> <div>RCL</div> , <div>STO</div> , or <div>x></div> | | | |
| | Dot matrix display | OP _ (with temporary alpha mode set) e.g. <div>▯x= ▯</div> | | | | OP _ (with temporary alpha mode set) e.g. <div>▯RCL ▯⁸</div> | | | |
| 2 | User input | <div>0</div> or <div>1</div> | Stack level or named register <div>X</div> , <div>Z</div> , <div>A</div> , <div>C</div> , <div>L</div> , or <div>J</div> | <div>ENTER↑</div> ⁹ leaves temp. alpha mode | <div>→</div> opens indirect addressing. | Stack level or named register <div>Z</div> ¹⁰ , <div>A</div> , <div>C</div> , <div>L</div> , or <div>J</div> | Register number <div>00</div> .. <div>98</div> ¹¹ | <div>→</div> opens indirect addressing. | |
| | Dot matrix display | OP n e.g. <div>▯x= 0 ?</div> | OP x e.g. <div>▯x≠ z ?</div> | OP r_ e.g. <div>▯x≠ r26?</div> | OP →_ e.g. <div>▯x<> →z</div> | OP x e.g. <div>▯RCL L</div> | OP nn e.g. <div>▯STO 18</div> | OP →_ e.g. <div>▯STO →45</div> | |
| 3 | User input | Compares <i>x+iy</i> with the real number 0. | | Compares <i>x+iy</i> with <i>z+it</i> . | Register number <div>00</div> ... <div>98</div> | Look right for more about indi- rect addressing. | | This is ^c LASTx. | |
| | Dot matrix display | | | | OP r nn e.g. <div>▯x≠ r26?</div> | Swaps <i>x</i> with the contents of the register where <i>Z</i> is pointing to, and <i>y</i> with the contents of the next one. | | Stores <i>x+iy</i> into 2 consecutive reg- isters, starting with the one where <i>R45</i> is pointing to. | |







⁸ For $\boxed{\text{RCL}}$ and $\boxed{\text{STO}}$, any of $\boxed{+}$, $\boxed{-}$, $\boxed{\times}$, or $\boxed{/}$ 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 $\boxed{\text{ENTER}\uparrow}$ preceding \boxed{Z} , e.g. $\boxed{\text{CPX}} \boxed{\text{STO}} \boxed{+} \boxed{\text{ENTER}\uparrow} \boxed{Z}$ for the latter.

¹¹ You may key in e.g. $\boxed{8} \boxed{\text{ENTER}\uparrow}$ instead of $\boxed{0} \boxed{8}$. 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. – Please take into account some registers may be allocated to special applications.

ADDRESSING LABELS

| | | | | | | | | |
|---|------------|--|--|--|--|---|--|--|
| 1 | User input | <div><div><div>A</div><div>B</div><div>C</div><div>D</div></div><div>XEQ label e.g. XEQ C</div></div> | | | | | <div><div>XEQ, GTO, LBL, LBL?, SLV, , , ,  or </div><div>OP _ e.g. GTO _</div></div> | |
| 2 | User input | <div><div><div>A</div><div>B</div><div>C</div><div>D</div></div><div>OP label e.g. Σ B</div></div> | | <div><div><div><div>ENTER↑</div><div>sets alpha mode.</div><div>OP ' _</div></div></div><div><div><div><div>¹²</div><div>opens indirect addressing and sets temporary alpha mode.</div><div>OP → _</div></div></div></div></div> | | <div><div>2-digit numeric label <div><div>0</div><div>0</div>...<div>9</div><div>9</div></div></div><div>OP nn e.g. LBL 07</div></div> | | |
| 3 | User input | <div><div>Sums up the function labeled B.</div><div>OP 'label' e.g. SLV'F1μ'</div></div> | | <div><div><div><div>Stack level or named register <div><div>X</div><div>Y</div><div>Z</div>...<div>K</div></div></div><div>OP → x e.g. J →T</div></div></div></div> | | <div><div><div><div>Register number <div><div>0</div><div>0</div>...<div>9</div><div>9</div></div>¹⁴</div><div>OP → nn e.g. XEQ →44</div></div></div></div> | | |
| | | <div><div>Solves the function F1μ (with F1μ keyed in as explained in footer).</div></div> | | <div><div>Integrates the function whose label is on stack level T.</div></div> | | <div><div>Executes the routine whose label is in R44.</div></div> | | |

Additionally, see [above](#) for a description of the way the WP 34S searches labels, and look up GTO in the [index of operations](#) for two special cases applying to this command exclusively.

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 decrement the number of free program steps by 2. – **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, PROBABILITIES 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:

- Discrete statistical distributions are confined to integers. 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:
 - 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.
 - Do you want an upper and 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.
 - 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, for 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 ... in the exponent | b | 3 | 4 | 5 | 6 | 7 | o | 9 | d | -1 | -2 | -3 | -4 | -5 | h | |
| Set by ... | 2 | BASE3, ... , BASE7, 8, BASE9, 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 2 nd □ 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. **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. These are stack depth and contrast set, complete decimal display settings, trig mode, choices for date and time display, the parameters of integer and fraction mode, and curve fitting model selected. STOM stores this information in the register you specify. RCLM recalls the contents of such a register and sets the calculator modes accordingly. Please note the user is responsible for recalling valid mode data – else the WP 34S may be driven into a lockup state! 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;
- SETEUR sets 24h, D.MY, decimal comma, E3ON, and JG1582 (these settings apply also to South America);
- SETIND sets 24h, D.MY, decimal point, E3OFF, and JG1752;
- SETUK sets 12h, D.MY, decimal point, E3ON, and JG1752.
- SETUSA sets 12h, M.DY, 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 unit 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 70 - 99, 80 - D, and 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 , , and are shown until they are resolved. If you pressed any of , , or erroneously, recovery is as easy as follows:

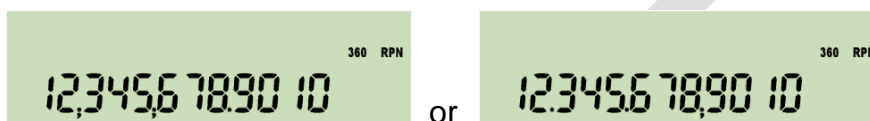
- = NOP = = = =
- = =
 = =
 = =

In addressing, progress is recorded as explained in the [tables above](#) in detail. You may cancel such pending operations by as described [below](#).

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.:



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 marks¹⁶, and additional marks may be chosen to separate thousands. Assume the display set to FIX 4, then 12.345678901 millions may look like:



with thousands separators on, and without them like:



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:



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**.

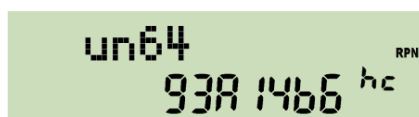
Floating point decimal numbers within $10^{-383} < x < 10^{+385}$ may be entered easily. Using a decimal mantissa, even numbers down to 10^{-394} can be keyed in. The calculator works with numbers down to 10^{-398} correctly. Smaller values are set to zero. For results $x \geq 10^{+385}$, error 4 or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

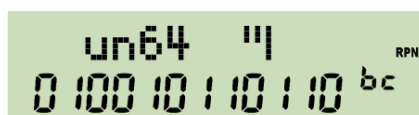
| Base | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|--|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| Sign and 1 st digit of exponent displayed | b | 3 | 4 | 5 | 6 | 7 | o | 9 | d | -1 | -2 | -3 | -4 | -5 | h |

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.

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.

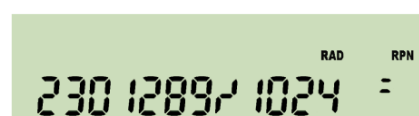
- Fraction mode** works similar to HP-35S. In particular, DENMAX sets the maximum allowable denominator (see the [index of operations](#)). Display will look like in the examples below. If the fraction is exactly equal, slightly less, or greater than the floating point number converted, “=”, “Lt”, or “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 and you will see:



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



Now, enter **a b/c** for converting this result into a proper fraction. You will get



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: |
|---------------------------|---|
| 1 2 . 3 . 4 ENTER↑ | $12 \frac{3}{4}$ |
| 1 . 2 ENTER↑ | $1 \frac{1}{5}$ |
| . 1 . 2 ENTER↑ | $\frac{1}{2}$ |
| . 1 2 ENTER↑ | $\frac{3}{25} \quad (= 0.12)$ |
| 1 . . 2 ENTER↑ | $1 \frac{0}{1} \quad (= 1 \frac{0}{2} !)$ |

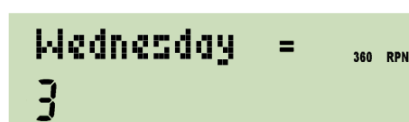
For comparison, please note the HP-32SII reads the last input here as $\frac{1}{2}$ – which is, however, not consistent with its other input interpretations in fraction mode.

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:



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.

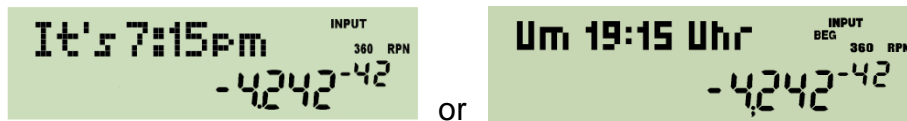
9. Output of the function **DAY** will look as follows for an input of 1.13201 in M.DY mode (equivalent to inputs of 13.01201 in D.MY or 2010.0113 in Y.MD). Expect similar displays after DAYS+.



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.:

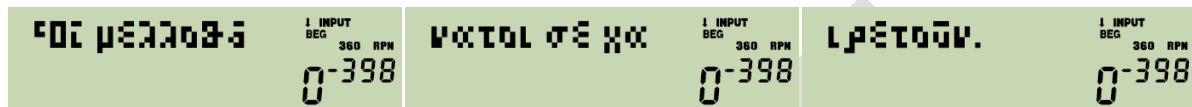




Different information may be appended to *alpha*. See the commands starting with “α” in the index of operations below. E.g. αTIME allows creating texts like

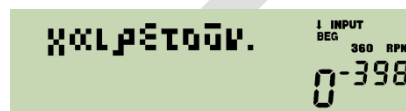


depending on time mode setting (12h / 24h). And αDATE will append – depending on date format setting – either 2011-04-16 or 16.04.2011 or 04/16/2011 to *alpha*.

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



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.



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 |
| ABCDEF GHIJKLM NOPQRSTUVWXYZ |
| ABCDEF GHIJKLM 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 |
| abcdef ghijklm nopqrstuvwxyz |
| abcdef ghijklm nopqrstuvwxyz |

| | | |
|---|-------------------|------------------------|
| Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω | | |
| ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ | | |
| ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ | | |
| α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω | | |
| αβγδεζηθικλμνξοπρστυφχψω | | |
| αβγδεζηθικλμνξοπρστυφχψω | | |
| 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 “true” or “false” 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 <i>names are printed in italics in this list</i> . |
| 10^x | f 10^x | DECM | |
| 12h | h MODE 12h | $\backslash\alpha$ | Sets 12h time display mode meaning 1:23 becomes 1:23 AM and 13:45 becomes 1:45 PM. This makes a difference in α TIME only. |
| 1COMPL | h MODE 1COMPL | $\backslash\alpha$ | Sets 1's complement mode like in <i>HP-16C</i> . |
| $1/x$ | f $1/x$ | DECM | |
| | B | DECM | Shortcut as long as label B is not defined yet. |
| 24h | h MODE 24h | $\backslash\alpha$ | Sets 24h time display mode meaning 1:23 AM becomes 1:23, and 1:45 PM becomes 13:45. |
| 2COMPL | h MODE 2COMPL | $\backslash\alpha$ | Sets 2's complement mode like in <i>HP-16C</i> . |
| 2^x | f 2^x | $\backslash\alpha$ | |
| ABS | f x | $\backslash\alpha$ | 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 | Returns $\arccos(x)$. |
| ACOSH | g \cosh^{-1} COS | DECM | Inverse hyperbolic cosine, known as <i>arcosh</i> . 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 n | $\backslash\alpha$ | Selects the format displaying “all” digits. For $x > 10^{13}$ or $x < 10^{-n}$, display will switch to SCI or ENG with the maximum number of digits displayable (see SCIOVR and ENGOVR). ALL 00 works like ALL in <i>HP-42S</i> . |
| AND | h AND | Integer | Works bitwise as in <i>HP-16C</i> . |
| | | DECM | Works like AND in <i>HP-28S</i> , i.e. x and y are interpreted before executing this operation. 0 is “false”, any other real number is “true”. |

| Name | Keys to press | in modes | Remarks |
|--------|---|---------------------|--|
| 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 two-number function in real domain. |
| | CPX X.FCN ANGLE | DECM | Calculates the angle as above, and returns it in X while clearing Y . This is a one-number function in complex domain. Note there is no need for pressing h here. |
| ASIN | g SIN⁻¹ | DECM | Returns $\arcsin(x)$. |
| ASINH | g HYP⁻¹ SIN | DECM | Inverse hyperbolic sine, known as <i>arsinh</i> . |
| 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 | Returns $\arctan(x)$. |
| ATANH | g HYP⁻¹ TAN | DECM | Inverse hyperbolic tangent, known as <i>artanh</i> . |
| BACK | h P.FCN BACK n | PRG | Jumps n program steps backwards ($1 \leq n \leq 99$). So e.g. BACK 01 goes to the previous step. Reaching step 000 stops program execution. |
| BASE | h MODE BASE n | $\backslash \alpha$ | Sets the base for integer calculations, with $2 \leq n \leq 16$. 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 | f 10 | | |
| BASE16 | g 16 | | |
| BASE2 | f 2 | | |
| BASE8 | g 8 | | |
| BATT | h 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? | h TEST BC? n | Integer | Tests the specified bit in x . |
| BestF | h STAT BestF | DECM | Selects the best curve fit model, maximizing the correlation like BEST does in HP-42S. |

| Name | Keys to press | in modes | Remarks |
|-----------------------------|---|----------|---|
| Binom | h PROB Binom etc. | DECM | Binomial distribution with the number of successes g in X , the probability of a success p₀ in J and the sample size n in K : |
| Binom _P | | | pmf ¹⁷ : $p_B(g; n; p_0) = \binom{n}{g} \cdot p_0^g \cdot (1 - p_0)^{n-g}$. |
| Binom ⁻¹ | | | 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 X . Binom ⁻¹ returns m for given probabilities F_B in X and p in J with sample size n in K . |
| B _n | h X.FCN B _n | DECM | Returns the Bernoulli number for an integer n > 0 given in X : $B_n = (-1)^{n+1} n \cdot \zeta(1-n)$. See below for ζ . |
| B _n [*] | h X.FCN B _n [*] | DECM | Returns the Bernoulli number according to its old definition for integer n > 0 given in X : $B_n^* = \frac{2 \cdot (2n)!}{(2\pi)^{2n}} \cdot \zeta(2n)$. See below for ζ . |
| BS? | h TEST BS? <u>n</u> | Integer | Tests the specified bit in x . |
| Cauch | h PROB Cauch etc. | DECM | Cauchy-Lorentz distribution with the location x₀ specified in J and the shape γ in K , also known as Lorentz or Breit-Wigner distribution: |
| Cauch _P | | | pdf: $f_{Ca}(x) = \frac{1}{\pi\gamma} \cdot \frac{1}{1 + \left(\frac{x - x_0}{\gamma}\right)^2}$ |
| Cauch ⁻¹ | | | cdf: $F_{Ca}(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x - x_0}{\gamma}\right)$. Cauch ⁻¹ returns x for a given probability F_{Ca} in X , with location x₀ in J and shape γ in K . |
| CB | h X.FCN CB <u>n</u> | Integer | Clears the specified bit in x . |
| CEIL | h X.FCN CEIL | DECM | Returns the smallest integer $\geq x$. |
| CF | h P.FCN CF <u>n</u> | \α | Clears the flag specified. |
| CLFLAG | h P.FCN CLFLAG | \α | Clears all user flags. |

