34S OWNER’S MANUAL

TABLE OF CONTENTS

WELCOME

Dear user, you hold in your hands the result of careful customizing. The 34S mechanics and hardware are of the new HP-30b Business Professional as is, while its firmware and user interface are newly written from scratch to make this little device a **compact scientific calculator like you have never had before**.

The function set of the 34S is based on the renowned HP-42S RPN Scientific, the most powerful RPN calculator built so far [1]. This set is extended and completely includes the functionality of the famous HP-16C, the fraction mode of the HP-32SII, statistical distributions as featured by the HP-21S, and even **more functions for mathematics, statistics, physics, engineering, programming etc.** like

- Euler's Beta function, Fibonacci number calculation, Lambert's W (all in real and complex domain), incomplete regularized Beta and Gamma,
- the error function as well as many distributions and their inverses: Poisson, binomial, geometric as well as exponential, Weibull for reliability analysis, and Gaussian with arbitrary mean and standard deviation,
- extended date and time calculations based on a real time clock,
- financial operations like mean rate of return or margin calculations,
- nearly 50 fundamental physical constants as precise as known by renowned institutes like NIST,
- over 50 different conversions between SI and old Imperial units,
- complete Greek and extended Latin letter fonts (upper and lower case in two sizes each).

And the 34S is the **first RPN calculator overcoming the limits of a 4-level stack** – forget worries about stack overflow in calculations. It features selectable stack size expanded by a complex LASTx register: traditional 4 stack levels for HP compatibility, 8 levels for easy calculations in complex domain, more advanced formulas, or whatever application you have in your mind. Furthermore, the 34S features over 100 general purpose registers, 476 program steps, 3 programmable hotkeys, 100 user flags, and a 31 byte alpha register for message generation.

If you know how to deal with a good old HP RPN calculator, you can start with your 34S right away. To show you the features of the 34S completely, however, we wrote this little manual. It starts with an overview on the active keyboard in various modes, so you know where to find what you are looking for. It continues with tables about addressing, navigating the catalogues and a paragraph about the display and indicators used to tell you what's going on. The major part of this booklet is taken by the index of operations, catalogue contents, constants and conversions provided. It closes with a list of messages the 34S will display if special input conditions prevent it from executing your command as expected.

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

Your 34S is the result of a collaboration of two individuals, an Australian and a German, though we did this in our free time, and so you may call it our hobby to some extent. We baptized it 34S in honour of one of the most powerful LED pocket calculators, the HP-34C, and since the 34S is our humble approach – with the hardware given – to a future 43S we can only dream of so far.

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

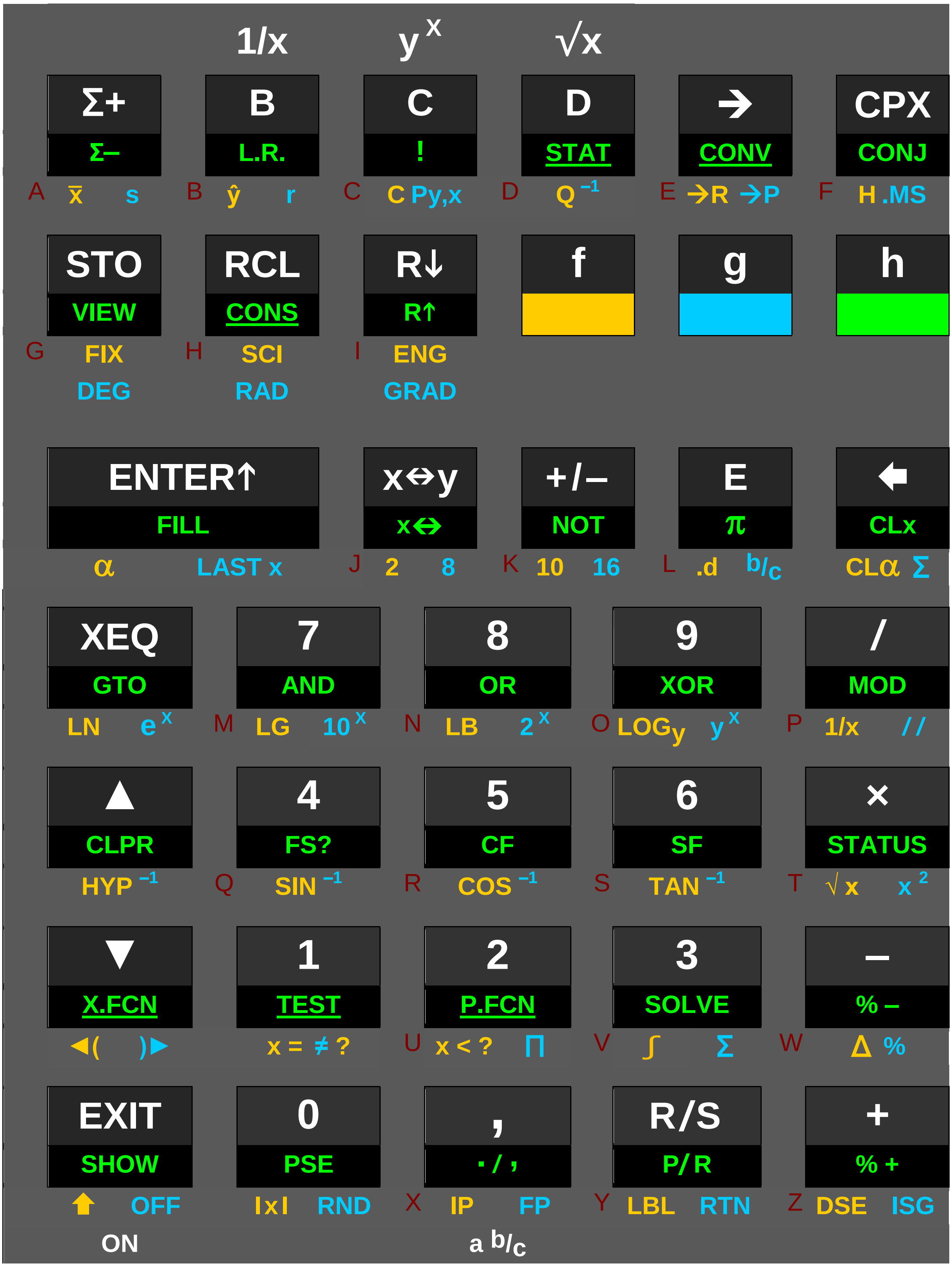
Enjoy!

Paul Dale and Walter Bonin

PRINT CONVENTIONS

Throughout this manual, commands are generally called by their names, usually written in CAPITALS. This CPX font is taken for explicit references to keys. Registers like stack level **X** or general purpose register **R23** are printed using Times New Roman. Lower case italic letters are taken for register contents (e.g. ***y*** or ***r45*** or ***alpha*** for the contents of **Y** or **R45** or the alpha register, respectively) as well as for variables. All this holds unless stated otherwise explicitly.

KEYBOARD



- Labels underlined open catalogues.
- [B], [C], and [D] directly call the user programs carrying these labels.

- Prefix ➡ combined with H, H.MS, DEG, RAD, GRAD, 2, 8, 10, 16 converts ***x***, while →R and →P convert polar and rectangular coordinates in ***x*** and ***y*** as usual.

- The complex prefix **CPX** may be combined with many functions. Names in listings will be merged, e.g. **CPX** **f** **COS** will be shown as ^{C}COS. Generally, where an arbitrary real function **f** works with ***x*** alone, its complex sibling ^{C}**f** works with $x_c = x + i\,y$. Where **f** uses ***x*** and ***y*** , thus ^{C}**f** uses 4 stack levels. Where **f** operates on one register ***r*** , ^{C}**f** operates on 2 registers at addresses ***r*** and ***r*** + *1* .

  In mathematical operations, one-number real functions replace ***x*** by the result **f(*x*)** stored in **X** again. The respective complex function replaces ***x*** by the real and ***y*** by the imaginary part of the complex result ^{C}**f(*x_c*)**. Higher stack levels remain unchanged. Such complex functions are ^{C}1/x, ^{C}ABS, ^{C}FIB, ^{C}FP, ^{C}IP, ^{C}ROUND, ^{C}SIGN, ^{C}W, ^{C}W^{-1}, ^{C}x!, ^{C}x^{2}, ^{C}√‾, ^{C}**+/-**, ^{C}Γ(x), logarithmic and exponential with bases 10, 2 and e, as well as hyperbolic, trigonometric and their inverses.