¹⁷ The pmf equals BINOMDIST(**g**; **n**; **p₀**; 0) and the cdf BINOMDIST(**m**; **n**; **p₀**; 1) in MS Excel.

| Name | Keys to press | in modes | Remarks |
|--------|---------------------------------|----------|--|
| CLREG | h X.FCN CLREG | All | Clears all general purpose registers. The stack contents are kept. |
| CLSTK | 0 g FILL | \alpha | Clears all stack registers. |
| | h P.FCN CLSTK | | |
| CLx | h CLx | All | Clears the lowest stack register, disabling stack lift as usual. |
| CLα | f CLα | All | Clears the alpha register like CLA in <i>HP-42S</i> . |
| CLΣ | g CLΣ | DECM | Clears all statistical sums in the respective general purpose registers. |
| COMB | f Cy,x | DECM | <p>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.</p> <p>Formula: $C_{y,x} = \binom{y}{x} = \frac{y!}{x!(y-x)!}$</p> |
| CONJ | CPX X.FCN CONJ | DECM | Changes the sign of y . |
| 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 | <p>Returns the population covariance for two data sets. It depends on the fit model selected. For LinF, it calculates</p> $COV_{xy} = \frac{1}{n^2} (n \sum x_i y_i - \sum x_i \sum y_i)$ <p>See s_{xy} for the sample covariance.</p> |
| CUBE | h X.FCN CUBE | \alpha | Returns x^3 . |
| CUBERT | h X.FCN CUBERT | \alpha | Returns $\sqrt[3]{x}$. |
| DATE | h P.FCN DATE | DECM | <p>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.</p> <p>The function DATE of <i>HP-12C</i> corresponds to DAYS+ in WP 34S (see below).</p> |

| Name | Keys to press | in modes | Remarks |
|--------|---------------------------------|--------------------|--|
| DAY | h X.FCN DAY | DECM | Assumes x containing a date in the format selected and extracts the day. |
| DAYS+ | h X.FCN DAYS+ | DECM | Works like DATE in <i>HP-12C</i> , 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 WDAY does. |
| DBLR | h X.FCN DBLR | Integer | Double precision commands for remainder, multiplication and division like in <i>HP-16C</i> . |
| DBL × | h X.FCN DBL× | | |
| DBL / | h X.FCN DBL/ | | |
| DEC | h P.FCN DEC r | $\backslash\alpha$ | Decrements r by one, equivalent to 1 STO- r , but without modifying the stack. |
| DECM | f H.d | $\backslash\alpha$ | Sets default decimal mode for calculations. |
| 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 | $\backslash\alpha$ | Sets default fraction format like in <i>HP-35S</i> , allowing maximum precision. Any denominator up to the value set by DENMAX may appear. |
| DENFAC | h MODE DENFAC | $\backslash\alpha$ | 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 | $\backslash\alpha$ | Sets fixed denominator format, i.e. the denominator equaling DENMAX always. |
| DENMAX | h MODE DENMAX | $\backslash\alpha$ | Works like \wedge/c in <i>HP-35S</i> , 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 n | DECM | Changes the number of decimals while keeping the display format (FIX, SCI, ENG) as is. With ALL set, DISP will change the switchover point (see ALL). |
| DROP | h P.FCN DROP | $\backslash\alpha$ | Drops x . See above for details. |

| Name | Keys to press | in modes | Remarks |
|----------------------|------------------------------------|--------------------|--|
| DSE | f DSE r | PRG & DECM | Given <code>cccccc.ffffii</code> in r , this function decrements r by ii , skipping next program line if then <code>cccccc ≤ fff</code> . |
| 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 | $\backslash\alpha$ | 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. |
| E3OFF | h MODE E3OFF | $\backslash\alpha$ | Toggle the thousands separator (either a point or a comma depending on the radix setting). |
| E3ON | h MODE E3ON | | |
| ENG | h ENG n | $\backslash\alpha$ | Sets engineering display format. |
| ENGOVR | h ENG ENTER↑ | $\backslash\alpha$ | Numbers exceeding the range displayable in ALL or FIX will be shown in engineering format. See SCIOVR. |
| ENTER↑ | ENTER↑ | $\backslash\alpha$ | See above for details. |
| ENTRY? | h TEST ENTRY? | All | Checks the entry flag. It is set if: <ul style="list-style-type: none"> any character is entered in alpha mode, or any command is accepted for entry (be it via ENTER↑, a function key, or R/S with a partial command line). |
| erf | h X.FCN erf | DECM | Returns the error function and its complementary, respectively: $erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\tau^2} d\tau \text{ and } erfc(x) = 1 - erf(x)$ |
| erfc | h X.FCN erfc | DECM | |
| ERR | h P.FCN ERR n | PRG | Raises the error specified. See below for the respective error codes. |
| EVEN? | h TEST EVEN? | $\backslash\alpha$ | Checks if x is integer and even. |
| e^x | f e^x | DECM | |
| ExpF | h STAT ExpF | DECM | Selects the exponential curve fit model. |

| Name | Keys to press | in modes | Remarks |
|---------------------|---|---------------------|--|
| Expon | h PROB Expon etc. | DECM | Exponential distribution with the rate λ specified in J : pdf ¹⁸ : $f_{Ex}(x) = \lambda \cdot e^{-\lambda x}$, cdf: $F_{Ex}(x) = 1 - e^{-\lambda x}$. Expon ⁻¹ returns the survival time t_s for a given probability F_{Ex} in X and rate λ in J . |
| Expon _P | | | |
| Expon ⁻¹ | | | |
| $e^x - 1$ | h X.FCN $e^x - 1$ | DECM | Returns more accurate results for the fractional part of e^x with $x \approx 0$. |
| FAST | h MODE FAST | All | Sets the processor speed to “fast”. This is start-up default and is kept for fresh batteries. |
| FB | h X.FCN FB <u>n</u> | Integer | Inverts (“flips”) the specified bit in x . |
| FC? | h TEST FC? <u>n</u> etc. | $\backslash \alpha$ | Tests if the flag specified is clear. Clears, flips, or sets this flag after testing, if applicable. |
| FC?C | | | |
| FC?F | | | |
| FC?S | | | |
| FF | h P.FCN FF <u>n</u> | $\backslash \alpha$ | Flips the flag specified. |
| FIB | h X.FCN FIB | $\backslash \alpha$ | Returns the Fibonacci number F_x . |
| FILL | g FILL | $\backslash \alpha$ | Copies x to all stack levels. See above for details. |
| FIX | h FIX <u>n</u> | $\backslash \alpha$ | 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? | $\backslash \alpha$ | Tests x for having a nonzero fractional part. |
| FRACT | h MODE FRACT | $\backslash \alpha$ | Sets fraction mode like in HP-35S, but keeps display format as set by PROFRC or IMPFRC. |
| FS? | h TEST FS? <u>n</u> etc. | $\backslash \alpha$ | Tests if the flag specified is set. Clears, flips, or sets this flag after testing, if applicable. |
| FS?C | | | |
| FS?F | | | |
| FS?S | | | |

¹⁸ The pdf corresponds to EXPONDIST(x ; λ ; 0) and the cdf to EXPONDIST(x ; λ ; 1) in MS Excel.

| Name | Keys to press | in modes | Remarks |
|--------------------|---|--------------------------------------|---|
| $F_P(x)$ | h PROB $F(x)$ etc. | DECM | F-distribution. The cdf $F(x)$ equals $1 - Q(F)$ in <i>HP-21S</i> . The degrees of freedom are specified in J and K . |
| $F(x)$ | | | |
| $F^{-1}(p)$ | | | |
| $f'(x)$ | h P.FCN $f'(x)$ <i>label</i> | DECM | Return the first or second derivative of $f(x)$, respectively, with the function $f(x)$ being specified in a routine starting with LBL <i>label</i> . The return stack will have y , z , and t cleared and the position x in L . |
| $f''(x)$ | h P.FCN $f''(x)$ <i>label</i> | | |
| GCD | h X.FCN GCD | $\backslash\alpha$ | Returns the Greatest Common Divisor of x and y . |
| Geom | h PROB Geom etc. | 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 . |
| Geom _P | | | |
| Geom ⁻¹ | | | |
| 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 <i>label</i> | PRG | Inserts an unconditional branch to <i>label</i> . |
| | | \backslash PRG, $\backslash\alpha$ | Positions the program pointer to <i>label</i> . |
| | h GTO \square A , B , C , or D | $\backslash\alpha$ | Positions the program pointer to one of these labels if defined. |
| | h GTO \square <i>nnn</i> | | ... to step <i>nnn</i> . |
| | h GTO \square \square | | ... to step 000 . |
| GTO α | h P.FCN GTO α | $\backslash\alpha$ | Takes the first three characters of <i>alpha</i> (or less if there are less available) as a label and positions the program pointer to it. |

| Name | Keys to press | in modes | Remarks |
|--------------|---|--------------------|---|
| H_n | h X.FCN H_n | DECM | Hermite's polynomials for probability: $H_n(x) = (-1)^n \cdot e^{x^2/2} \cdot \frac{d^n}{dx^n} \left(e^{-x^2/2} \right)$ 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: $H_{np}(x) = (-1)^n \cdot e^{x^2} \cdot \frac{d^n}{dx^n} \left(e^{-x^2} \right)$ with n in Y . |
| H.MS | f H.MS | DECM | Assumes X containing <i>decimal</i> hours or degrees, and displays them converted in the format hhhh°mm'ss.dd" as shown in the paragraph above . Will return to the previous decimal display with the next keystroke thereafter. |
| H.MS+ | h P.FCN H.MS+ | DECM | Assumes X and Y containing times or degrees in the format hhhh.mmssdd, and adds or subtracts them, respectively. |
| H.MS- | h P.FCN H.MS- | | |
| IMPFR | g d/c | $\backslash\alpha$ | Sets fraction mode allowing improper fractions in display (i.e. $\frac{5}{3}$ instead of $1\frac{2}{3}$). 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 X into the equivalent improper fraction, if applicable. |
| INC | h P.FCN INC \underline{r} | $\backslash\alpha$ | Increments r by one, equivalent to 1 STO+ r , but without modifying the stack. |
| INT? | h TEST INT? | $\backslash\alpha$ | 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 \underline{r} | PRG & DECM | Given cccccc.ffffi in r , this function increments r by ii, skipping next program line if then cccccc > fff. |
| ISZ | h P.FCN ISZ \underline{r} | PRG | Increments r by one, skipping next program line if then $ r < 1$. |

| Name | Keys to press | in modes | Remarks |
|--------------|---|--------------------|---|
| I β | h X.FCN I β | DECM | Returns the regularized incomplete beta function $\frac{\beta_x(x, y, z)}{\beta(y, z)} = \frac{1}{\beta(y, z)} \cdot \int_0^x t^{y-1} (1-t)^{z-1} dt$ with β_x being the incomplete beta function and β being Euler's beta (see below). |
| I Γ | h X.FCN I Γ | DECM | Returns the regularized incomplete gamma function $\frac{\gamma(x, y)}{\Gamma(x)}$ with $\gamma(x, y) = \int_0^y t^{x-1} e^{-t} dt$ being the lower incomplete gamma function. For Γ see below. |
| 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 | $\backslash\alpha$ | See above for details. |
| LBL | f LBL <i>label</i> | PRG | Identifies programs and routines for execution and branching. See opportunities for specifying <i>label</i> in the table above . |
| LBL? | h TEST LBL? <i>label</i> | All | Tests for the existence of the label specified, anywhere in program memory. See opportunities for specifying <i>label</i> in the table above . |
| LCM | h X.FCN LCM | $\backslash\alpha$ | 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. |

| Name | Keys to press | in modes | Remarks |
|---------------------|--|----------|--|
| LgNrm | h PROB LgNrm etc. | DECM | <p>Lognormal distribution with $\mu = \ln \bar{x}_g$ specified in J and $\sigma = \ln \varepsilon$ in K. See \bar{x}_g and ε below.</p> <p>pdf: $f_{L_n}(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}$,</p> <p>cdf: $F_{L_n}(x) = \Phi\left(\frac{\ln x - \mu}{\sigma}\right)$ with $\Phi(z)$ denoting the standard normal cdf.</p> <p>LgNrm⁻¹ returns x for a given probability F_{L_n} in X, μ in J, and σ in K.</p> |
| 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 <i>HP-16C</i> . |
| LN | g LN | DECM | Returns the natural logarithm of x , i.e. the logarithm of x for base e . |
| L _n | h X.FCN L _n | DECM | <p>Laguerre's polynomials (compare L_nα below):</p> $L_n(x) = \frac{e^x}{n!} \cdot \frac{d^n}{dx^n} (x^n e^{-x}) = L_n^{(0)}(x) \text{ with } n \text{ in } \mathbf{Y},$ <p>solving the differential equation</p> $x \cdot y'' + (1 - x)y' + ny = 0.$ |
| 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 α | h X.FCN L _n α | DECM | <p>Laguerre's generalized polynomials with n in Y and α in Z:</p> $L_n^{(\alpha)}(x) = \frac{x^{-\alpha} e^x}{n!} \cdot \frac{d^n}{dx^n} (x^{n+\alpha} e^{-x}).$ |
| LNβ | h STAT LNβ | DECM | Returns the natural logarithm of Euler's β function. See there. |
| | h X.FCN LNβ | | |
| LNΓ | h STAT LNΓ | DECM | Returns the natural logarithm of $\Gamma(x)$. See there. |
| | h STAT LNΓ | | |
| 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. |

| Name | Keys to press | in modes | Remarks |
|---------------------|--|--------------------|---|
| 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 pdf: $f_{Lg}(x) = e^{-\frac{x-\mu}{s}} / s \cdot \left(1 + e^{-\frac{x-\mu}{s}}\right)^2$, |
| Logis _P | | | 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 $x + i y$. |
| LZOFF | h MODE LZOFF | $\backslash\alpha$ | Toggles leading zeros like flag 3 does in <i>HP-16C</i> . Relevant in integer modes only. |
| 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 <i>HP-16C</i> , but with the mask length following the command instead of taken from X . |
| MASKR | | | |
| MAX | h X.FCN MAX | $\backslash\alpha$ | Returns the maximum of x and y . |
| MIN | h X.FCN MIN | $\backslash\alpha$ | 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). |
| MONTH | h X.FCN MONTH | DECM | Assumes x containing a date in the format selected and extracts the month. |
| M.DY | h MODE M.DY | $\backslash\alpha$ | Sets the format for date display. |
| NAND | h X.FCN NAND | $\backslash\alpha$ | Works in analogy to AND. |
| NaN? | h TEST NaN? | $\backslash\alpha$ | Tests x for “Not a Number”. |
| nBITS | h X.FCN nBITS | Integer | Counts bits set in x like #B does on <i>HP-16C</i> . |

| Name | Keys to press | in modes | Remarks |
|---------------------|--------------------------------------|--------------------|---|
| NOP | h P.FCN NOP | PRG | “Empty” step FWIW. |
| NOR | h X.FCN NOR | $\backslash\alpha$ | Works in analogy to AND. |
| Norml | h PROB Norml etc. | DECM | Normal distribution with an arbitrary mean μ specified in J and standard deviation σ in K : pdf ¹⁹ : $f_N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, cdf: $F_N(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$ with Φ denoting the standard normal distribution. |
| Norml _p | | | |
| Norml ⁻¹ | h PROB Norml etc. | DECM | Returns x for a given probability F_N in X , mean μ in J , and standard deviation σ in K ²⁰ . |
| NOT | | $\backslash\alpha$ | Works in analogy to AND. |
| nΣ | h STAT nΣ | DECM | Recalls the number of accumulated data points. Necessary for basic statistics. |
| ODD? | h TEST ODD? | $\backslash\alpha$ | Checks if x is integer and odd. |
| OFF | g OFF | PRG | Inserts a step to turn the WP 34S off under program control. |
| OR | h OR | $\backslash\alpha$ | 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: $P_n(x) = \frac{1}{2^n n!} \cdot \frac{d^n}{dx^n} \left[(x^2 - 1)^n \right]$ with n in Y , solving the differential equation $\frac{d}{dx} \left[(1-x^2) \cdot \frac{d}{dx} f(x) \right] + n(n+1)f(x) = 0.$ |

¹⁹ The pdf corresponds to $\text{NORMDIST}(x; \mu; \sigma; 0)$ and the cdf to $\text{NORMDIST}(x; \mu; \sigma; 1)$ in MS Excel.

²⁰ This corresponds to $\text{NORMINV}(F_N; \mu; \sigma)$ in MS Excel.

| Name | Keys to press | in modes | Remarks |
|---------------------|---|---------------------|---|
| Poiss | | DECM | <p>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 $k = 1$:</p> <p>pmf ²¹: $P_p(g; \lambda) = \frac{\lambda^g}{g!} e^{-\lambda}$,</p> <p>cdf: $F_p(m; \lambda) = \sum_{g=0}^m P_p(g; \lambda)$, with the maximum number of successes m in X.</p> <p>Poiss ⁻¹ returns m for given probabilities F_p in X and p in J with sample size n in K.</p> |
| Poiss _{Sp} | h PROB Poiss etc. | | |
| Poiss ⁻¹ | | | |
| PowerF | h STAT PowerF | DECM | Selects the power curve fit model. |
| PRCL | h X.FCN PRCL n | $\backslash \alpha$ | Recall the user program space from flash segment n to RAM where it may be edited (see above). |
| PRIME? | h TEST PRIME? | $\backslash \alpha$ | Checks if the absolute value of the integer part of x is a prime. |
| PROFRC | f a b/c | DECM | Sets fraction mode like in <i>HP-35S</i> , 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 <i>alpha</i> and stops program execution (equaling α VIEW followed by STOP actually). See below for more. |
| PSE | h PSE nn | PRG | Refreshes the display and pauses program execution for nn ticks, with $0 \leq nn \leq 99$. The pause will be terminated early as soon as a key is pressed. |
| PSTO | h X.FCN PSTO | $\backslash \alpha$ | Stores the user program space in flash segment n or exchanges it with the contents of flash segment n , respectively (see above). |
| P \leftrightarrow | h X.FCN P \leftrightarrow | | |
| RAD | g RAD | DECM | Sets angular mode to radians. |

²¹ The pmf corresponds to POISSON($g; \lambda; 0$) and the cdf to POISSON($g; \lambda; 1$) in MS Excel.

| Name | Keys to press | in modes | Remarks |
|-------|--|--------------------|---|
| 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 <i>HP-42S</i> . |
| | | Integer | Returns a random bit pattern for the word size set. |
| RCF | h X.FCN RCF s | $\backslash\alpha$ | Works like RCL but recalls from flash memory. |
| RCFRG | h X.FCN RCFRG | $\backslash\alpha$ | Recover all general purpose registers from the backup region (see SAVE and above). |
| RCFST | h X.FCN RCFST | $\backslash\alpha$ | Recover the system state from the backup region (see SAVE and above). |
| RCF+ | h X.FCN RCF + s etc. | $\backslash\alpha$ | Work like RCL+ etc. but recall from flash memory. |
| RCF− | | | |
| RCF× | | | |
| RCF/ | | | |
| RCF↑ | h X.FCN RCF ▲ s etc. | $\backslash\alpha$ | Work like RCL↑ and RCL↓ but recall from flash memory. |
| RCF↓ | | | |
| RCL | RCL | $\backslash\alpha$ | See the addressing table above for $^{\circ}\text{RCL}$. |
| RCLM | RCL MODE s | $\backslash\alpha$ | Recalls mode settings stored via STOM as described above . |
| | h P.FCN RCLM s | | |
| RCLS | h P.FCN RCLS s | $\backslash\alpha$ | 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. |
| RCL+ | RCL + s | $\backslash\alpha$ | Recalls the content of address s , executes the specified operation on it and pushes the result on the stack. E.g. RCL−12 subtracts $r12$ from x and displays the result (like RCL 12 − , but without losing a stack level). See the addressing table above for $^{\circ}\text{RCL}$. |
| RCL− | RCL − s | | |
| RCL× | RCL × s | | |
| RCL/ | RCL / s | | |
| RCL↑ | RCL ▲ s | $\backslash\alpha$ | RCL↑ (↓) recalls the maximum (minimum) of the values in s and X . |
| RCL↓ | RCL ▼ s | | |

| Name | Keys to press | in modes | Remarks |
|--------|---|----------|--|
| RDX, | h MODE RDX, | \α | Sets the decimal mark to a comma. |
| | h P.FCN RDX, | | |
| RDX. | h ./, | | Toggles the radix mark. |
| | h MODE RDX. | \α | Sets the decimal mark to a point. |
| | h P.FCN RDX. | | |
| RECV | h X.FCN RECV | \α | Prepares your WP 34S for receiving data via serial I/O. See Appendix A for more. |
| RJ | h X.FCN RJ | Integer | Right adjusts, in analogy to LJ on HP-16C. |
| RL | h X.FCN RL <u><i>n</i></u> | Integer | Works like <i>n</i> consecutive RLs / RLCs on HP-16C. For RL, $1 \leq n \leq 63$. For RLC, $1 \leq n \leq 64$. RL 0 and RLC 0 execute as NOP. |
| RLC | h X.FCN RLC <u><i>n</i></u> | | |
| RMDR | h RMDR | \α | MOD of HP-42S equals RMD of HP-16C. |
| ROUND | g RND | DECM | Rounds <i>x</i> using the current display format, like RND in HP-42S. |
| | | FRC | Rounds <i>x</i> using the current denominator, like RND in HP-35S. |
| ROUNDI | h X.FCN ROUNDI | DECM | Rounds <i>x</i> to next integer. ½ rounds to 1. |
| RR | h X.FCN RR <u><i>n</i></u> | Integer | Works like <i>n</i> consecutive RRs / RRCs on HP-16C. See RL / RLC for more. |
| RRC | h X.FCN RRC <u><i>n</i></u> | | |
| 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. |
| 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 none, program execution halts. |

| Name | Keys to press | in modes | Remarks |
|---------------|--------------------|--------------------|--|
| R-CLR | 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 | 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 copied into R45 , R46 , R47 , respectively. For $x < 0$, R-COPY will read nn registers from flash memory, starting with register number $ ss $. |
| R-SORT | 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 | 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. |
| $R\uparrow$ | | $\backslash\alpha$ | Rotates the stack contents one level up or down, respectively. See above for details. |
| $R\downarrow$ | | | |
| s | | DECM | Takes the statistical sums, calculates the sample standard deviations s_y and s_x and pushes them on the stack. |
| SAVE | SAVE | $\backslash\alpha$ | 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 | SB \underline{n} | Integer | Sets the specified bit in x . |
| SCI | \underline{n} | $\backslash\alpha$ | Sets scientific display format. |
| SCIOVR | | $\backslash\alpha$ | Numbers exceeding the range displayable in ALL or FIX will be shown in scientific format (default as in vintage HP calculators). Compare ENGOVR. |
| SEED | SEED | DECM | Stores a seed for random number generation. |

| Name | Keys to press | in modes | Remarks |
|--------|-----------------------------------|--------------------|---|
| SENDA | h X.FCN SENDA | $\backslash\alpha$ | Sends all RAM data via serial I/O. See Appendix A for more. |
| SENDP | h X.FCN SENDP | $\backslash\alpha$ | Sends the user program memory via serial I/O. See Appendix A for more. |
| SENDER | h X.FCN SENDER | $\backslash\alpha$ | Sends the general purpose registers 00 to 99 via serial I/O. See Appendix A for more. |
| 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 \bar{x} and \bar{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 \bar{x}_w). |
| SETCHN | h MODE SETCHN | $\backslash\alpha$ | 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. | $\backslash\alpha$ | 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. | $\backslash\alpha$ | Set some regional preferences (see above). |
| SETUSA | | | |
| SF | h P.FCN SF <u>n</u> | $\backslash\alpha$ | Sets the flag specified. |
| SIGN | h X.FCN SIGN | $\backslash\alpha$ | 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 $x + iy$ in X and Y . |
| SIGNMT | h MODE SIGNMT | $\backslash\alpha$ | 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 $\frac{\sin(x)}{x}$. |
| SINH | f HYP SIN | DECM | Returns the hyperbolic sine of x . |

| Name | Keys to press | in modes | Remarks |
|--------|--|--------------------|--|
| SKIP | h P.FCN SKIP <u><i>n</i></u> | PRG | Skips <i>n</i> program steps forwards ($1 \leq n \leq 99$). 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 <u>occupied</u> program memory, the same will happen as if a RTN had been encountered. |
| SL | h X.FCN SL <u><i>n</i></u> | Integer | Works like <i>n</i> (up to 63) consecutive SLs on HP-16C. SL 0 executes as NOP. |
| SLOW | h MODE SLOW | All | Sets the processor speed to “slow”. This is also entered for low voltage. |
| SLV | f SLV <u><i>label</i></u> | 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 X and Y when calling SLV. For the rest, the user interface is as in HP-15C. Please note SLV acts as a test. |
| SLVQ | h X.FCN SLVQ | DECM | Assumes the stack $[x, y, z, \dots]$ containing the parameters $[c, b, a, \dots]$ of the quadratic equation $ax^2 + bx + c = 0$. SLVQ returns its two roots $-\frac{b}{2a} \pm \sqrt{\left(\frac{b}{2a}\right)^2 - \frac{c}{a}}$ in Y and X . Please note SLVQ works for real numbers only. |
| SPEC? | h TEST SPEC? | $\backslash\alpha$ | True if <i>x</i> is special, i.e. infinity or NaN. |
| SR | h X.FCN SR <u><i>n</i></u> | Integer | Works like <i>n</i> consecutive SRs on HP-16C. SR 0 executes as NOP. |
| SSIZE4 | h MODE SSIZE4 | $\backslash\alpha$ | 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? | $\backslash\alpha$ | Returns the number of stack levels accessible. |
| STO | STO <u><i>d</i></u> | $\backslash\alpha$ | See the addressing table above for ^c STO. |
| STOM | STO MODE <u><i>s</i></u> | $\backslash\alpha$ | Stores mode settings for later use as described above . Take RCLM to recall them. |
| | h P.FCN STOM <u><i>s</i></u> | | |
| STOP | R/S | PRG | Stops program execution. May be used to wait for an input, for example. |

| Name | Keys to press | in modes | Remarks |
|-------|-------------------------------------|---------------------|--|
| STOS | h P.FCN STOS d | $\backslash \alpha$ | Stores all stack levels in a set of 4 or 8 registers, starting at destination d . |
| STO+ | STO + d | $\backslash \alpha$ | <p>Executes the specified operation on the content of address d and stores the result into said address.</p> <p>E.g. STO-12 subtracts x from $r12$ like the sequence RCL 12 x\rceily - STO 12 does, but without touching the stack at all.</p> <p>See the addressing table above for cSTO.</p> |
| STO- | STO - d | | |
| STOx | STO x d | | |
| STO/ | STO / d | | |
| STO↑ | STO ▲ d | $\backslash \alpha$ | STO↑ (↓) takes the maximum (minimum) of the values in d and X and stores it. |
| 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 | <p>Returns the standard deviation for weighted data</p> $s_w = + \sqrt{\frac{\sum y_i \cdot \sum (y_i \cdot x_i^2) - [\sum (y_i \cdot x_i)]^2}{(\sum y_i)^2 - \sum y_i^2}}$ <p>with the weights entered in y via $\Sigma +$.</p> |
| sxy | h STAT sxy | DECM | <p>Returns the sample covariance for two data sets. It depends on the fit model selected. For LinF, it returns</p> $s_{xy} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \cdot (n - 1)}$ <p>See COV for the population covariance.</p> |
| S.L | h P.FCN S.L | DECM | Shifts the decimal radix mark left by x decimals, equivalent to multiplying x by 10^x . |
| S.R | h P.FCN S.R | DECM | Shifts the decimal radix mark right by x decimals, equivalent to dividing x through 10^x . |
| 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 | $\backslash \alpha$ | Returns the number of ticks from the real time clock at execution time. With the quartz built in, 1 tick = 0.1 s. Without, it may be 10% more or less. |