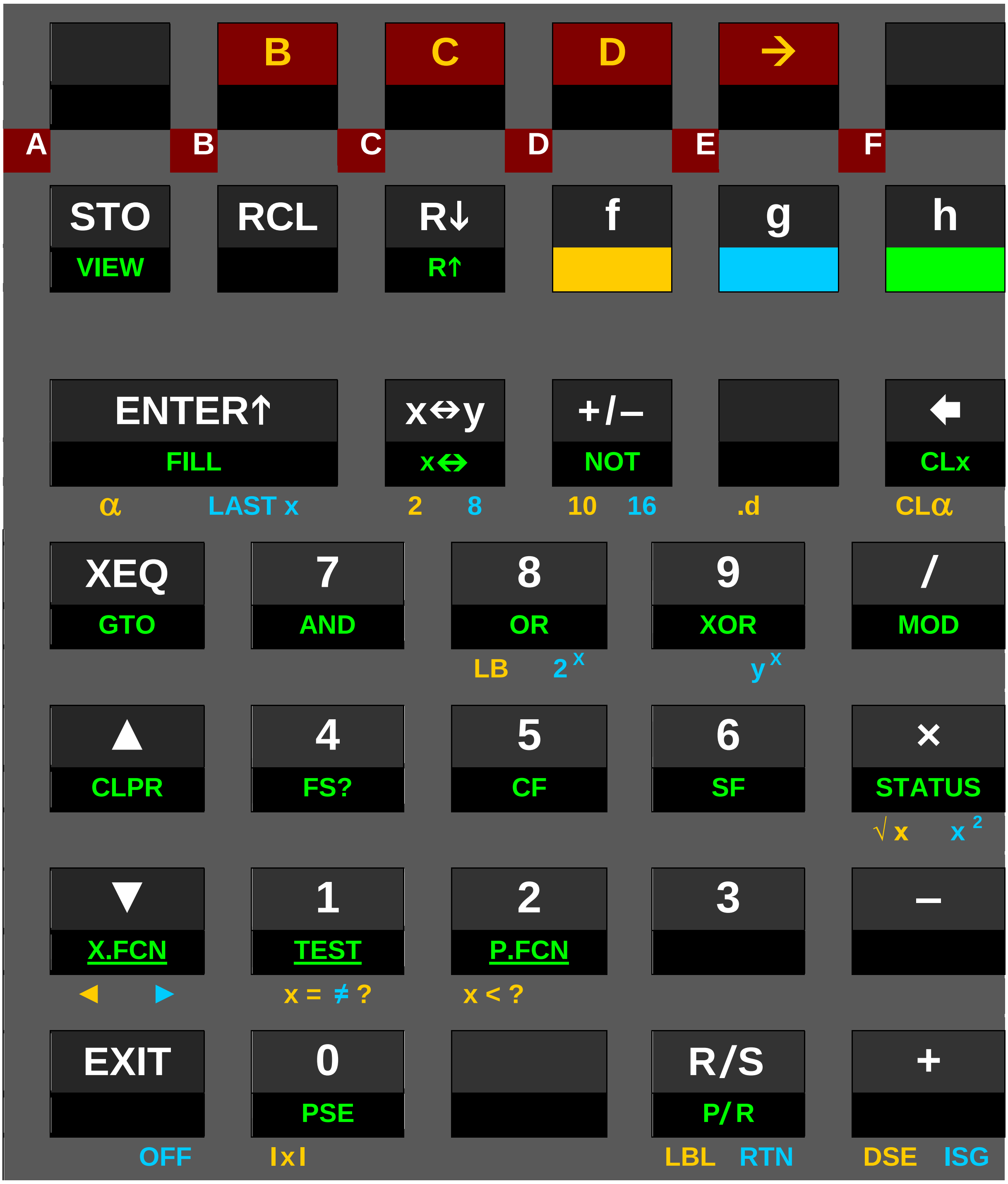
  Two-number real functions replace ***x*** by the result **f(*x*,*y*)** stored in **X**. Level **Y** is filled with the content of the next higher level. This holds for higher levels as well, only the content of the top level is repeated as shown below in a *stack diagram*. – The respective complex functions replace ***x*** by the real and ***y*** by the imaginary part of the complex result ^{C}**f(x_c,y_c)**. The next stack levels are filled with the contents of higher levels as shown below in some *stack diagrams*. Complex two-number functions are ^{C}LOGy, ^{C}y^{x}, ^{C}β(x,y), ^{C}//, and the basic arithmetic operations.

  There is one three-number real function – %MRR – included replacing ***x*** by the result **f(*x*,*y*,*z*)** stored in **X**. There, **Y** is filled with ***t*** and so on, and the content of the top level is repeated twice. No such complex function is featured.

- Calculator modes are as described in the *paragraph about indicators* below.

Please see the *index of operations* for more.

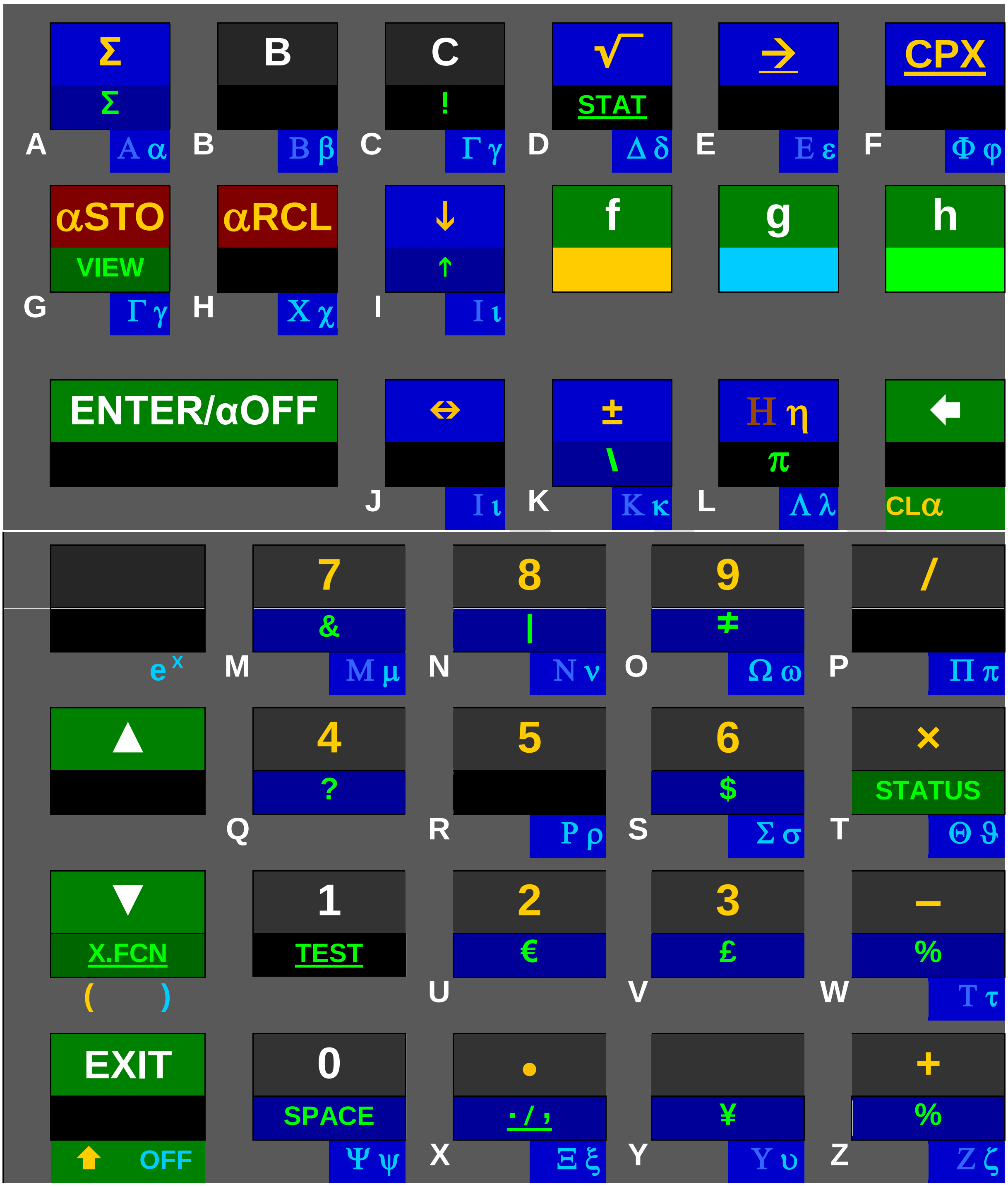
Virtual active keyboard in **hexadecimal mode**:



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

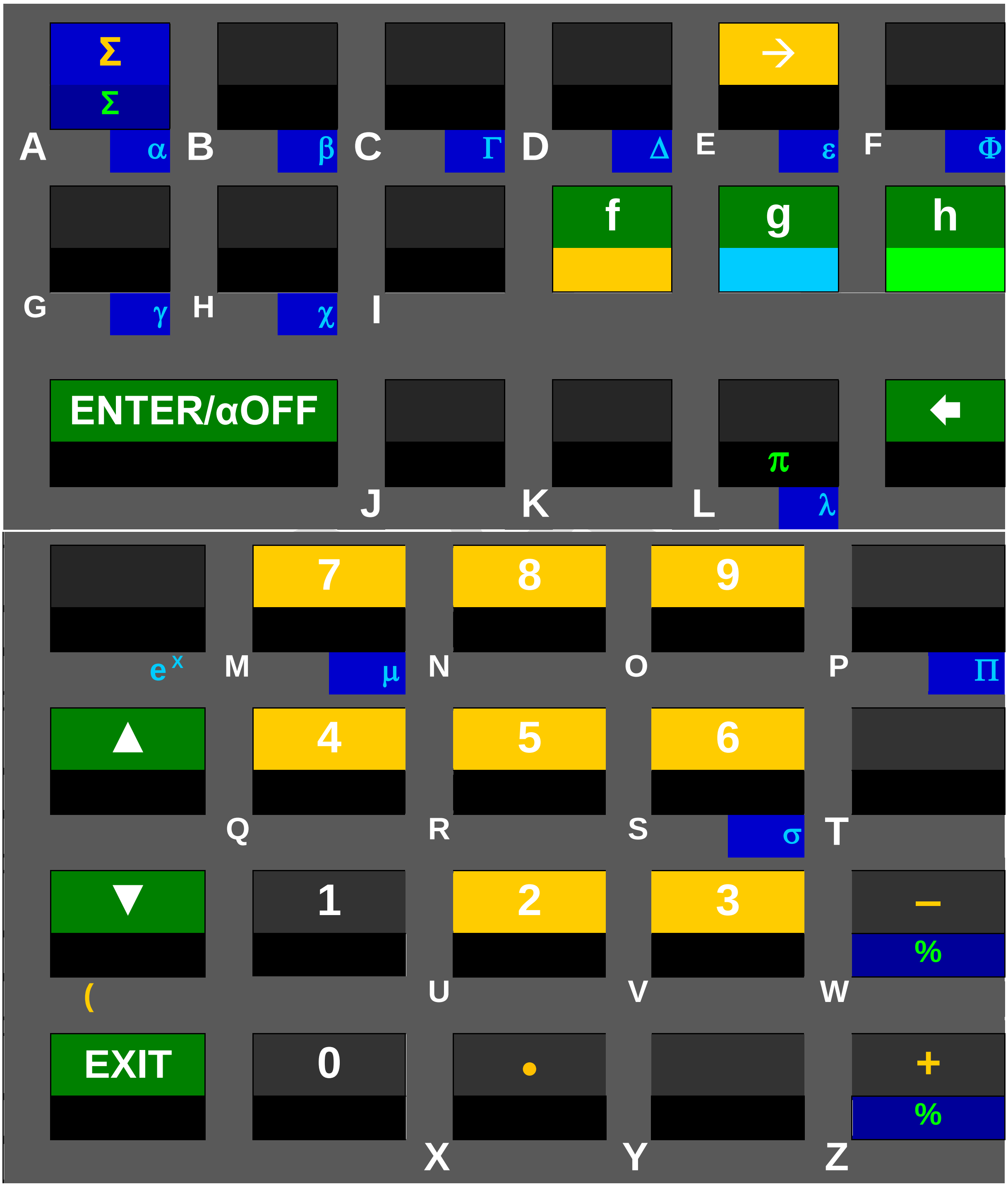
In the other integer bases, the active keyboard will look alike, but those top keys not needed for numeric input there will keep their default primary functions, except Σ+ and CPX . Attempts to enter an illegal digit will throw an *error*.

Virtual active keyboard in **alpha mode:**



Therein, ***alpha*** is displayed in the dot matrix, and the numeric line is accessible by commands only. All labels except those shown on green or red background append characters to ***alpha*** immediately or via alpha catalogues. Labels shown on blue background here deviate from the standard printed on these locations. Generally, ⬆ toggles upper and lower case, and PSE appends a space. If ***alpha*** is going to exceed 31 characters, the leftmost character is discarded. Primary function of most keys is appending the letter printed bottom left of such a key. Then f is used for reaching the key tops there, and g leads to homonymic Greek letters where applicable. There are three exceptions: **η** is accessed via f E, and **τ** via g − (one key below **T**), and **ψ** via g 0 (below PSE). Omicron is not featured since looking exactly like **O** in either case. – Some logic symbols are accessible via the Boolean operations. 4 currency symbols are located next to each other as follows: **$** at the letter S, **€** at U for Euro, **£** at V, and **¥** at Y for Yen or Yuan. The catalogues h STAT, f ➡, f CPX, h TEST, and h ./, feature even more characters (see *below*).

Virtual keyboard in **temporary alpha mode**, active in catalogues or comparisons:



Labels printed on green background allow for catalogue browsing and exiting, as well as error recovery. ➡ and 2 … 9 are primary in comparisons but need f-shift in catalogues. See previous page, *addressing tables* and *catalogues* below for more.

MEMORY

Stack registers

	L	I **
	D *	
	C *	
	B *	
	A *	
	T	
Alpha (31 bytes)	Z	
	Y	
Display	X	

General purpose registers

R00	
R01	
R02	
R03	
…	
…	
R86	
R87 Σ *x*	
R88 Σ *x*²	
R89 Σ *y*	
R90 Σ *y*²	
R91 Σ (*x y*)	X = R100
R92 *n*	Y = R101
R93 Σ (ln *x*)	Z = R102
R94 Σ (ln² *x*)	T = R103
R95 Σ (ln *y*)	A = R104
R96 Σ (ln² *y*)	B = R105
R97 Σ(ln*x* ln*y*)	C = R106
R98 Σ (*x* ln *y*)	D = R107
R99 Σ (*y* ln *x*)	L = R108
	I = R109
J ***	J = R110
K ***	K = R111

User flags

00
01
02
03
…
…
99
B *Overflow*
C *Carry*
D *Danger*

Program steps

001
002
003
…
…
474
475
476

The stack may be 4 or 8 levels high. So either **T** or **D** will be the top stack level. **A – D** will be allocated as stack registers as required for the stack size specified.

Please see below for top level repetition and stack contents in complex calculations. Register **I** takes the imaginary part of the last argument when a complex function was executed (see LASTx).

After use of $\boxed{\Sigma+}$, general purpose registers **R87 – R99** will contain statistical sums as indicated. **J** and **K** may be taken for parameters of statistical distributions.

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

Flag **D** is set if “NaN” and “infinite” are legal results of commands.

STACK MECHANICS

The 34S offers a choice of either 4 or 8 stack levels. What happens with the contents of the individual levels depends on the function, its domain (integer/real or complex) and the stack size chosen.

Real and integer functions in a 4-level stack work as known for decades. Everything works alike in a larger stack on the 34S – just with more levels for intermediate results. Calculating formulas from inside out stays a wise strategy in either stack. With more levels, however, stack overflow will hardly ever happen, even for the most complex formulas you calculate in your life as a scientist or engineer.

Calculating with complex numbers uses 2 registers or levels for each such number as explained above.