| Name | Keys to press | in modes | Remarks |
|--------------------|-------------------------------------|--------------------|--|
| TIME | h P.FCN 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. Čebyšev, Tschebyschow, Tschebyscheff) polynomials of first kind $T_n(x)$ with n in Y , solving the differential equation $(1-x^2)y''-x \cdot y'+n^2y=0.$ |
| $t_p(x)$ | 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 . |
| $t(x)$ | | | |
| $t^{-1}(p)$ | | | |
| 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 $(1-x^2)y''-3x \cdot y'+n(n+2)y=0.$ |
| UNSIGN | h MODE UNSIGN | $\backslash\alpha$ | Sets unsigned mode for integers. |
| VIEW | h VIEW <u>s</u> | $\backslash\alpha$ | Displays the content of address s until the next key is pressed. See below for more. |
| VW α + | h VIEW <u>s</u> | α | Displays the alpha register in the top line plus the contents of address s in the bottom line until the next key is pressed. See below for more. |
| W | h X.FCN W | DECM | Returns Lambert's W for given $x \geq -1/e$. |
| WDAY | h X.FCN WDAY | 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). |
| W^{-1} | h X.FCN W^{-1} | DECM | Returns x for given W (≥ -1). See W above. |
| Weibl | h PROB Weibl etc. | DECM | Weibull distribution with the shape parameter b in J and the characteristic lifetime T in K : pdf ²² : $f_w(t) = \frac{b}{T} \left(\frac{t}{T}\right)^{b-1} e^{-\left(\frac{t}{T}\right)^b},$ cdf: $F_w(t) = 1 - e^{-\left(\frac{t}{T}\right)^b}.$ Weibl $^{-1}$ returns the survival time t_s for given probability F_w , b in J and T in K . |
| Weibl _p | | | |
| Weibl $^{-1}$ | | | |

²² The pdf equals WEIBULL(x ; **b**; **T**; 0) and the cdf WEIBULL(x ; **b**; **T**; 1) in MS Excel.

| Name | Keys to press | in modes | Remarks |
|--------------|---|--------------------------------------|--|
| WSIZE | h MODE WSIZE n | $\backslash\alpha$ | Works like on <i>HP-16C</i> , 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? | $\backslash\alpha$ | Recalls the word size set. |
| x^2 | g x² | $\backslash\alpha$ | |
| XEQ | XEQ <u>label</u> | PRG | Calls the respective subroutine. |
| | | \backslash PRG, $\backslash\alpha$ | Executes the respective program. |
| | B , C , or D (you may need f for reaching these hotkeys in integer bases >10.) | PRG | Calls the respective subroutine, so e.g. XEQ C will be inserted when C is pressed. |
| | | \backslash PRG, $\backslash\alpha$ | Executes the respective program if defined. |
| XEQ α | h P.FCN XEQ α | $\backslash\alpha$ | Takes the first three characters of <i>alpha</i> (or less if there are less) as a label and calls or executes the respective routine. |
| XNOR | h X.FCN XNOR | $\backslash\alpha$ | Works in analogy to AND. |
| XOR | h XOR | $\backslash\alpha$ | Works in analogy to AND. |
| \bar{x} | 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 . |
| \bar{x}_g | h STAT \bar{x}_g | 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 , ε_m , and ε_p . |
| \bar{x}_w | h STAT \bar{x}_w | DECM | Returns the weighted arithmetic mean $\frac{\sum xy}{\sum y}$. See also sw and SERRw. |
| \hat{x} | h STAT \hat{x} | 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 $\Gamma(x + 1)$. |

| Name | Keys to press | in modes | Remarks |
|------------------------|-------------------------|--|---|
| $x \rightarrow \alpha$ | | All | Interprets x as character code. Appends the respective character to <i>alpha</i> , similar to XTOA in HP-42S. |
| $x \leftrightarrow$ | | $\backslash \alpha$ | Swaps the contents of \mathbf{X} and \mathbf{r} , in analogy to $x \leftrightarrow y$. |
| $x \leftrightarrow y$ | | $\backslash \alpha$ | Swaps x and y , performing $\text{Re} \leftrightarrow \text{Im}$ if a complex operation was executed immediately before. See above for details. |
| $x < \dots ?$ | $x < ?$ | $\backslash \alpha$ | <p>Compare x with \mathbf{a}. E.g. $x < ?$ will compare x with the contents of register \mathbf{K}, and will be listed as $\mathbf{x} < \mathbf{K} ?$ in a program. See the examples given in the addressing table above for more.</p> <p>$x \approx ?$ will be true if the <u>rounded</u> values of x and \mathbf{a} are equal (see ROUND).</p> <p> $x = ?$ and $x \neq ?$ compare the complex number $x + i y$ as explained in the addressing table above.</p> |
| $x \leq \dots ?$ | $x \leq ?$ | | |
| $x = \dots ?$ | $x = ?$ | | |
| $x \approx \dots ?$ | $x \approx ?$ | | |
| $x \neq \dots ?$ | $x \neq ?$ | | |
| $x \geq \dots ?$ | $x \geq ?$ | | |
| $x > \dots ?$ | $x > ?$ | | |
| YEAR | YEAR | DECM | Assumes x containing a date in the format selected and extracts the year. |
| y^x | | $\backslash \alpha$ | In integer modes x must be ≥ 0 . |
| | | $\backslash \alpha$ & $\backslash (13, 14, 15, h)$ | Shortcut working as long as label C is not defined yet. |
| \hat{y} | | DECM | Returns a forecast \mathbf{y} (in \mathbf{X}) for a given x following the fit model chosen. See L.R. for more. |
| Y.MD | Y.MD | $\backslash \alpha$ | Sets the format for date display. |
| αDATE | αDATE | $\backslash \text{integer}$ | Takes x as a date and appends it to <i>alpha</i> in the format set. See DATE. – To append a date stamp to <i>alpha</i> , call DATE αDATE . |
| αDAY | αDAY | $\backslash \text{integer}$ | Takes x as a date, recalls the name of the respective day and appends its first 3 letters to <i>alpha</i> . |
| αGTO | $\alpha \text{GTO } nn$ | $\backslash \alpha$ | Takes the contents of \mathbf{Rnn} 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. |

| Name | Keys to press | in modes | Remarks |
|----------------|---|----------------|---|
| α IP | h X.FCN α IP | All | Appends the integer part of x to <i>alpha</i> , similar to AIP in <i>HP-42S</i> . |
| α LENG | h X.FCN α LENG | All | Returns the number of characters found in <i>alpha</i> , like ALENG in <i>HP-42S</i> . |
| α MONTH | h X.FCN α MONTH | \integer | Takes x as a date, recalls the name of the respective month and appends its first 3 letters to <i>alpha</i> . |
| α OFF | h P.FCN α OFF | PRG & α | Work like AOFF and AON in <i>HP-42S</i> , turning alpha mode off and on. |
| α ON | h P.FCN α ON | PRG & \alpha | |
| α RCL | f RCL <u>s</u> | α | Interprets the content of the source s as characters and appends them to <i>alpha</i> . |
| | h X.FCN α RCL <u>s</u> | \alpha | |
| α RC# | h X.FCN α RC# <u>s</u> | All | Takes the content of s as a number, converts it to a string in the format set, and appends this to <i>alpha</i> . If e.g. $s = 1234$ and ENG 2 and RDX. are set, then <u>1.23E3</u> will be appended. |
| α RL | h X.FCN α RL <u>n</u> | All | Rotates <i>alpha</i> by n characters like AROT in <i>HP-42S</i> , but with $n \geq 0$ and the parameter trailing the command instead of taken from X . α RL 0 executes as NOP. |
| α RR | h X.FCN α RR <u>n</u> | All | Works like α RL but rotates to the right. |
| α SL | h X.FCN α SL <u>n</u> | All | Shifts the n leftmost characters out of <i>alpha</i> , like ASHF in <i>HP-42S</i> . α SL 0 equals NOP. |
| α SR | h X.FCN α SR <u>n</u> | All | Works like α SL but takes the n rightmost characters instead. |
| α STO | f STO <u>d</u> | α | Stores the first (i.e. leftmost) 6 characters in the alpha register into destination d . |
| | h X.FCN α STO <u>d</u> | \alpha | |
| α TIME | h X.FCN α TIME | \integer | Takes x as a decimal time and appends it to <i>alpha</i> in the format hh:mm:ss according to the time mode selected. See TIME. – To append a time stamp to <i>alpha</i> , call TIME α TIME. |

| Name | Keys to press | in modes | Remarks |
|------------------------|---|--------------------|--|
| α VIEW | h VIEW α | All | Displays alpha in the top line and - - - in the bottom line until the next key is pressed. See below for more. |
| | h P.FCN α VIEW | | |
| | h X.FCN α VIEW | | |
| α XEQ | h P.FCN α XEQ <i>nn</i> | $\backslash\alpha$ | Takes the contents of Rnn 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. |
| $\alpha \rightarrow x$ | f x ◀▶ α | 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 $B(x, y) = \frac{\Gamma(x) \cdot \Gamma(y)}{\Gamma(x+y)}$ with $\text{Re}(x) > 0$, $\text{Re}(y) > 0$. Called β here for avoiding ambiguities. |
| | h X.FCN β | | |
| Γ | h STAT Γ | DECM | Returns $\Gamma(x)$. Additionally, h ! calls $\Gamma(x+1)$. |
| | h X.FCN Γ | | |
| Δ DAYS | h X.FCN Δ DAYS | DECM | Assumes X and Y containing dates in the format chosen and calculates the number of days between them. Works like in HP-12C. |
| $\Delta\%$ | g Δ% | DECM | Returns $100 \cdot \frac{x-y}{y}$ like %CH in HP-42S. |
| ε | h STAT ε | DECM | Calculates the scattering factors (or geometric standard deviations) for lognormally distributed data $\ln(\varepsilon_y) = \sqrt{\frac{\sum \ln^2(y) - 2n \cdot \ln(\bar{y}_G)}{n-1}}$ and $\ln(\varepsilon_x)$ and pushes them on the stack. This ε works for the geometric mean \bar{x}_g in analogy to s for the arithmetic mean \bar{x} but <u>multiplicative</u> . |
| ε_m | h STAT ε_m | DECM | Works like ε but pushes the scattering factors of the geometric means $\varepsilon_m = \varepsilon^{\frac{1}{\sqrt{n}}}$ on the stack. |
| ε_p | h STAT ε_p | DECM | Works like ε but with a denominator n instead of n-1 , returning the scattering factors of the populations. – Streichkandidaten. Zusatzabschnitt über lognv Daten vorne einfügen. |

| Name | Keys to press | in modes | Remarks |
|------------------|--|----------|--|
| 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 <u>label</u> | DECM | Computes a product with the routine specified by label . Initially, X contains the loop control number in the format <code>cccccc.ffffii</code> 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 <code>cccccc</code> by <code>ii</code> and runs said routine again if then <code>cccccc > fff</code> , else returns the resulting product in X . |
| Σ | g Σ <u>label</u> | DECM | Computes a sum with the routine specified by label . Initially, X contains the loop control number in the format <code>cccccc.ffffii</code> and the sum is set to 0. Each run through the routine specified computes a summand. At its end, this summand is added to said sum; the operation then decrements <code>cccccc</code> by <code>ii</code> and runs said routine again if then <code>cccccc > fff</code> , else returns the resulting sum in X . |
| σ | h STAT σ | DECM | Works like s but returns the standard deviations of the populations instead. |
| $\Sigma \ln^2 x$ | 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. |
| $\Sigma \ln^2 y$ | | | |
| $\Sigma \ln x$ | | | |
| $\Sigma \ln xy$ | | | |
| $\Sigma \ln y$ | | | |
| $\Sigma x \ln y$ | | | |
| $\Sigma y \ln x$ | | | |
| σ_w | h STAT σ_w | DECM | Works like sw but returns the standard deviation of the population instead. $\sigma_w = + \sqrt{\frac{\sum y_i (x_i - \bar{x}_w)^2}{\sum y_i}}$ |

| Name | Keys to press | in modes | Remarks |
|---------------------------|--|---------------------|--|
| Σx | 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. |
| Σx^2 | | | |
| $\Sigma x^2 y$ | | | |
| Σxy | | | |
| Σy | | | |
| Σy^2 | | | |
| $\Sigma +$ | h $\Sigma +$ | DECM | Adds a data point to the statistical sums. |
| | A | DECM | Shortcut as long as label A is not defined yet. |
| $\Sigma -$ | h $\Sigma -$ | 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 <i>HP-32E</i> and $1 - Q(z)$ in <i>HP-21S</i> with $z = x$. |
| $\Phi^{-1}(p)$ | g Φ^{-1} | | |
| χ^2 | 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 <i>HP-21S</i> . |
| $\chi^2 \text{INV}$ | | | |
| χ^2_P | | | |
| $(-1)^x$ | h X.FCN $(-1)^x$ | $\backslash \alpha$ | For x not being a natural number, this function will return $\cos(\pi \cdot x)$. |
| $+$ | + | $\backslash \alpha$ | Returns $y + x$. |
| $-$ | - | $\backslash \alpha$ | Returns $y - x$. |
| \times | x | $\backslash \alpha$ | Returns $y \cdot x$. |
| $/$ | / | $\backslash \alpha$ | Returns y / x . |
| $+/-$ | +/- | $\backslash \alpha$ | Unary minus like CHS in <i>HP-35</i> . |
| $\rightarrow \text{DEG}$ | \rightarrow (g) DEG | DECM | Takes x as an angle in the angular mode currently set and converts it to degrees. Prefix g may be omitted. |
| $\rightarrow \text{GRAD}$ | \rightarrow (g) GRAD | DECM | Like $\rightarrow \text{DEG}$, but converts to gon or grads. |

| Name | Keys to press | in modes | Remarks |
|-------|---------------|----------|---|
| →H | | DECM | Takes x as hours or degrees in the format $hhhh.mmssdd$ and converts them into a decimal time or angle. |
| →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 | | 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 | () | DECM | Works like →DEG, but converts to radians. |
| →REC | | DECM | Assumes X and Y containing 2D polar coordinates (r, θ) and converts them to the respective Cartesian coordinates (x, y) . |
| % | | DECM | Returns $\frac{x \cdot y}{100}$, leaving Y unchanged. |
| %MG | MG | DECM | Returns the margin ²³ $100 \cdot \frac{x-y}{x}$ in % for a price x and cost y , like %MU-Price in HP-17B. |
| %MRR | MRR | DECM | Returns the mean rate of return in percent per period, i.e. $100 \cdot \left[\left(\frac{x}{y} \right)^{\frac{1}{z}} - 1 \right]$ with $x = FV =$ future value after z periods, $y = PV =$ present value. For $z = 1$, $\Delta\%$ returns the same result easier. |
| %T | T | DECM | Returns $100 \cdot \frac{x}{y}$, interpreted as % of total. |
| %Σ | Σ | DECM | Returns $100 \cdot \frac{x}{\sum x}$. |
| | Σ | | |
| %+MG | +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. |