<table>
<tr><th rowspan="3"></th><th rowspan="3">Level</th><th rowspan="3">Assumed contents at the beginning:</th><th colspan="6">Stack contents after executing …</th><th colspan="2">… complex functions of</th><th colspan="2" rowspan="2">… <b>integer/real</b> functions of 2 numbers like /</th></tr>
<tr><th colspan="6">… the complex stack register operations</th><th rowspan="2">1 number like ^{C}x^2</th><th rowspan="2">2 numbers like ^{C}/</th></tr>
<tr><th colspan="2">^{C}ENTER, ^{C}FILL</th><th colspan="3">^{C}x↔y, ^{C}R↓, ^{C}R↑</th><th>^{C}LASTx</th><th>Before</th><th>After</th></tr>
<tr><td rowspan="4">For 4 stack levels:</td><td>T</td><td>t = Im(y_c) = Im(t_c)</td><td colspan="2">Im(x_c)</td><td colspan="3">Im(x_c)</td><td rowspan="2">x_c</td><td rowspan="2">$y_c = t_c$</td><td rowspan="2">$y_c = t_c$</td><td>t</td><td>t</td></tr>
<tr><td>Z</td><td>z = Re(y_c) = Re(t_c)</td><td colspan="2">Re(x_c)</td><td colspan="3">Re(x_c)</td><td>z</td><td>t</td></tr>
<tr><td>Y</td><td>y = Im(x_c)</td><td colspan="2">Im(x_c)</td><td colspan="3">Im(y_c)</td><td rowspan="2">$lastx_c$</td><td>Im($^C(x_c)^2$)</td><td>Im(y_c / x_c)</td><td>y</td><td>z</td></tr>
<tr><td>X</td><td>x = Re(x_c)</td><td colspan="2">Re(x_c)</td><td colspan="3">Re(y_c)</td><td>Re($^C(x_c)^2$)</td><td>Re(y_c / x_c)</td><td>x</td><td>y / x</td></tr>
<tr><td rowspan="8">For 8 stack levels:</td><td>D</td><td>d = Im(t_c)</td><td rowspan="2">z_c</td><td rowspan="2">x_c</td><td rowspan="2">t_c</td><td rowspan="2">x_c</td><td rowspan="2">z_c</td><td rowspan="2">z_c</td><td rowspan="2">t_c</td><td>d = Im(t_c)</td><td>d</td><td>d</td></tr>
<tr><td>C</td><td>c = Re(t_c)</td><td>c = Re(t_c)</td><td>c</td><td>d</td></tr>
<tr><td>B</td><td>b = Im(z_c)</td><td rowspan="2">y_c</td><td rowspan="2">x_c</td><td rowspan="2">z_c</td><td rowspan="2">t_c</td><td rowspan="2">y_c</td><td rowspan="2">y_c</td><td rowspan="2">z_c</td><td>d = Im(t_c)</td><td>b</td><td>c</td></tr>
<tr><td>A</td><td>a = Re(z_c)</td><td>c = Re(t_c)</td><td>a</td><td>b</td></tr>
<tr><td>T</td><td>t = Im(y_c)</td><td rowspan="2">x_c</td><td rowspan="2">x_c</td><td rowspan="2">x_c</td><td rowspan="2">z_c</td><td rowspan="2">x_c</td><td rowspan="2">x_c</td><td rowspan="2">y_c</td><td rowspan="2">z_c</td><td>t</td><td>a</td></tr>
<tr><td>Z</td><td>z = Re(y_c)</td><td>z</td><td>t</td></tr>
<tr><td>Y</td><td>y = Im(x_c)</td><td rowspan="2">x_c</td><td rowspan="2">x_c</td><td rowspan="2">y_c</td><td rowspan="2">y_c</td><td rowspan="2">t_c</td><td rowspan="2">$lastx_c$</td><td rowspan="2">$^C(x_c)^2$</td><td rowspan="2">y_c / x_c</td><td>y</td><td>z</td></tr>
<tr><td>X</td><td>x = Re(x_c)</td><td>x</td><td>y / x</td></tr>
</table>

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

ADDRESSING AND COMPARING REAL NUMBERS

1	User input	x= ?, x≠ ?, x< ?, x≤ ?, x≥ ?, or x> ?				RCL, STO, αRCL, αSTO, VIEW, x≷, DSE, ISG, DSZ, ISZ, FIX, SCI, ENG, DISP, BASE, CB and many more bit commands, or CF and the other flag commands		
	Dot matrix display	OP _ (with temporary alpha mode switched on) e.g. ▪x > _				OP _ e.g. RCL _ [2]		
2	User input	0 or 1	X, Y, ... , K	ENTER↑ [3] closes alpha.	→ closes alpha.	(ENTER↑ [4]) X, Y, ... , K	*Number of register or flag or bit(s) or decimals* [5]	→
	Dot matrix display	OP *n* e.g. x ≤ 0 ?	OP *x* e.g. x ≥ y ?	OP r_	OP →_	OP *x* e.g. SCI Z	OP *nn* e.g. SF 15	OP →_ (indirect addressing)
3	User input	Compares ***x*** with the number **0**.	Compares ***x*** with the number on stack level **Y**.	0 0 … 9 9	Look right for more about indirect addressing.	Sets scientific display with the number of decimals specified in stack level **Z**.	X, Y, ... , K	0 0 … 9 9
	Dot matrix display			OP r *nn* e.g. x ≠ r23?			OP → *x* e.g. VIEW →L	OP → *nn* e.g. STO →45
				Compares ***x*** with the number stored in **R23**.			Shows the content of the register where **L** is pointing to.	Stores ***x*** into the location where **R45** is pointing to.

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

[3] You may skip this for register numbers >19 since pressing a numeric key >1 will close temporary alpha mode implicitly in comparisons.

[4] There is no general need switching to alpha mode via ENTER↑ – only RCL and STO require ENTER↑ being pressed for accessing **Z** or **T** here. – Some legal stack operations may be useless, e.g. **x<>X** .

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

ADDRESSING AND COMPARING COMPLEX NUMBERS

1	User input	CPX x= ? or x≠ ?				CPX RCL, STO, or x⇄		
	Dot matrix display	OP _ (with temporary alpha mode switched on) e.g. ▪x = _				OP _ e.g. ▪RCL _		
2	User input[6]	0 or 1	X, Z, A, C, L, or J	ENTER↑ [7] closes alpha.	→ closes alpha.	(ENTER↑ [8]) Z, A, C, L, or J	0 0 … 9 8 [9]	→
	Dot matrix display	OP *n* e.g. ▪x = 0 ?	OP *x* e.g. ▪x ≠ z ?	OP r_	OP →_	OP *x* e.g. ▪RCL L	OP *nn* e.g. ▪STO 18	OP →_ (indirect addressing)
3	User input	Compares ***x***+*i****y*** with the real number **0**.	Compares ***x***+*i****y*** with ***z***+*i****t*** .	0 0 … 9 8	Look right for more about indirect addressing.	Works like complex LASTx.	X, Y, ... , K	0 0 … 9 9
	Dot matrix display			OP r *nn* e.g. ▪x ≠ r26?			OP → *x* e.g. ▪x<> →Z	OP → *nn* e.g. ▪STO →45
				Compares ***x***+*i****y*** with a complex number having its real part in **R26** and imaginary part in **R27**.			Swaps ***x*** with the contents of the register where **Z** is pointing to, and ***y*** with the contents of the next one.	Stores ***x***+*i****y*** into 2 consecutive registers, starting with the one where **R45** is pointing to.

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

[7] You may skip this for register numbers >19 since pressing a numeric key >1 will close temporary alpha mode implicitly in comparisons.

[8] Only RCL and STO require ENTER↑ for addressing **Z**, else you may skip this keystroke.

[9] For numbers <10, you may key in e.g. 8 ENTER↑ instead of 0 8 . Some registers may be allocated to special applications. Take also care of pairs, since a complex operation will always affect two registers: the one specified and the one following this. We recommend storing complex numbers with their real parts at even register numbers.

ADDRESSING LABELS

1	User input	GTO, XEQ, LBL, π, Σ, ∫ or SOLVE [10]				
	Dot matrix display	OP _ (e.g. GTO _)				
2	User input	B, C, or D	ENTER↑ turns alpha mode on.	→ [11]		*2-digit numeric label* 00 … 99
	Dot matrix display	OP '*name*' e.g. Σ 'B'	OP '_	OP →_ (indirect addressing)		OP *nn* e.g. LBL 07
3	User input	Sum up the function labeled **B.**	*Label* [12]	X, Y, ... , K [13]	00 … 99 [14]	
	Dot matrix display		OP '*name*' e.g. SLV'F1µ'	OP → *x* e.g. ∫ →T	OP → *nn* e.g. XEQ →44	
			Solve the function **F1µ** (with F1µ keyed in).	Integrate the function which's label is on stack level **T**.	Execute the routine which's label is in **R44**.	