²³ Margin corresponds to „Handelsspanne“ in German.














| Name | Keys to press | in modes | Remarks |
|----------------------|---------------|---|---|
| $\sqrt{}$ | | $\backslash\alpha$ | |
| | | $\backslash\alpha$, $\backslash14$, $\backslash15$, $\backslash h$ | Shortcut working as long as label D is not defined yet. |
| \int | <u>label</u> | 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 <i>HP-15C</i> . |
| $\infty?$ | $\infty?$ | $\backslash\alpha$ | Tests x for infinity. |
| // | | DECM | Returns $\left(\frac{1}{x} + \frac{1}{y}\right)^{-1}$. |

Alphanumeric input:

| Character | Keys to press | in modes | Remarks |
|-----------|-------------------|---------------------------|---|
| _ | | α | Appends a blank space to <i>alpha</i> . |
| ° | | DECM | Separates degrees or hours from minutes and seconds, so input format is <code>hhhh.mmssdd</code> . The user has to take care where an arbitrary real number represents such an angle or time. |
| 0 ... 9 | ... | $\backslash\alpha$ | Standard numeric input. For integer bases <10, input of illegal digits throws an error message . Please note you cannot enter more than 12 digits in the mantissa. |
| | | in addressing | Register input. See the tables above for more. |
| | , , , ..., | α | Appends the respective digit to <i>alpha</i> . |
| A ... F | ... (grey print) | 11, 12, 13, 14, 15, h | Numeric input for digits >10. See page 6 for more information. |
| A ... Z | ... (grey print) | in addressing | Register input. See the addressing tables above for the letters applicable. |
| | | α | Appends the respective Latin letter to <i>alpha</i> . Use to toggle cases. |
| EEX | | DECM & \backslash FRACT | Works like in the Pioneers. |

| Character | Keys to press | in modes | Remarks |
|------------|--------------------|----------|---|
| A ... Ω | ... (grey print) | α | Appends the respective Greek letter to <i>alpha</i> . Use to toggle cases. See p. 7 for more. |
| (| | α | Appends the respective symbol to <i>alpha</i> . |
|) | | | |
| +, −, × | ... | | |
| / | Second | DECM | A persistent 2 nd in input switches to fraction mode. It will be interpreted as explained below. Please note you cannot enter after you entered twice – but you may delete the 2 nd dot while editing the input line. |
| | | FRC | First is interpreted as a space, 2 nd as a fraction mark. E.g. input of results in 2 ¾ in the display. Improper fractions may be entered starting with a , e.g. . |
| | | α | |
| ± | | α | Appends the respective symbol to <i>alpha</i> . |
| , | | | |
| . | | | |
| ‘.’ or ‘,’ | | DECM | Inserts a radix mark as selected. |
| ! | | α | Appends the respective symbol to <i>alpha</i> . |
| ↔ | | | |
| ≠ | | | |
| % | | | |
| \$ | (grey print) | | |
| € | (grey print) | | |
| £ | | | |
| ¥ | (grey print) | | |
| √ | | | |
| & | | | |
| \ | | | |
| | | | |

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 <i>alpha</i> 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 <i>HP-42S</i> . |
| | Input pending | Deletes the last digit or character put in. |
| | α | Deletes the rightmost character in <i>alpha</i> . |
| | PRG | Deletes current step. |
|  /  | Else | Acts like CLx. |
| | Integer | Shifts the display window to the left / right like in <i>HP-16C</i> . Helpful while working with small bases. |
| | α | Toggles upper and lower case. |
| | $\backslash\alpha$ | Enters a memory browser. |
|  | $\backslash\alpha$ | Enters a memory browser. |
| | $\backslash\text{PRG}$ | Clears all registers and programs if confirmed. |
| | $\backslash\text{PRG}$ | Positions the program pointer to step 000 and clears the subroutine return stack. |
| | PRG | Clears the program memory if confirmed. |
|  CLALL | PRG | Clears the program memory if confirmed. |
| | $\backslash\text{PRG}$ | Positions the program pointer to step 000 and clears the subroutine return stack. |
| | PRG | Clears the program memory if confirmed. |
| | PRG | Clears the program memory if confirmed. |
|  | Catalog open | Selects the current item like XEQ below. |
| | α | Turns alpha mode off. |
| | 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. |
|  | $\backslash\text{PRG}$ & program running | Stops the running program like R/S . See below. |
| | PRG | Leaves programming mode like  . See below. |
| | α | Turns alpha mode off like ENTER . See above. |
| | Else | Does nothing. |
|  | 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. |
| | $\backslash\text{PRG}$ & program running | Stops the running program like R/S . See below. |
| | PRG | Leaves programming mode like  . See below. |
|  | α | Turns alpha mode off like ENTER . See above. |
| | Else | Does nothing. |
| | Else | Does nothing. |
| | Else | Does nothing. |

| Keys to press | in modes | Remarks |
|---------------------------------|--------------------|---|
| g OFF | \PRG | Turns calculator off. |
| ON | Calculator off | Turns calculator on. |
| ON | All | There are several ON -key combinations available. See below for more. |
| 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 00, DEG, DENANY, DENMAX 9999, DECM, LinF, PROFRC, RDX., SCIOVR, SSIZE4, WSIZE 64, Y.MD. |
| R/S | \PRG, \α | If a program is running: Stops it immediately. “Stopped” will be shown in the upper line until the next keystroke. Else: 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. |
| h SHOW | 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 below . |
| 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 <i>alpha</i> always. For starting a new string, use CLα first. Alpha constants will be listed like e.g. ‘Test 1’. |
| → f 2 | \α | These commands show <i>x</i> 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 → |
| → f 10 | | |
| → (g) 16 | | |
| → (g) 8 | | |

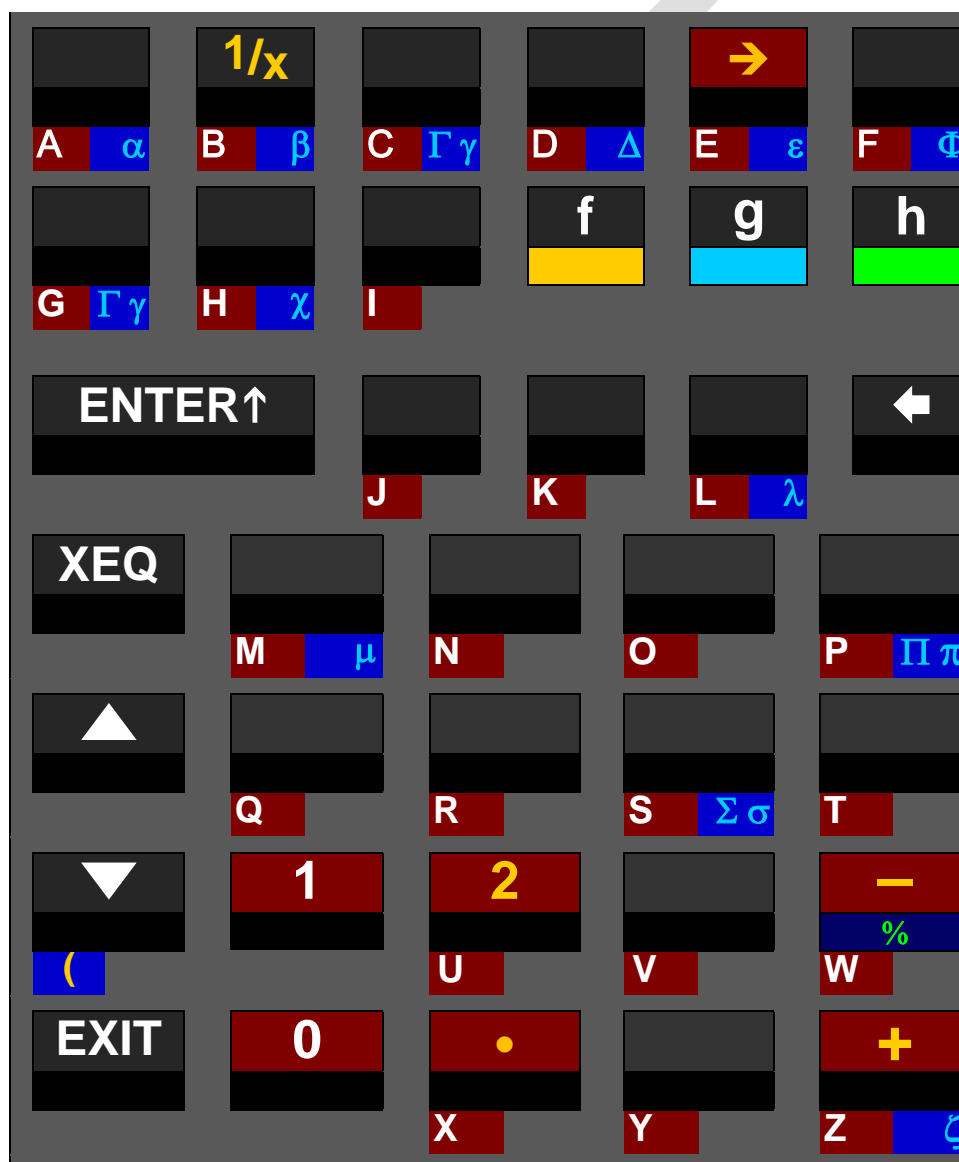
CATALOGS

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 | $\backslash\alpha$ | <p>Predefined alpha labels. Some special rules apply here:</p> <p>▲ 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 <i>n</i> for flash memory segment <i>n</i>).</p> <p>0 – 3 go to the first alpha label in the flash segment specified.</p> <p>ENTER↑ goes to the alpha label as displayed, while XEQ or R/S execute it. These keystrokes will perform a label search as described above. Labels in XROM cannot be accessed by ENTER↑.</p> <p>. goes to the first alpha label in XROM.</p> <p>← or EXIT leave CAT returning to the state as it was before.</p> |
| h CONST | DECM | Constants like in HP35s. Picking a constant will recall it. See the constants listed in a table below . |
| CPX CONST | DECM | This catalog contains the same constants as in real domain. Picking one, however, does a complex recall here. So, if the stack did look like $[x, y, \dots]$ before calling CONST, it will contain [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 | $\backslash\alpha$ | Mode setting functions. |
| h PROB | DECM | Extra probability distributions. |
| h P.FCN | $\backslash\alpha$ | Extra programming functions. |
| f R↓ | α | Subscripts. |
| h R↑ | α | Superscripts. |
| h STAT | DECM | Extra statistical functions. |
| h TEST | $\backslash\alpha$ | 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. **f** **B** (= **1/x**) eases operations in [CONV](#).

▲ 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. This position is lost when the WP 34S is turned off.

See the [*table below about addressing cataloged items*](#), and the next pages for detailed item lists of the various catalogs.

Within each catalog, items are sorted alphabetically (see [above](#) for the sorting order). You may access particular items fast and easily by typing the first characters of their names. See [below](#) for some examples and constraints.

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

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

DRAFT

Catalog contents in detail:

[illegible]

| X.FCN varies with the mode set, except in PRG. It contains in ... | | | | | |
|---|--------------------|---------------------|----------------|--------------------|---------------|
| ... alpha mode: | ... decimal mode: | | | ... integer modes: | |
| | AGM | MIN | XNOR | ASR | RECV |
| CLALL | ANGLE | MONTH | YEAR | BATT | RESET |
| CLREG | BATT | NAND | α DATE | CB | RJ |
| RESET | B_n | NOR | α DAY | CLALL | RL |
| VERS | B_n^* | P_n | α IP | CLFLAG | RLC |
| α DATE | CEIL | PRCL | α LENG | CLREG | RR |
| α DAY | CLALL | PSTO | α MONTH | CUBE | RRC |
| α IP | CLREG | $P \leftrightarrow$ | α RCL | CUBERT | SAVE |
| α LENG | CUBE | RAD \rightarrow | α RC# | DBLR | SB |
| α MONTH | CUBERT | RCF | α RL | DBL* | SEED |
| α RC# | DAY | RCFRG | α RR | DBL/ | SENDA |
| α RL | DAYS+ | RCFST | α SL | FB | SENDP |
| α RR | DECOMP | RCF+ | α SR | FIB | SENDR |
| α SL | DEG \rightarrow | RCF- | α STO | GCD | SIGN |
| α SR | D \rightarrow J | RCF \times | α TIME | LCM | SL |
| α TIME | erf | RCF/ | α VIEW | LJ | SR |
| | erfc | RCF \uparrow | β | LOAD | VERS |
| | $e^x - 1$ | RCF \downarrow | Γ | MASKL | VW α + |
| | FIB | RECV | Δ DAYS | MASKR | XNOR |
| | FLOOR | RESET | ζ | MAX | α IP |
| | GCD | ROUNDI | $(-1)^x$ | MIN | α LENG |
| | GRAD \rightarrow | SAVE | %MG | MIRROR | α RCL |
| | H_n | SENDA | %MRR | NAND | α RC# |
| | H_{np} | SENDP | %T | nBITS | α RL |
| | $I\beta$ | SENDR | % Σ | NOR | α RR |
| | $I\Gamma$ | SETDAT | %+MG | PRCL | α SL |
| | JG1582 | SETTIM | | PSTO | α SR |
| | JG1752 | SIGN | | $P \Leftarrow$ | α STO |
| | J \rightarrow D | SINC | | RCF | α VIEW |
| | LCM | SLVQ | | RCFRG | $(-1)^x$ |
| | L_n | T_n | | RCFST | |
| | LN1+x | U_n | | RCF+ | |
| | $L_n\alpha$ | VERS | | RCF- | |
| | LN β | VW α + | | RCF \times | |
| | LN Γ | W | | RCF/ | |
| | LOAD | WDAY | | RCF \uparrow | |
| | MAX | W^{-1} | | RCF \downarrow | |