[10] SOLVE will be displayed and listed as SLV. The routines labelled B, C, and D may be called for execution directly via B , C , or D , respectively, without pressing XEQ before.

[11] Works with all these operations except LBL.

[12] Such a label may consist of up to 3 alphanumeric characters. The 3rd character terminates entry and closes alpha mode. For labels with less than 3 characters, a closing ENTER↑ is mandatory.

[13] There is no need for switching to alpha mode before. Some stack levels may be reserved in certain functions, please check the index of operations.

[14] Some registers may be allocated to special applications, so take care.

ADDRESSING ITEMS IN CATALOGUES

1	User input	CONS, CONV, P.FCN, STAT, TEST, X.FCN	CPX or STAT in alpha mode	TEST, ./., or → in alpha mode
	Dot matrix display	**Shows 1st item in selected catalogue.** Temporary alpha mode is on. (e.g. BC? in P.FCN)	(e.g. Á in CPX)	(e.g. , in PUNCT)
2	User input	ENTER↑, ▼, ▲, EXIT, or 1st character (e.g. F)	ENTER↑, ▼, ▲, EXIT, or letter (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	ENTER↑, ▼, ▲, EXIT, or 2nd character (e.g. S)	↓	↓
	Dot matrix display	**Shows 1st item starting with this sequence *)** (e.g. FS?)	↓	↓
4	User input	ENTER↑, ▼, ▲, or EXIT (e.g. ▼)		
	Dot matrix display	**Shows next item in this catalogue** (e.g. FS?C)	(e.g. Ò)	(e.g. ")

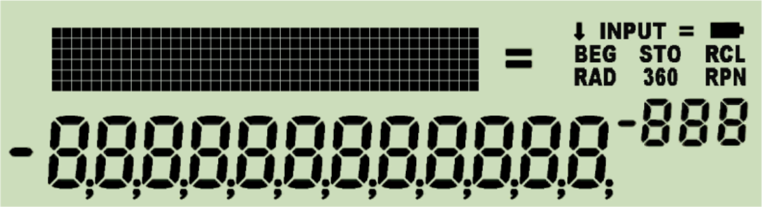
Continue browsing this way until the desired item is displayed

		(e.g. FS?F).	(e.g. Ö).	(e.g. :).
n	User input	ENTER↑	ENTER↑	
	Dot matrix display	**Result** Calculator returns to the mode set before and executes or inserts the selected operation.	**Character appended to *alpha*** (e.g. Östl. Seite:) Calculator leaves the catalogue returning to alpha mode.	

*) If the character or sequence specified is not found in this catalogue then the first item following alphabetically will be shown.

DISPLAY

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, adjusted to the right,
2. passing additional information to the user, adjusted to the left.

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

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

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

There are a number of indicators signaling the mode the calculator is running in:

Indicator or *annunciator*	***INPUT***	**b**	**d**	**h**	**o**	***STO***
Mode name if different	α	**2**			**8**	**PRG**
Set by …	αON	BASE2	BASE10	BASE16	BASE8	PRGON
Cleared by …	αOFF	any other BASE setting, FLOAT, FRACT				PRGOFF

Indicator or *annunciator*	***360***	***RAD***	**G**	**H.MS**	**/c**
Set by …	DEG	RAD	GRAD	H.MS TIME →H.MS	BASE1 FRACT 2nd □ in input
Cleared by …	GRAD RAD	DEG GRAD	DEG RAD	BASE COS, SIN, TAN FLOAT, FRACT →HR	BASE ≠1 FLOAT

A running program is signaled by a flashing ***RCL*** annunciator. ***RPN*** may be lit permanently. Most other settings are indicated in the dot matrix section, like the different date modes being signaled there by **D.MY** or **M.DY**. Defaults Y.MD and FLOAT are not indicated. Time modes (12h / 24h) are seen in the time string directly. Please check the examples below.

Some mode and display settings may be stored and recalled collectively by the commands STOM and RCLM. RCLM recalls a 17-bit word containing mode data packed as follows, starting with the least significant bit:

Bits	Meaning	Values and corresponding settings		
0, 1	Display format for real numbers	0 = ALL 2 = SCI	1 = FIX 3 = ENG	
2 … 5	Number of decimals	0 … 12		
6, 7	Angular mode	0 = DEG	1 = RAD	2 = GRAD
8, 9	Date display format	0 = Y.MD	1 = D.MY	2 = M.DY
10 … 12	Curve fit model	0 = LinF 3 = LogF	1 = ExpF 4 = BestF	2 = PowerF
13	Time display format	0 = 24h	1 = 12h	
14, 15	Integer sign mode	0 = 2COMPL 2 = UNSIGN	1 = 1COMPL 3 = SIGNMT	
16	Stack depth	0 = 4 levels	1 = 8 levels	

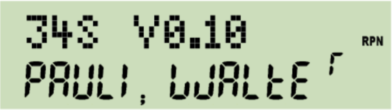
E.g. the start-up default with 4 stack levels,
FIX 4, DEG, Y.MD, LinF, 24h, 2COMPL is $00000000000010001_2 = 17_{10}$

Settings for 8 stack levels, SCI 2, RAD,
D.MY, BestF, 12h, UNSIGN correspond to $11011000101001010_2 = 110922_{10}$

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

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

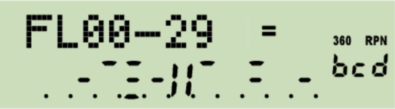
1. VERS generates a display like this:



```
34S V0.10          RPN
PAULI, WALtE r
```

This tells you have a 34S with firmware version 0.10 – the display on your 34S may deviate from this example. Pressing any key will delete this message and return to previous state.

2. STATUS displays the status of 30 flags very concisely, 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:



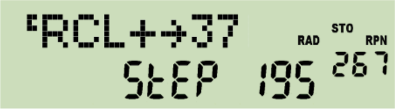
Within the numeric section, each row of horizontal bars in the mantissa shows the status of 10 flags. When a flag is set, the respective bar turns black. So here the top row of bars indicates flags 0 and 1 being clear, flags 2 and 3 set, and flag 4 clear. Then, the divider II separates the first group of five flags from the next. Top row bars on the right side of the II indicate flags 5 and 7 are set – 6, 8, and 9 are clear. Next row shows flags 11, 13, 14, and 17, 19 are set, and the lowest row indicates 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 80 - D. Scrolling up by ▲ reverts this. Alternatively, pressing a digit, e.g. 5, will show 30 flags starting with 10 times this digit, e.g. flags 50 - 79. The numeric exponent always indicates the status of the 3 hotkeys top left on the keyboard.

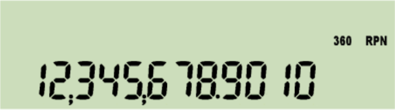
The status will be displayed until any key is pressed but ▼, ▲, or a digit < 9.

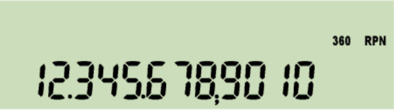
3. During command input, the dot matrix shows the command chosen until the input is completed, i.e. until the required parameters are entered. The prefixes f, g, and h are shown until they are resolved. In addressing, progress is recorded as explained in the addressing tables above in detail.

4. In programming mode, the numeric display indicates the program step (001 – 476) in the mantissa and the number of free steps in the exponent, while the dot matrix shows the command contained in the respective step.

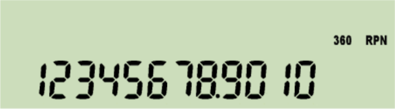


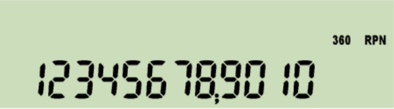
5. For floating point numbers, the mantissa will be displayed adjusted to the right, the exponent to the left. Within the mantissa, either points or commas may be selected as radix marks[15], and additional marks may be chosen to separate thousands. Assume the display set to FIX 4, then 12.345678901 millions may look like:



or


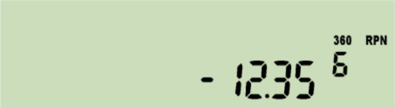
with thousands separators on, and without them like:



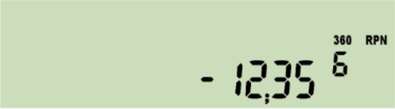
or


[15] Beginning with point 7 below, decimal input is written using a point as radix mark throughout this manual, though 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.

With ENG 2 and after changing the sign, the same number looks like this:



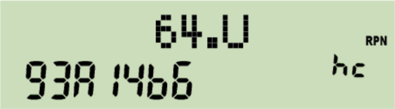
or



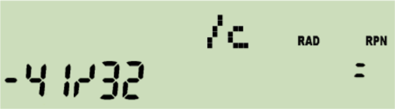
6. In integer modes, numbers are displayed adjusted to the left. Word size and complement setting are indicated in the dot matrix using a format WW.C, with C being 1 or 2 for 1's or 2's complement, U for unsigned, or S for sign-and-mantissa mode. Sign and 1st digit of the exponent show the base, a "c" in the 2nd digit signals a carry bit set, an "o" in the 3rd an overflow. Integer bases are indicated as follows:

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

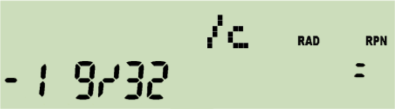
The example shows the 34S in unsigned hexadecimal mode with word size 64 and carry set:



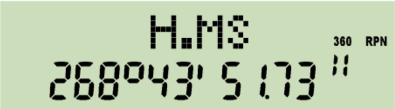
7. In fraction mode, the fraction will be shown in the mantissa section of the numeric display, adjusted to the left. "=", "Lt", or "Gt" is indicated in the exponent if the fraction is exactly equal, slightly less, or greater than the floating point number converted, respectively.
E.g. -1.28125 will be displayed as follows, depending on the setting for proper fractions:



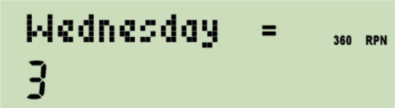
or



8. In H.MS mode, input format is `hhhh.mmssdd` with the number of hours or degrees limited to 9000. Output is adjusted to the right. It may look like this:

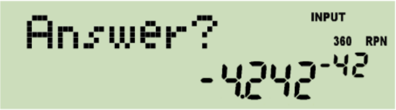


9. Output of the function DAY will look like the following for an input of 1.132010 in M.DY mode (equivalent to inputs of 13,012010 in D.MY or 2010.0113 in Y.MD).



The display may look alike for a result of DAYS+.

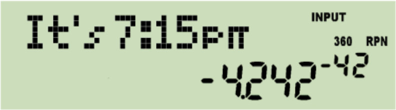
10. In alpha mode, the contents of the alpha register are displayed in the dot matrix while the numeric section keeps the result of the last numeric operation.



Answer?

-4.242^{-42}

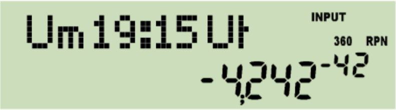
Different information may be appended to ***alpha***. See the commands starting with “α” in the *index of operations*. E.g. αTIME allows creating texts like



It's 7:15pm

-4.242^{-42}

or



Um 19:15 Uhr

-4.242^{-42}

depending on the time mode setting (12h / 24h).

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

INDEX OF OPERATIONS

This lists all functions available on the 34S with their names and the necessary keystrokes. Names printed in **bold** face belong to commands directly accessible on the keyboard, the others are accessible via catalogues. These names will show up in program listings as well. Sorting is case insensitive and works as follows: 0 … 9, A … Z, α … ω, (,), +, -, *, /, ±, “,”, “.”, !, ?, ↔, ←, ↑, ↓, →, <, ≤, =, ≠, ≥, >, #, °, %, √, ∫, ∞. Super- and subscripts are handled like normal characters.

Generally, functions and keystroke programming will work like on the HP-42S, special bit and integer functions as on the HP-16C, unless stated otherwise under remarks. We recommend you get the manuals of these vintage calculators, e.g. on the DVD distributed by www.hpmuseum.org.

Functions available on the 34S for the first time on an RPN calculator are highlighted under remarks, as are functions carrying a familiar name but deviating in their functionality here.

Parameters will be taken from the lowest stack levels unless being 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 may be given in registers **J** and **K** if mentioned in the last column. Commands with their names printed in *italics* in this table will also work with complex parameters.

Each function is listed stating the calculator mode(s) it will work in, abbreviated by their *indicators*. In this column an “&” represents a logical AND, a comma a logical OR, and a backslash stands for “all but”. So e.g. 2^X works in all modes but alpha. “FLOAT^H” stands for “FLOAT, H.MS”. All operations will also work in mode PRG unless stated otherwise explicitly.

Name	Keys to press	in modes	Remarks
c…	CPX …	FLOAT	Indicates a complex operation (see *above*). CPX may be combined with *many functions*.
0 … 9	0 … 9	\α	Standard numeric input. For integer bases <10, input of illegal digits throws an *error message*.
10^x	g 10^x	FLOAT	
12h	h X.FCN 12h	FLOAT	Sets 12h time display. 21:10 becomes 9:10 p.m.
1COMPL	h X.FCN 1COMPL	Integer	Sets 1's complement mode like in HP-16C.
1/x	f 1/x	FLOAT	
	B	FLOAT	Shortcut as long as label B is not defined yet.
24h	h X.FCN 24h	FLOAT	Sets 24h time display. 1:23 p.m. becomes 13:23.
2COMPL	h X.FCN 2COMPL	Integer	Sets 2's complement mode like in HP-16C.
2^x	g 2^x	\α	
A … F	A … F (red print)	-1, -2, -3, -4, -5, h	Numeric input for digits >10. For bases <16, key defaults are switched as applicable.
A … Z	A … Z (red print)	α	Alphabetic input. See page 6 for more information. Find alpha catalogues below.
ABS	f |x|	\α	^{C}ABS returns the magnitude $r=\sqrt{x^2+y^2}$ in **X** and clears **Y**.
ACOS	g COS^{-1}	$FLOAT^{H}$	
ACOSH	g HYP^{-1} COS	FLOAT	
ALL	h X.FCN ALL	FLOAT	Selects the format displaying "all" digits.
AND	h AND	Integer	Works bitwise as in HP-16C.
		FLOAT	Works like AND in HP-28S, i.e. ***x*** and ***y*** are interpreted before executing this operation. 0 is "false", any other real number is "true".
ANGLE	h X.FCN ANGLE	FLOAT	Calculates the angle between positive x-axis and the straight line connecting the origin with the point (***x, y***). Returns it in **X** and clears **Y**.
ASIN	g SIN^{-1}	$FLOAT^{H}$	
ASINH	g HYP^{-1} SIN	FLOAT	
ASR	h X.FCN ASR ***n***	Integer	Works like ***n*** (up to 63) consecutive ASRs in HP-16C. ASR 0 executes as NOP.

Name	Keys to press	in modes	Remarks
ATAN	g TAN^{-1}	FLOAT^{H}	
ATANH	g HYP^{-1} TAN	FLOAT	
BASE	h X.FCN BASE ***n***	\α	Sets the base for integer calculations, with 2 ≤ ***n*** ≤ 16. Popular bases are directly accessible on the keyboard. Furthermore, BASE0 calls FLOAT, and BASE1 calls FRACT. Actual base setting is indicated in the exponent as explained above.
BASE10	f 10		
BASE16	g 16		
BASE2	f 2		
BASE8	g 8		
BC?	h TEST BC? ***n***	PRG & integer	Tests the specified bit in ***x*** **.** Executes the next program line if this bit is clear, else skips it.
BestF	h STAT BestF	FLOAT	Selects the best curve fit model, maximizing the correlation like BEST in HP-42S.
BS?	h TEST BS? ***n***	PRG & integer	Tests the specified bit in ***x*** . Executes the next program line if this bit is set, else skips this line.
B(k)	h STAT B(k)	FLOAT	= BINOMDIST(***x; j; k;*** **1**) in MS Excel, with the sample size ***j*** and the gross error probability ***k***. B^{-1} returns the number of successes ***g*** for a given probability ***p*** . A similar function like B is found in the HP 30b now.
B $^{-1}$(p)	h STAT B^{-1}(p)		
CB	h X.FCN CB ***n*** h P.FCN CB ***n***	Integer	Clears the specified bit in ***x*** .
CEIL	h X.FCN CEIL	FLOAT	Returns the smallest integer ≥ ***x*** .
CF	h CF ***n***	\α	Clears the flag specified.
CLFLAG	h P.FCN CLFLAG	\α	Clears all user flags.
CLREG	h X.FCN CLREG	\α	Clears all general purpose registers.
CLSTK	0 h FILL	\α	Clears all stack registers.
CLx	h CLX	\α	^{C}CLx clears both **X** and **Y**.
CLα	f CLα	All	Clears the alpha register like CLA in HP-42S.
CLΣ	g CLΣ	FLOAT	Clears all statistical sums.