| CPX |
|------------------|
| X.FCN |
| c AGM |
| c CONJ |
| c CUBE |
| c CUBERT |
| c DROP |
| c $e^x - 1$ |
| c FIB |
| c LN1+x |
| c LN β |
| c LN Γ |
| c RCF |
| c SIGN |
| c SINC |
| c W |
| c W^{-1} |
| c β |
| c Γ |
| $^c(-1)^x$ |

| CPX | | | | |
|---------|---|---|---------|---------|
| À | À | À | à | à |
| Á | Á | Á | á | á |
| Â Ã Ä Å | Â | Â | â ã ä å | â ã ä å |
| Ä | Ä | Ä | ä (ä) | ä |
| Å | Å | Å | å | å |
| Ć | Ć | Ć | ć | ć |
| Č | Č | Č | č | č |
| Ç | Ç | Ç | ç | ç |
| È | È | È | è | è |
| É | É | É | é | é |
| Ê Ë Æ | Ê | Ê | ê ë æ | ê ë æ |
| Ë | Ë | Ë | ë (ë) | ë |
| | | | ħ | ħ |
| Ì | Ì | Ì | ì | ì |
| Í | Í | Í | í | í |
| Î Ï Î Ï | Î | Î | î ï î ï | î ï î ï |
| Ĭ | Ĭ | Ĭ | ĭ (ĭ) | ĭ |
| Ñ Ñ | Ñ | Ñ | ñ ñ | ñ ñ |
| Ò | Ò | Ò | ò | ò |
| Ó | Ó | Ó | ó | ó |
| Ô Õ Ö | Ô | Ô | ô õ ö | ô õ ö |
| Ö | Ö | Ö | ö (ö) | ö |
| Ø | Ø | Ø | ø | ø |
| Ř | Ř | Ř | ř | ř |
| Š | Š | Š | š | š |
| | | | ß | ß |
| Ù | Ù | Ù | ù | ù |
| Ú | Ú | Ú | ú | ú |
| Û Ü | Û | Û | û ü | û ü |
| Ü | Ü | Ü | ü (ü) | ü |
| Ů | Ů | Ů | ů | ů |
| Ý | Ý | Ý | ý | ý |
| ÿ | ÿ | ÿ | ÿ | ÿ |
| Ž | Ž | Ž | 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 ∞ |
| 0 1 2 A B c e k m n p u μ ∞ |
| 0 1 2 A B c e k m n p u μ ∞ |

| |
|--|
| R↑ (superscripts) |
| c ° 2 x x̄ x̂ x̃ x̄ x̅ x̆ ẋ ẍ x̉ x̊ x̋ x̌ x̍ x̎ x̏ x̐ x̑ x̒ x̓ x̔ x̕ x̖ x̗ x̘ x̙ x̚ x̛ x̜ x̝ x̞ x̟ x̠ x̡ x̢ x̣ x̤ x̥ x̦ x̧ x̨ x̩ x̪ x̫ x̬ x̭ x̮ x̯ x̰ x̱ x̲ x̳ x̴ x̵ x̶ x̷ x̸ x̹ x̺ x̻ x̼ x̽ x̾ x̿ x̺ x̻ x̼ x̽ x̾ x̿ |
| 0 1 2 A B c e k m n p u μ ∞ |
| 0 1 2 A B c e k m n p u μ ∞ |

The letters provided in the WP 34S allow for correct writing the languages of more than 3·10⁹ people (still only half of mankind yet), i.e.:

Afrikaans, Català, Cebuano, Český, 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 ǎ here, and for e, i, o, and u as well. With six 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. | | | | | |
| 2 | User input | XEQ , ▼ , ▲ , EXIT , or 1 st 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 2 nd 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. Ö) (e.g. ?) | | | | | |
| ... | | | | | | | |
| Continue browsing this way until reaching the item desired | | | | | | | |
| | | (e.g. FS?F). | (e.g. Ü). | (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 2010, copied in July 2011, unless stated otherwise explicitly. Green background denotes exact or almost exact values. The more the color turns to red, the less precise the respective constant is known ²⁴.

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

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

| | Numeric value | Unit | Remarks |
|-----------------------|--|-----------------|---|
| a | 365.2425 (<i>per definition</i>) | <i>d</i> | Gregorian year |
| a₀ | 5.2917721092E-11 (<i>3.2E-10</i>) | <i>m</i> | Bohr radius $= \frac{\alpha}{4\pi \cdot R_{\infty}}$ |
| a_m⊕ | 384.4E6 (?) | <i>m</i> | Semi-major axis of the Moon's orbit around the Earth |
| a⊕⊙ | 1.495979E11 (?) | <i>m</i> | Semi-major axis of the Earth's orbit around the sun |
| c | 2.99792458E8 (<i>per definition</i>) | $\frac{m}{s}$ | Vacuum speed of light |
| c₁ | 3.74177153E-16 (<i>4.4E-8</i>) | $m^2 \cdot W$ | First radiation constant $= 2\pi \cdot h \cdot c^2$ |
| c₂ | 0.014387770 (<i>9.1E-7</i>) | $m \cdot K$ | Second radiation constant $= \frac{hc}{k}$ |
| e | 1.602176565E-19 (<i>2.2E-8</i>) | <i>C</i> | Electron charge $= \frac{2}{K_J R_K} = \Phi_0 G_0$ |
| eE | 2.718281828459045... | 1 | Euler's e. Please note the letter e represents the electron charge elsewhere in this table. |
| F | 96485.3365 (<i>2.2E-8</i>) | $\frac{C}{mol}$ | Faraday's constant $= e N_A$ |
| Fα | 2.50290787509589282228... | 1 | Feigenbaum's α |
| Fδ | 4.66920160910299067185... | 1 | Feigenbaum's δ |
| g | 9.80665 (<i>per definition</i>) | $\frac{m}{s^2}$ | Standard earth acceleration |

²⁴ 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 |
|-----------------------------------|----------------------------|----------------------------|--|
| G | 6.67384E-11 (1.2E-4) | $\frac{m^3}{kg \cdot s^2}$ | Newton's gravitation constant. See GM below for a more precise value. |
| G_o | 7.7480917346E-5 (3.2E-10) | $1/\Omega$ | Conductance quantum $= 2e^2/h = 2/R_K$ with the v. Klitzing constant $R_K = 25812.8074434 \Omega$ |
| Gc | 0.915965594177... | 1 | Catalan's constant |
| g_e | 2.00231930436153 (2.6E-13) | 1 | (Landé's) electron g-factor |
| GM | 3.986004418E14 (2.0E-9) | m^3/s^2 | Newton's gravitation constant times the Earth's mass with its atmosphere included (according to WGS84, see Sa below). |
| h | 6.62606957E-34 (4.4E-8) | $J \cdot s$ | Planck constant |
| ħ | 1.054571726E-34 (4.4E-8) | | $= h/2\pi$ |
| k | 1.3806488E-23 (9.1E-7) | J/K | Boltzmann constant $= R/N_A$ |
| l_p | 1.616199E-35 (6.0E-5) | m | Planck length $= \sqrt{\hbar G/c^3} = t_p c$ |
| m_e | 9.10938291E-31 (4.4E-8) | kg | Electron mass |
| m_ec² | 8.18710506E-14 (4.4E-8) | J | |
| M_m | 7.349E22 (?) | kg | Mass of the Moon |
| m_n | 1.674927351E-27 (4.4E-8) | kg | Neutron mass |
| m_nc² | 1.505349631E-10 (4.4E-8) | J | |
| m_p | 1.672621777E-27 (4.4E-8) | kg | Proton mass |
| m_pc² | 1.503277484E-10 (4.4E-8) | J | |
| M_p | 2.17651E-8 (6.0E-5) | kg | Planck mass $= \sqrt{\hbar c/G} \approx 22 \mu g$ |
| m_u | 1.660538921E-27 (4.4E-8) | kg | Atomic unit mass $= 10^{-3} kg / N_A$ |
| m_uc² | 1.492 417 954E-10 (4.4E-8) | J | |
| m_μ | 1.883531475E-28 (5.1E-8) | kg | Muon mass |
| m_μc² | 1.692 833 667E-11 (5.1E-8) | J | |
| M_☉ | 1.9891E30 (?) | kg | Mass of the sun |

| | Numeric value | Unit | Remarks |
|------------------------|--------------------------------|-------------------------|---|
| M_⊕ | 5.9736E24 (?) | kg | Mass of the Earth |
| N_A | 6.02214129E23 (4.4E-8) | 1/mol | Avogadro's number |
| NaN | | | "not a number" |
| p_o | 101325 (per definition) | Pa | standard atmospheric pressure |
| q_p | 1,8755459E-18 (6.0E-5) | As | Planck charge = $\sqrt{4\pi\epsilon_0\hbar c} \approx 11.7e$. This was in CODATA 2006, but 2010 no more. |
| R | 8.3144621 (9.1E-7) | $\frac{J}{mol \cdot K}$ | Molar gas constant |
| r_e | 2.8179403267E-15 (9.7E-10) | m | Classical electron radius = $\alpha^2 \cdot a_0$ |
| R_m | 1737530 (?) | m | Mean radius of the Moon |
| R_∞ | 1.0973731568539E7 (5.0E-12) | 1/m | Rydberg constant = $\alpha^2 m_e c / 2h$ |
| R_⊙ | 6.96E8 (?) | m | Mean radius of the sun |
| R_⊕ | 6.371010E6 (?) | m | Mean radius of the Earth |
| Sa | 6.3781370E6 (per definition) | m | Semi-major axis of the model WGS84 used to define the Earth's surface for GPS and other surveying purposes (→ http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html) |
| Sb | 6.3567523142E6 (1.6E-11) | m | Semi-minor axis of WGS84 |
| Se² | 6.69437999014E-3 (1.5E-12) | 1 | First eccentricity squared of WGS84 |
| Se'² | 6.73949674228E-3 (1.5E-12) | 1 | Second eccentricity squared of WGS84 (it is really called e' ² in this article, I apologize) |
| Sf⁻¹ | 298.257223563 (per definition) | 1 | Flattening parameter of WGS84 |
| T_o | 273.15 (per definition) | K | = 0°C, standard temperature |
| t_p | 5.39106E-44 (6.0E-5) | s | Planck time = $\sqrt{\hbar G / c^5} = \frac{l_p}{c}$ |
| T_p | 1.416833E32 (6.0E-5) | K | Planck temperature $= \frac{c^2}{k} \sqrt{\frac{\hbar c}{G}} = \frac{M_p c^2}{k} = \frac{E_p}{k}$ |

| | Numeric value | Unit | Remarks |
|----------------|-----------------------------|---|--|
| V_m | 0.022413968 (9.1E-7) | m^3/mol | Molar volume of an ideal gas at standard conditions $= \frac{RT_0}{p_0}$ |
| Z_0 | 376.730313461... | Ω | Characteristic impedance of vacuum $= \sqrt{\frac{\mu_0}{\epsilon_0}} = \mu_0 c$ |
| α | 7.2973525698E-3 (3.2E-10) | 1 | Fine-structure constant $= \frac{e^2}{4\pi\epsilon_0\hbar c} \approx \frac{1}{137}$ |
| γ_{EM} | 0.57721566490153286... | 1 | Euler-Mascheroni constant |
| γ_p | 2.675222005E8 (2.4E-8) | $\frac{1}{s \cdot T}$ | Proton gyromagnetic ratio $= 2\mu_p/\hbar$ |
| ϵ_0 | 8.854187817...E-12 | $\frac{A \cdot s}{V \cdot m}$ or F/m | Electric constant, vacuum permittivity $= \frac{1}{\mu_0 c^2}$ |
| λ_c | 2.4263102389E-12 (6.5E-10) | m | Compton wavelength of the electron $= \hbar/m_e c$ |
| λ_{cn} | 1.3195909068E-15 (8.2E-10) | | Compton wavelength of the neutron $= \hbar/m_n c$ |
| λ_{cp} | 1.32140985623E-15 (7.1E-10) | | Compton wavelength of the proton $= \hbar/m_p c$ |
| μ_0 | 1.2566370614...E-6 | $\frac{V \cdot s}{A \cdot m}$ | Magnetic constant, also known as vacuum permeability $= 4\pi \cdot 10^{-7} \frac{V \cdot s}{A \cdot m}$ (per definition) |
| μ_B | 9.27400968E-24 (2.2E-8) | J/T or $A \cdot m^2$ | Bohr's magneton $= e\hbar/2m_e$ |
| μ_e | -9.28476430E-24 (2.2E-8) | | Electron magnetic moment |
| μ_n | -9.6623647E-27 (2.4E-7) | | Neutron magnetic moment |
| μ_p | 1.410606743E-26 (2.4E-8) | | Proton magnetic moment |
| μ_u | 5.05078353E-27 (2.2E-8) | | Nuclear magneton $= e\hbar/2m_p$ |
| μ_μ | -4.49044807E-26 (3.4E-8) | | Muon magnetic moment |

| | Numeric value | Unit | Remarks |
|------------|--------------------------|---------------------|--|
| π | 3.141592653589793... | 1 | |
| σ_B | 5.670373E-8 (3.6E-6) | $\frac{W}{m^2 K^4}$ | Stefan Boltzmann constant $= \frac{2\pi^5 k^4}{15h^3 c^2}$ |
| Φ | 1.61803398874989485... | 1 | Golden ratio $= \frac{1+\sqrt{5}}{2}$ |
| Φ_0 | 2.067833758E-15 (2.2E-8) | V s | Magnetic flux quantum $= \frac{h}{2e} = \frac{1}{K_J}$ with the Josephson constant $K_J = 4.83597870 \cdot 10^{14} \text{ Hz/V}$ |
| ω | 7.292115E-5 (2E-8) | rad/s | Angular velocity of the Earth according to WGS84 (see Sa above) |
| ∞ | | 1 | Infinity (may the Lord of Mathematics forgive us calling this a constant) |

UNIT CONVERSIONS

Find below the contents of the catalog CONV²⁵. Navigation works as in the other catalogs. There is one specialty, however: **f** **B** (i.e. $\frac{1}{x}$) will execute the inverse of the conversion displayed and leave CONV.

Example: Assume the display set to FIX 3. Then keying in

4 **h** **CONV** **A** will display **acres→ha** and 1.619 below telling you 4 acres equal 1.619 hectares.

Now press **f** **B** and you will get 9.884 instead, being the amount of acres equaling 4 hectares.

Press **h** **CONV** again and you will see **acres→ha** and 4.000 below confirming what was just said.