Name	Keys to press	in modes	Remarks
COMB	f Cy,x	FLOAT	Returns the number of possible sets 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 not counted separately. Formula: $C_{y,x} = \binom{y}{x} = \frac{y!}{x! \cdot (y-x)!}$
CONJ	h CONJ	FLOAT	Changes the sign of ***y*** .
CORR	g r	FLOAT	Returns the correlation coefficient for the current statistical data and curve fitting model
COS	f COS	FLOAT^H	
COSH	f HYP COS	FLOAT	
DATE	h X.FCN DATE	FLOAT	Recalls the date from the real time clock and displays it in the numeric section in the format selected. See D.MY, M.DY, and Y.MD. The function DATE in HP-12C corresponds to DAYS+ here (see below).
DAY	h X.FCN DAY	FLOAT	Takes ***x*** as a date in the format selected and returns the day of week in the dot matrix and a corresponding integer in the numeric display (Sunday = 7).
DAYS+	h X.FCN DAYS+	FLOAT	Works like DATE in HP-12C, adding ***x*** days on a date in **Y** in the format selected and displaying the resulting date including the day of week in the same format as DAY does.
DBLR	h X.FCN DBLR	Integer	Double precision commands like in HP-16C, but here for up to 128 bits.
DBL ×	h X.FCN DBL×		
DBL /	h X.FCN DBL/		
DEG	g DEG	FLOAT	Sets angular mode to degrees.
DECOMP	h X.FCN DECOMP	/c	Decomposes the fraction in **X**, resulting in a stack [***numerator(x), denominator(x), y, z***] or [***num(x), den(x), y, z, t, a, …***] , respectively. Reversible by division.
DENANY	h X.FCN DENANY	FLOAT	Sets default fraction format like in HP-35S, allowing maximum precision. Any denominator up to the selected maximum is possible.
DENFAC	h X.FCN DENFAC	FLOAT	Sets "factors of the maximum denominator" format like in HP-35S. With e.g. DENMAX set to 60, possible denominators are 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, and 60.

Name	Keys to press	in modes	Remarks
DENFIX	h X.FCN DENFIX	FLOAT	Sets fixed denominator format like in HP-35S. In this format, the denominator will be the maximum set via DENMAX always.
DENMAX	h X.FCN DENMAX	FLOAT	Works like /c in HP-35S, but maximum value settable is 9999. The max. denominator will be set to 9999 if ***x*** = 0 or ***x*** > 9999 at execution time. For ***x*** = 1 the current setting is recalled.
DISP	h X.FCN DISP ***n***	FLOAT	Changes the number of decimals while keeping the display format.
DROP	h P.FCN DROP	\α	Drops ***x*** , changing stack contents to [***y, z, t, t***] or [***y, z, t, a, b, b***] or [***y, z, t, a, b, c, d, d***] , respectively. See *above* for complex DROP.
DSE	f DSE ***r***	PRG	Given `cccccc.fffii` in ***r***, this function decrements ***r*** by `ii`, and skips the next program line if `ccccccc` is then ≤ `fff` for DSE, or = 0 for DSZ.
DSZ	h P.FCN DSZ ***r***		
D.MY	h X.FCN D.MY	FLOAT	Sets the format for date calculations.
D→J	h X.FCN D→J	FLOAT	Assumes **X** containing a date and converts it to a Julian day number.
D→R	h X.FCN D→R	FLOAT	Assumes **X** containing degrees and converts them to radians. Angular mode is kept.
E	E (the key)	FLOAT	Like EEX in vintage calculators.
E3OFF	h X.FCN E3OFF	FLOAT	Toggle the thousands separator, being either a comma or a point depending on the radix setting.
E3ON	h X.FCN E3ON		
ENG	f ENG ***n***	FLOAT	Selects engineering display format.
ENTER↑	ENTER↑	\α	See *above* for ^{C}ENTER.
ERF	h STAT ERF	FLOAT	Calculates the error function erf(***x***).
e ^{x}	g e^{x}	FLOAT	
ExpF	h STAT ExpF	FLOAT	Selects the exponential curve fit model.
Ex(t)	h STAT Ex(t)	FLOAT	= EXPONDIST(***x; j; 1***) in MS Excel, with **J** containing the rate λ. Ex^{-1} returns the survival time t_s for a given probability ***p*** .
Ex $^{-1}$(p)	h STAT Ex^{-1}(p)		
e^{x} -1	h X.FCN e^{x}-1	FLOAT	
FB	h X.FCN FB ***n***	Integer	Inverts ("flips") the specified bit in ***x*** .
	h P.FCN FB ***n***		

Name	Keys to press	in modes	Remarks
FC?	h TEST FC? ***n***	PRG	Tests the flag specified and executes the next program line if this flag is clear, else skips this line. Clears, flips, or sets this flag after testing, if applicable.
FC?C	h TEST FC?C ***n***		
FC?F	h TEST FC?F ***n***		
FC?S	h TEST FC?S ***n***		
FF	h X.FCN FF ***n***	\α	Flips the flag specified.
	h P.FCN FF ***n***		
FIB	h X.FCN FIB	\α	Calculates the Fibonacci number F_x .
FILL	h FILL	\α	Copies ***x*** to all other stack levels. See [above](#) for ^{C}FILL.
FIX	f FIX ***n***	FLOAT	Selects fixed point display format.
FLOAT	f .d	\α	Works like DECM in HP-42S. Additionally, converts H.MS data in **X** to decimal format, if applicable.
	g H	H.MS	
FLOOR	h X.FCN FLOOR	FLOAT	Returns the largest integer ≤ ***x*** .
FP	g FP	FLOAT	Returns the fractional part of ***x*** .
FRACT	g b/c	FLOAT	Sets fraction mode like in HP-35S. See PROFRC, IMPFRC, and DEN… for more. Absolute decimal equivalents must be >10E-5 and <10E5.
FS?	h FS? ***n***	PRG	Tests the flag specified and executes the next program line if this flag is set, else skips this line. Clears, flips, or sets this flag after testing, if applicable.
FS?C	h TEST FS?C ***n***		
FS?F	h TEST FS?F ***n***		
FS?S	h TEST FS?S ***n***		
F(x)	h STAT F(x)	FLOAT	F works like Q(F) , F^{-1} like F$_{P}$ in HP-21S. The degrees of freedom are given in **J** and **K**. Similar functions are found in the HP 30b now.
F $^{-1}$(p)	h STAT F^{-1}(p)		
GCD	h X.FCN GCD	\α	Returns the Greatest Common Divisor of ***x*** and ***y*** .
Ge(k)	h STAT Ge(k)	FLOAT	Geometric distribution, returns $1-(1-p_0)^k$. The gross error probability ***p₀*** must be given in **J**. Ge^{-1} returns the number of failures ***f*** before the 1st success for a given probability ***p*** .
Ge $^{-1}$(p)	h STAT Ge^{-1}(p)		
GRAD	g GRAD	FLOAT	Sets angular mode to gon or grads.

Name	Keys to press	in modes	Remarks
GTO	h GTO ***label***	PRG	Inserts an unconditional branch to the label specified.
		\PRG, \α	Positions the program pointer to this label.
	h GTO . ***nnn***	\α	Positions the program pointer to line ***nnn*** or to PRGM TOP, respectively (not programmable). Compare RTN.
	h GTO . .		
H.MS	g H.MS	FLOAT	Sets H.MS mode for time calculations. See the paragraph about display above.
H.MS+	+	H.MS	Assumes **X** and **Y** containing times in the format `hhhh.mmssdd` , and adds or subtracts them, respectively.
H.MS–	−		
IMPFRC	h X.FCN IMPFRC	FLOAT	Allows improper fractions in fraction mode. Default are proper fractions, see PROFRC.
IP	f IP	FLOAT	Returns the integer part of ***x*** .
ISG	g ISG ***r***	PRG	Given `cccccc.fffii` in ***r***, this function increments ***r*** by `ii`, and skips the next program line if `ccccccc` is then > `fff` for ISG, or = 0 for ISZ.
ISZ	h P.FCN ISZ ***r***		
I β	h X.FCN Iβ	FLOAT	Calculates the regularized incpl. beta function $\frac{B(x,y,z)}{\beta(y,z)}$ with $B(x,y,z)=\int_0^x t^{y-1}(1-t)^{z-1}dt$.
I Γ	h X.FCN IΓ	FLOAT	Calculates the regularized incomplete gamma function $\frac{\gamma(x,y)}{\Gamma(x)}$ with $\gamma(x,y)=\int_0^y t^{x-1}e^{-t}dt$.
J→D	h X.FCN J→D	FLOAT	Assumes **X** containing a Julian day number and converts it to a date in the format selected.
LASTx	g LASTx	\α	See above for ^{C}LASTx .
LBL	f LBL ***label***	PRG	Identifies programs and routines for execution and branching. See opportunities for *label* in the table above.
LCM	h X.FCN LCM	\α	Returns the Least Common Multiple of ***x*** and ***y***.
LEAP?	h TEST LEAP?	PRG & FLOAT	Takes ***x*** as a date in the format selected, extracts the year, and tests for a leap year. Executes the next program line if true, else skips this line.
LinF	h STAT LinF	FLOAT	Selects the linear curve fit model.