Leave the catalog via **EXIT** and the display will return to 9.884.

The constant **T₀** may be useful for conversions of temperatures, too; it is found in the [catalog CONST](#) and is not repeated here since being only added or subtracted. The conversion factors or divisors listed below for your information are user transparent in executing a conversion – those printed on light green background in this table 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 m ² | Area |
| ar.→dB | 10 * lg(R) | Amplitude ratio | Ratio |
| atm→Pa | * 1.01325E5 | | Pressure |
| AU→km | * 1.495979E8 | Astronomic units | Length |
| bar→Pa | * 1E5 | | Pressure |
| 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 |

²⁵ For most readers, many of the units appearing here may look obsolete at least. They die hard, however, in some corners of this world. All these corners have in common is English being spoken there. 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→ar. | $10^{R_{dB}/20}$ | | Ratio |
| dB→pr. | $10^{R_{dB}/10}$ | Power ratio | Ratio |
| fathom→m | * 1.8288 | | Length |
| feet→m | * 0.3048 | | Length |
| flozUK→ml | * 28.41306 | $1\text{ l} = \frac{1}{1000}\text{ m}^3$ | Volume |
| flozUS→ml | * 29.57353 | | |
| galUK→l | * 4.54609 | | |
| galUS→l | * 3.785418 | | |
| G→° | * 0.9 | Grads or gon | Angle |
| g→oz | / 28.34952 | | Mass |
| G→rad | * $\pi / 200$ | | Angle |
| g→tr.oz | / 31.10348 | | Mass |
| ha→acres | / 0.4046873 | $1\text{ ha} = 10^4\text{ m}^2$ | Area |
| HP _e →W | * 746 | Electric horse power | Power |
| hpUK→W | * 745.6999 | British horse power | Power |
| inches→cm | * 2.54 | | Length |
| inHg→Pa | * 3386.389 | | Pressure |
| J→Btu | / 1055.056 | | Energy |
| J→cal | / 4.1868 | | Energy |
| J→eV | / 1.602E-19 | | Energy |
| J→kWh | / 3.6E6 | | Energy |
| kg→lb | / 0.4535924 | | Mass |
| kg→stones | / 6.35029318 | | Mass |
| km→AU | / 1.495979E8 | Astronomic units | Length |
| km→l.y. | / 9.460730E12 | Light years | Length |
| km→miles | / 1.609344 | | Length |
| km→nmi | / 1.852 | Nautical miles | Length |
| km→pc | / 3.085678E16 | Parsec | Length |

| Conversion | | Remarks | Class |
|------------|---------------|---------------------------------------|----------|
| kWh→J | * 3.6E6 | | Energy |
| lbf→N | * 4.448222 | | Force |
| lb→kg | * 0.4535924 | | Mass |
| l.y.→km | * 9.460730E12 | Light years | Length |
| l →cft | / 28.31685 | 1 l = $\frac{1}{1000}$ m ³ | Volume |
| l →galUK | / 4.54609 | | |
| l →galUS | / 3.785418 | | |
| miles→km | * 1.609344 | | Length |
| ml→flozUK | / 28.41306 | 1 ml = 1 cm ³ | Volume |
| ml→flozUS | / 29.57353 | | |
| mmHg→Pa | * 133.3224 | 1 torr = 1 mm Hg | Pressure |
| m→fathom | / 1.8288 | | Length |
| m→feet | / 0.3048 | | Length |
| m→yards | / 0.9144 | | Length |
| nmi→km | * 1.852 | Nautical miles | Length |
| N→lbf | / 4.448222 | | Force |
| oz→g | * 28.34952 | Ounces | Mass |
| Pa→atm | / 1.01325E5 | 1 Pa = 1 N/m ² | Pressure |
| Pa→bar | / 1E5 | | Pressure |
| Pa→inHg | / 3386.389 | | Pressure |
| Pa→mmHg | / 133.3224 | | Pressure |
| Pa→psi | / 6894.757 | | Pressure |
| Pa→torr | / 133.3224 | | Pressure |
| pc→km | * 3.085678E16 | 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 |

| Conversion | | Remarks | Class |
|-------------------|--------------|--------------------------|----------|
| stones→kg | * 6.35029318 | | Mass |
| 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 |
| tr.oz→g | * 31.10348 | Troy ounces | Mass |
| t→s.tons | / 0.9071847 | 1 t = 10 ³ kg | Mass |
| t→tons | / 1.016047 | | |
| W→HP _e | / 746 | | Power |
| W→hpUK | / 745.6999 | | Power |
| W→PS(hp) | * 735.4988 | | Power |
| yards→m | * 0.9144 | | Length |

You may, of course, combine conversions as you like. For example, filling your tires with a maximum pressure of 30 psi the following will help you at a gas station in Europe and beyond:

resulting in 2.1 bar.

Now you can set the filler and will not blow your tires.

In cases of emergency of a particular kind, remember Becquerel equals Hertz, Gray is the unit for deposited or absorbed energy ($1\text{Gy} = 1\text{J/kg}$), 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, $1\text{Ci} = 3.7 \cdot 10^{10} \text{Bq} = 3.7 \cdot 10^{10} \text{decays/s}$. 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 $1\text{R} = 2.58 \cdot 10^{-4} \text{As/kg}$. 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. You find them listed in CAT. The respective routines are located in XROM, thus not taking any steps from user program memory. The following global labels are used:

| | |
|-----|---|
| TVM | <p>Time Value of Money almost as known since the <i>HP-80</i>. This routine contains the equation</p> $PV \cdot (1 + I)^n + PMT \cdot \frac{k}{I} \cdot [(1 + I)^n - 1] + FV = 0 \quad \text{with}$ $PMT = -\frac{I}{k} \cdot \left[PV + \frac{PV + FV}{(1 + I)^n - 1} \right] = \text{periodic payment} = r80,$ $PV = \frac{PMT \cdot \frac{k}{I} - FV}{(1 + I)^n} - PMT \cdot \frac{k}{I} = \text{present value} = r81,$ $FV = PMT \cdot \frac{k}{I} - (1 + I)^n \cdot \left(PMT \cdot \frac{k}{I} + PV \right) = \text{future value} = r82,$ <p><i>I</i> = interest rate per period = r83, <i>n</i> = number of periods = r84, <i>k</i> = 1 if payment is made at the end of each period = flag 80 clear, = 1 + <i>I</i> if it is made at the beginning of each period = flag 80 set.</p> <p>Store all you know and solve for the unknown. E.g. solving for PMT may look like:</p> <pre> LBL 'PMT' ;routine is entered with a first guess in X. SLV 01 NOP ;this step must be included since SLV acts as a test. RTN LBL 01 STO 80 ;initial or previous guess XEQ 'TVM' RTN </pre> <p>See SLV for more.</p> |
| WHO | Displays credits. |
| δx | Provides the step size for differentiation. See f'(x) and f''(x) for more information. |


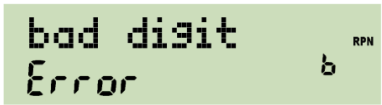
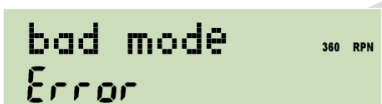
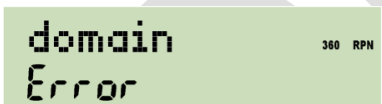


MESSAGES

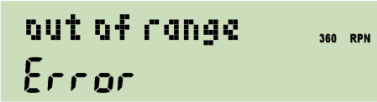


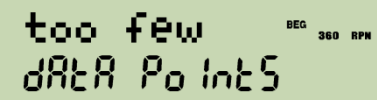


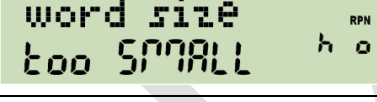
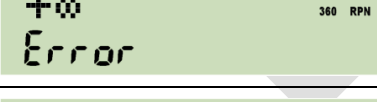
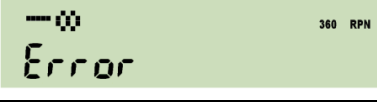

There are some commands generating messages, also in the dot matrix section of the display. Four of them, DAY, DAYS+, STATUS, and VERS, were introduced above in the [paragraph about display](#) already. Others are PROMPT, α VIEW and many more alpha commands, and the test commands as mentioned [above](#).

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

 or , 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 |
|---|------------|----------|--|
|  | 2 | DECM | Invalid date format or incorrect date in input, e.g. month >12, day >31 etc. |
|  | 9 | Integer | Invalid digit in integer input, e.g. 2 in binary, 9 in octal, or +/- in unsigned mode. |
|  | 13 | All | Caused by calling an operation in a mode where it is not defined, e.g. SIN in hexadecimal. |
|  | 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 CPX), by $0/0$, $\text{LN}(0)$, $\Gamma(0)$, $\text{TAN}(90^\circ)$ and equivalents, $\text{ATANH}(x)$ for $ \text{Re}(x) \geq 1$, $\text{ACOSH}(x)$ for $\text{Re}(x) < 1$, etc. |
|  | 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 $j < 0$. |
|  | 6 | All | Attempt to address an undefined label. |

| Message | Error Code | Mode(s) | Explanation and Examples |
|---|------------|---------------|---|
|  | 8 | All | <ul style="list-style-type: none"> 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). |
|  | 7 | PRG | Nested use of solve, integrate, sum or product is not allowed. |
|  | 12 | All | STOS or RCLS attempt using registers that would overlap the stack. Will happen with e.g. SSIZE = 8 and STOS 94. |
|  | 15 | DECM | A statistical calculation was started based on too few data points, e.g. regression or standard deviation for < 2 points. |
|  | 10 | All | Keyboard input is too long for the buffer (should never happen, but who knows). |
|  | 3 | All | An instruction with an undefined op-code occurred (should never happen, but who knows). |
|  | 14 | Integer, \PRG | Stack or register content is too big for the word size set. |
|  | 4 | \α, \PRG | <ul style="list-style-type: none"> Division of a number > 0 (or < 0) by zero. Divergent sum or product or integral. Positive (or negative) overflow in DECM (see above). |
|  | 5 | | |
|  | 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.

PROGRAMMED INPUT AND OUTPUT

A number of commands may be employed for controlling I/O of programs. In the index [above](#), their behavior is described if they are entered from the keyboard. Executed by a program, however, this will differ in a characteristic way.

With a program running, the display will be updated at certain instances only instead of after each operation. So where a command in manual mode shows an information until the next key is pressed, it will show it until the next display update in automatic mode. Such an update will occur with PROMPT, PSE, STOP, VIEW, VW α +, and α VIEW only. This allows for the following operations:

- Output of messages or other information for a defined time interval using the following code segment
...
VIEW
PSE
...
(or simply PSE alone) for plain numeric calculated output or
...
 α VIEW (or even VW α +) *alpha*
PSE
...
for complex alphanumeric information you composed in *alpha*.
- Asking (“prompting”) for numeric input employing
...
 α VIEW (or VW α +) *alpha*
STOP
...
or simply PROMPT, the latter being identical to
VW α + X
STOP
Whatever number you key in will be in X when you continue the program by pressing **R/S** . If you want it elsewhere, take care of it.
- Prompting for alphanumeric input by
...
 α ON
PROMPT
 α OFF
...
Whatever you key in will be appended to *alpha* here. Again, the program will continue when you pressed **R/S** .

Have fun!

APPENDIX A: SUPPORT COMMANDS FOR SERIAL I/O ETC.

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 and Gene Wright. And take care of the following two disclaimers:

ATTENTION 1: If your PC does not feature a serial interface, you will need an USB-to-serial converter to connect the special cable to your PC. Following our experience, those converters containing FTDI chips will work – others may not.

ATTENTION 2: Flashing your 20b or 30b with the WP 34S file `calc.bin` will overwrite the HP firmware in said calculator completely! After this flash finished, you will have a WP 34S – i.e. it will react as documented in this very manual – no more an HP 20b or 30b!

After flashing, a keyboard overlay is very helpful for further work since most labels deviate from the ones used on said business calculators. You can get adhesive overlays from Eric Rechlin.

Commands for handling the flash memory on your WP 34S

Flash memory is very useful for backups as explained [above](#). Alternatively to the commands SAVE and LOAD contained in X.FCN (see the [index of operations](#)), you may use another approach. Hold down ON (i.e. **EXIT**) and press one of the following keys:

STO for backup: Creates a copy of the RAM in flash memory like SAVE does.

RCL for restore: Restores the most recent backup like LOAD does.

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 will be lost!

These ON key combinations have to be pressed twice in a row without releasing the ON key to be executed.

We recommend doing a SAVE or ON+STO before flashing a new release! After flashing, your backup will still be available – if you used ON+S to get into SAM-BA boot mode and didn't accidentally press the ERASE button on the cable.

Further commands for flash memory operations are PRCL, PSTO, P↔, RCF, RRCL, and SRCL. See the [index of operations](#).

Mapping of memory regions to emulator state files

| Region | Start address in flash | State file | Remarks |
|---------|------------------------|-------------|---|
| Unnamed | 0x11FC00 | wp34s-R.dat | Backup of registers and state |
| 0 | 0x11F800 | wp34s-0.dat | Backup of program memory |
| 1 | 0x11F400 | wp34s-1.dat | Space for generic user programs. |
| 2 | 0x11F000 | wp34s-2.dat | The files <code>wp34s-x.dat</code> are written whenever a respective flash command is executed. |
| 3 | 0x11EC00 | wp34s-3.dat | |

| Region | Start address in flash | State file | Remarks |
|--------|------------------------|------------|---|
| RAM | n/a | wp34s.dat | Backup of the emulator RAM area (registers, state, and programs) – this file is written only when exiting the emulator. |

All files are only read into memory at emulator startup.

Data transfer between your WP 34S and your PC (method 1)

The entire RAM is saved to address `0x11F800` (relative address `0x1F800`) by **SAVE** or **ON + [STO]**. This content can be copied to your PC or loaded from it if the special interface cable mentioned above is connected.

1. From calculator to PC:

- Press **ON + [STO]**, then **ON + [D]** (see below), then **ON + [S]**,
- Press **ON** once and start SAM-BA on the PC. Both devices should connect.
- Set the start address to `0x11F800` and the size to `0x800`.
- Enter a file name of your choice in the receive field. You can now receive the file with SAM-BA.
- Move it into your emulator directory (where `wp34sgui.exe` is stored) under the name `wp34s.dat`.
- The emulator should accept the file. Your registers and programs will then be in place.
- To get your calculator back in business, start the "Boot from flash" script in SAM-BA – the same procedure you should know from flashing the firmware.
- Press **ON** to power up. The most recent backup (i.e. the one of step a. here) will be automatically restored. If not, then the backup area in flash is no longer valid (most probably you have accidentally pressed the ERASE button on the cable). You can still try **LOAD** or **RCLS**.

2. From PC to calculator:

- Execute steps 1.a + b.
- Set the start address to `0x11F800`.
- Point SAM-BA to your `wp34s.dat` file from the emulator.
- You can now send the short file with SAM-BA.
- Execute steps 1.g + h.

The program regions accessible with the commands **PSTO**, **PRCL** and **P↵** are stored at addresses `0x11F000` and above (see the table above) and have a length of `0x400` (1 kB) each. The emulator creates files `wp34s-x.dat`, with **x** being the region number. You can handle these files the same way as the complete state file from the emulator. The regions have identical formatting and can be swapped by copying their data to the 'wrong' place. The register and state portion of the backup area at `0x11FC00` is formatted differently.

If you want to get your emulator data from your PC into your calculator all in once, do the following in Windows:

```
copy /b calc.bin+wp34s-3.dat+wp34s-2.dat+wp34s-1.dat+wp34s.dat  
calc-full.bin
```

As an alternative, the following will copy the backup data instead of the RAM state file:

```
copy /b calc.bin+wp34s-3.dat+wp34s-2.dat+wp34s-1.dat+wp34s-0.dat  
+wp34s-R.dat calc-full.bin
```

The resulting file can be transferred into flash as described in sequence 2 and all data will be readily available.

Data transfer between your WP 34S and your PC (method 2)

You will need the special interface cable mentioned above again or a [modified 20b or 30b](#) as described elsewhere. The original cable provided by HP draws power from the batteries and should be disconnected from the calculator while not in use.

Communication is between your WP 34S and another WP 34S. The Windows emulator counts as a valid partner so you can exchange data between your WP 34S and the PC. Since PCs tend to have more than one port it is necessary to tell the emulator which one to use. Create a file "wp34s.ini" in the directory where the state files reside and put the name of the port as the only line in this file. Mine says "COM9:" without the quotes – it is the very same port SAM-BA uses to access my WP 34S for flashing.

The following commands allow for sending programs, registers or all RAM. They are found in the X.FCN catalog.

Start the command RECV on the receiving device. It will display "Wait..".

On the sender you have three choices:

1. SENDP will send the user program space. After successful termination, the receiver will display "Program".
2. SENDR will send the registers 00 to 99. The receiver will display "Register" after successful termination.
3. SENDA will send the complete 2 KB of non volatile memory. The receiver will display "All RAM" after successful termination.

The commands for sending and receiving feature a fixed timeout of some 10 seconds for setting up the connection. After an interval of inactivity of said length, "I/O Error" is displayed indicating no communication has occurred. If "I/O Error" appears in the middle of a transmission try again.

The little "=" annunciator is lit while the serial port is in use. **EXIT** can be used to abort the communication.

More keyboard commands employing ON

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

+ or **-** : Adjust display contrast.

C : Tells the system a quartz crystal 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!

D Enters debugging mode (use at your own risk).

. : Toggles 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 <u>n</u> | <p>Recalls internal constants, selected by the number specified:</p> <table><tr><td>0</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr><tr><td>2</td><td>5.01402</td><td>Kronrod only weight loop initializer (constants 5 - 14 below)</td></tr><tr><td>3</td><td>15.02903</td><td>Gauss-Kronrod weight loop initializer (constants 15 - 29 below)</td></tr><tr><td></td><td></td><td>Midpoint location is 0.5.</td></tr><tr><td>4</td><td>0.149445554002916905664936468389821</td><td>Kronrod weight for midpoint k10</td></tr><tr><td>5</td><td>0.995657163025808080735527280689003</td><td>Kronrod location of k0 and k20</td></tr><tr><td>6</td><td>0.011694638867371874278064396062192</td><td>Kronrod weight for k0 and k20</td></tr><tr><td>7</td><td>0.930157491355708226001207180059508</td><td>Kronrod location of k2 and k18</td></tr><tr><td>8</td><td>0.054755896574351996031381300244580</td><td>Kronrod weight for k2 and k18</td></tr><tr><td>9</td><td>0.780817726586416897063717578345042</td><td>Kronrod location of k4 and k16</td></tr><tr><td>10</td><td>0.093125454583697605535065465083366</td><td>Kronrod weight for k4 and k16</td></tr><tr><td>11</td><td>0.562757134668604683339000099272694</td><td>Kronrod location of k6 and k14</td></tr><tr><td>12</td><td>0.123491976262065851077958109831074</td><td>Kronrod weight for k6 and k14</td></tr><tr><td>13</td><td>0.294392862701460198131126603103866</td><td>Kronrod location of k8 and k12</td></tr><tr><td>14</td><td>0.142775938577060080797094273138717</td><td>Kronrod weight for k8 and k12</td></tr><tr><td>15</td><td>0.973906528517171720077964012084452</td><td>Location of g0, g9, k1 and k19</td></tr><tr><td>16</td><td>0.066671344308688137593568809893332</td><td>Gauss weight for g0 and g9</td></tr><tr><td>17</td><td>0.032558162307964727478818972459390</td><td>Kronrod weight for k1 and k19</td></tr><tr><td>18</td><td>0.865063366688984510732096688423493</td><td>Location of g1, g8, k3 and k17</td></tr><tr><td>19</td><td>0.149451349150580593145776339657697</td><td>Gauss weight for g1 and g8</td></tr><tr><td>20</td><td>0.075039674810919952767043140916190</td><td>Kronrod weight for k3 and k17</td></tr><tr><td>21</td><td>0.679409568299024406234327365114874</td><td>Location of g2, g7, k5 and k15</td></tr><tr><td>22</td><td>0.219086362515982043995534934228163</td><td>Gauss weight for g2 and g7</td></tr><tr><td>23</td><td>0.109387158802297641899210590325805</td><td>Kronrod weight for k5 and k15</td></tr><tr><td>24</td><td>0.433395394129247190799265943165784</td><td>Location of g3, g6, k7 and k13</td></tr><tr><td>25</td><td>0.269266719309996355091226921569469</td><td>Gauss weight for g3 and g6</td></tr><tr><td>26</td><td>0.134709217311473325928054001771707</td><td>Kronrod weight for k7 and k13</td></tr><tr><td>27</td><td>0.148874338981631210884826001129720</td><td>Location of g4, g5, k9 and k11</td></tr><tr><td>28</td><td>0.295524224714752870173892994651338</td><td>Gauss weight for g4 and g5</td></tr><tr><td>29</td><td>0.147739104901338491374841515972068</td><td>Kronrod weight for k9 and k11</td></tr></table> <p>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 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.</p> | 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 | 0.149445554002916905664936468389821 | Kronrod weight for midpoint k10 | 5 | 0.995657163025808080735527280689003 | Kronrod location of k0 and k20 | 6 | 0.011694638867371874278064396062192 | Kronrod weight for k0 and k20 | 7 | 0.930157491355708226001207180059508 | Kronrod location of k2 and k18 | 8 | 0.054755896574351996031381300244580 | Kronrod weight for k2 and k18 | 9 | 0.780817726586416897063717578345042 | Kronrod location of k4 and k16 | 10 | 0.093125454583697605535065465083366 | Kronrod weight for k4 and k16 | 11 | 0.562757134668604683339000099272694 | Kronrod location of k6 and k14 | 12 | 0.123491976262065851077958109831074 | Kronrod weight for k6 and k14 | 13 | 0.294392862701460198131126603103866 | Kronrod location of k8 and k12 | 14 | 0.142775938577060080797094273138717 | Kronrod weight for k8 and k12 | 15 | 0.973906528517171720077964012084452 | Location of g0, g9, k1 and k19 | 16 | 0.066671344308688137593568809893332 | Gauss weight for g0 and g9 | 17 | 0.032558162307964727478818972459390 | Kronrod weight for k1 and k19 | 18 | 0.865063366688984510732096688423493 | Location of g1, g8, k3 and k17 | 19 | 0.149451349150580593145776339657697 | Gauss weight for g1 and g8 | 20 | 0.075039674810919952767043140916190 | Kronrod weight for k3 and k17 | 21 | 0.679409568299024406234327365114874 | Location of g2, g7, k5 and k15 | 22 | 0.219086362515982043995534934228163 | Gauss weight for g2 and g7 | 23 | 0.109387158802297641899210590325805 | Kronrod weight for k5 and k15 | 24 | 0.433395394129247190799265943165784 | Location of g3, g6, k7 and k13 | 25 | 0.269266719309996355091226921569469 | Gauss weight for g3 and g6 | 26 | 0.134709217311473325928054001771707 | Kronrod weight for k7 and k13 | 27 | 0.148874338981631210884826001129720 | Location of g4, g5, k9 and k11 | 28 | 0.295524224714752870173892994651338 | Gauss weight for g4 and g5 | 29 | 0.147739104901338491374841515972068 | Kronrod weight for k9 and k11 |
| 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 | 0.149445554002916905664936468389821 | Kronrod weight for midpoint k10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | 0.995657163025808080735527280689003 | Kronrod location of k0 and k20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | 0.011694638867371874278064396062192 | Kronrod weight for k0 and k20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | 0.930157491355708226001207180059508 | Kronrod location of k2 and k18 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 0.054755896574351996031381300244580 | Kronrod weight for k2 and k18 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | 0.780817726586416897063717578345042 | Kronrod location of k4 and k16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | 0.093125454583697605535065465083366 | Kronrod weight for k4 and k16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | 0.562757134668604683339000099272694 | Kronrod location of k6 and k14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 0.123491976262065851077958109831074 | Kronrod weight for k6 and k14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | 0.294392862701460198131126603103866 | Kronrod location of k8 and k12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | 0.142775938577060080797094273138717 | Kronrod weight for k8 and k12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | 0.973906528517171720077964012084452 | Location of g0, g9, k1 and k19 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | 0.066671344308688137593568809893332 | Gauss weight for g0 and g9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | 0.032558162307964727478818972459390 | Kronrod weight for k1 and k19 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 18 | 0.865063366688984510732096688423493 | Location of g1, g8, k3 and k17 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 19 | 0.149451349150580593145776339657697 | Gauss weight for g1 and g8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | 0.075039674810919952767043140916190 | Kronrod weight for k3 and k17 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | 0.679409568299024406234327365114874 | Location of g2, g7, k5 and k15 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | 0.219086362515982043995534934228163 | Gauss weight for g2 and g7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 23 | 0.109387158802297641899210590325805 | Kronrod weight for k5 and k15 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 24 | 0.433395394129247190799265943165784 | Location of g3, g6, k7 and k13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | 0.269266719309996355091226921569469 | Gauss weight for g3 and g6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 26 | 0.134709217311473325928054001771707 | Kronrod weight for k7 and k13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 27 | 0.148874338981631210884826001129720 | Location of g4, g5, k9 and k11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 28 | 0.295524224714752870173892994651338 | Gauss weight for g4 and g5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 29 | 0.147739104901338491374841515972068 | Kronrod weight for k9 and k11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Name | Purpose and remarks |
|----------|---|
| | <p>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.</p> <p>Registers:</p> <ul style="list-style-type: none"> $n+0$.. $n+1$: 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 $f(b)$ <p>Flags:</p> <ul style="list-style-type: none"> $n+0$.. $n+7$: 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. |
| | <p>The built in solver loop looks like this in principle, assuming $n = 0$:</p> <pre> 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 $f(c)$ $x \approx 0?$; 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 </pre> <p>The actual solver is fairly complex. A combination of quadratic interpolation and a guarded secant method is used.</p> |
| XEQUSR | Calls a user subroutine (used by SLV, \int , Π and Σ). The subroutine is defined by the argument to the initial command (either numeric or alpha label). |

APPENDIX B: 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 \blacktriangle and \blacktriangledown , 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_6 to the table of constants; put CLSTK on a key, moved CL Σ 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, α DAY, α MONTH, α RC#, Σ , as well as F-, t-, and χ^2 -distributions and their inverses; reassigned DATE, modified DENMAX, FLOAT, α ROT, and α SHIFT; 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, $I\beta$, $I\Gamma$, α DATE, α RL, α RR, α SL, α SR, α TIME, 12h, 24h, fraction mode limits, normal distribution and its inverse for arbitrary μ and σ , and Boolean operations working within FLOAT; deleted α ROT, α SHIFT, 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 $\Delta\%$ 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 $\sqrt{}$ 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, $^{\circ}$ ENTER, α BEG, α END, 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?, $^{\circ}$ DROP, $^{\circ}$ FILL, $^{\circ}$ R \downarrow , $^{\circ}$ R \uparrow , 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, α OFF, α ON, Π , and Σ ; renamed ROUNDI, WSIZE?, $\beta(x,y)$, $\Gamma(x)$ and the constant p_0 ; deleted DROPY (use $x\leftrightarrow y$, DROP instead), α APP, α BEG, α END, 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 α TIME, 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 | <p>Started the Windows emulator.</p> <p>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.</p> |
| 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 | <p>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 . <i>hotkey</i>, KEY?, RAD→, SCIOVR, SERRw, SLVQ, sw, sxy, TICKS, TVM, x_g, ε, ε_m, ε_p, ζ, σ_w, $(-1)^x$, 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 key-stroke; made D.MY default instead of Y.MD; moved degrees to radians conversions to CONV; removed cCLx, H.MS mode, %+ and %-; corrected errors.</p> |
| 1.18 | 5.6.11 | <p>Expanded program memory; modified label addressing ($A \neq 'A'$) and fraction mode limits, changed ANGLE to work in real and complex domains, renamed MOD to RMDR, changed the keyboard layout; put BACK, ERR, SKIP, and SPEC? to the main index; added CAT and the I/O commands for flash memory, expanded R-COPY; corrected $x \rightarrow \alpha$.</p> |
| 2.0 | 21.7.11 | <p>Entered beta test phase.</p> <p>Added DAY, MONTH, YEAR, FAST, SLOW, S.L, S.R, VWα+, flag A, ON + and –, some constants, and a paragraph about I/O; renamed old DAY to WDAY, RRCL to RCFRG, SRCL to RCFST; added an inverse conversion shortcut, stones↔kg, and changed Pa↔mbar to Pa↔bar; modified the VIEW commands, ALL, DISP, MODE, RCLM, STOM, and X.FCN; repaired hyperlinks; corrected some errors; included flash.txt; updated the first chapters, explained stack mechanics in more detail.</p> |
| 2.1 | 27.7.11 | Added serial I/O commands and some constants, updated the values in CONST to CODATA 2010. |