Name	Keys to press	in modes	Remarks
LJ	h X.FCN LJ	Integer	
LN	f LN	FLOAT	
LN1+X	h X.FCN LN1+X	FLOAT	
LN β	h X.FCN LNβ	FLOAT	Calculates the natural logarithm of β(***x***, ***y***) or Γ(***x***), respectively. See these functions.
LN Γ	h X.FCN LNΓ		
LOG$_{10}$	f LG	FLOAT	
LOG$_{2}$	f LB	\α	Calculates the logarithm of ***x*** for base 2.
LogF	h STAT LogF	FLOAT	Selects the logarithmic curve fit model.
LOGy	f LOGy	FLOAT	Calculates the logarithm of ***x*** for base ***y*** .
	CPX f LOGy	FLOAT	Calculates the logarithm of the complex number ***x*** + *i* ***y*** for the complex base ***z*** + *i* ***t*** .
L.R.	h L.R.	FLOAT	Calculates 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***	Integer	Work like MASKL and MASKR on HP-16C, but with the mask length following the command instead of taken from **X**.
MASKR	h X.FCN MASKR ***n***		
MAX	h X.FCN MAX	\α	Returns the maximum of ***x*** and ***y*** .
MIN	h X.FCN MIN	\α	Returns the minimum of ***x*** and ***y*** .
MIRROR	h X.FCN MIRROR	Integer	Reflects the bit pattern in ***x*** (e.g. 000101 becomes 101000 for word size 6).
MOD	h MOD	\α	MOD of HP-42S equals RMD of HP-16C.
M.DY	h X.FCN M.DY	FLOAT	Sets the format for date calculations.
NAND	h X.FCN NAND	\α	Works in analogy to AND.
NaN?	h TEST NaN?	PRG	Tests ***x*** for "not a number" and executes the next program line if true, else skips this line.
nBITS	h X.FCN nBITS	Integer	Counts bits set in ***x*** like #B on HP-16C.
NOP	h P.FCN NOP	PRG	
NOR	h X.FCN NOR	\α	Works in analogy to AND.

Name	Keys to press	in modes	Remarks
NOT	h NOT	\α	Works in analogy to AND.
nΣ	h STAT nΣ	FLOAT	Recalls the number of accumulated data points. Necessary for basic statistics.
N(x)	h STAT N(x)	FLOAT	= NORMDIST(***x; j; k;* 1**) in MS Excel, with the mean ***j*** and the standard deviation ***k***. N^{-1} equals NORMINV(***x; j; k***) .
N^{-1}(p)	h STAT N^{-1}(p)		
OFF	h P.FCN OFF	PRG	
ON	h P.FCN ON		
OR	h OR	\α	Works in analogy to AND.
PAUSE	h PSE	PRG	Pauses program execution for about 1 s.
PERM	g Py,x	FLOAT	Returns the number of possible arrangements 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 are counted separately. Formula: $P_{y,x} = x! \cdot C_{y,x}$, see COMB and FACT.
PowerF	h STAT PowerF	FLOAT	Selects the power curve fit model.
PROFRC	h X.FCN PROFRC	FLOAT	Allows only proper fractions in fraction mode. Compare IMPFRC.
PROMPT	h P.FCN PROMPT	PRG	Displays ***alpha*** and stops program execution. Enter the requested value and press R/S to continue.
P(k)	h STAT P(k)	FLOAT	= POISSON(***x; j*k;* 1**) in MS Excel, with the gross error probability ***j*** and the sample size ***k***. Alternatively, the Poisson parameter λ may be in **J** if **K** contains 1. P^{-1} returns the number of successes ***g*** for a given probability ***p***.
P^{-1}(p)	h STAT P^{-1}(p)		
Q(x)	f Q	FLOAT	Works like Q in HP-32E and Q(z) in HP-21S.
Q^{-1}(p)	g Q^{-1}	FLOAT	Works like Q^{-1} in HP-32E and z_P in HP-21S.
RAD	g RAD	FLOAT	Sets angular mode to radians.
RAND#	h STAT RAND#	FLOAT	Returns a random number between 0 and 1 like RAN in HP-42S.
	h X.FCN RAND#	Integer	Returns a random bit pattern within the word size given.
RCL	RCL **s**	\α	

Name	Keys to press	in modes	Remarks
RCLM	h P.FCN RCLM	PRG	Recalls selected mode settings into **X**. See the paragraph about *indicators* above.
RCL+	RCL + ***s***	\α	Recalls the content of address ***s***, executes the specified operation on it and places the result in **X**. E.g. RCL –12 recalls the contents of **R12**, subtracts ***x*** from it and voilá. RCL↑ (↓) recalls the maximum (minimum) of the value in ***s*** and **X**. See the *addressing table above* also for complex RCL.
RCL–	RCL − ***s***		
RCL×	RCL × ***s***		
RCL/	RCL / ***s***		
RCL↑	RCL ▲ ***s***		
RCL↓	RCL ▼ ***s***		
RDX,	h ./,	FLOAT	Toggle the radix mark.
RDX.			
RJ	h X.FCN RJ	Integer	Works in analogy to LJ.
RL	h X.FCN RL ***n***	Integer	Works like ***n*** consecutive RLs / RLCs on HP-16C. For RL, 1 ≤ ***n*** ≤ 63. For RLC, 1 ≤ ***n*** ≤ 64. RL 0 and RLC 0 execute as NOP.
RLC	h X.FCN RLC ***n***		
ROUND	g RND	FLOAT	Rounds ***x*** using the current display format, like RND in HP-42S.
		/c	Rounds ***x*** using the current denominator, like RND in HP-35S.
ROUNDI	h X.FCN ROUNDI	FLOAT	Rounds ***x*** to next integer. ½ rounds to 1.
RR	h X.FCN RR ***n***	Integer	Works like ***n*** consecutive RRs / RRCs on HP-16C. See RL / RLC for more.
RRC	h X.FCN RRC ***n***		
RTN	g RTN	PRG	Last command in a routine. Will return control to the calling routine in program execution.
		\PRG	In program execution: Returns control to the calling routine, i.e. moves the program pointer to the line following the most recent XEQ instruction encountered. If there is no matching XEQ, program execution halts. Entered from the keyboard: Moves the program pointer to the first line of the routine observed. Compare GTO.
RTN+1	n/a	PRG	Internal support routine.
R-CLR	h P.FCN R-CLR	FLOAT	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**.

Name	Keys to press	in modes	Remarks
R-COPY	h P.FCN R-COPY	FLOAT	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, contents of registers 07 - 09 will be moved to registers 45 - 47, overwriting their old contents.
R-SORT	h P.FCN R-SORT	FLOAT	Interprets ***x*** in the form `ss.nn`. Sorts the contents of ***nn*** registers starting with number ***ss***. Assume ***x*** = 49.026 and **R49** and **R50** contain 1.2 and -3.4, respectively; then R-SORT will end with the contents of these 2 registers swapped.
R-SWAP	h P.FCN R-SWAP	FLOAT	Works like R-COPY but swaps the register contents of source and destination.
R↑	h R↑	\α	Rotates the stack contents one level up or down, respectively. See above for complex rotations.
R↓	R↓		
R→D	h X.FCN R→D	FLOAT	Assumes **X** containing radians and converts them to degrees. Angular mode is kept.
s	g s	FLOAT	Uses the current statistical data for calculating s_y and s_x, and pushes them on the stack.
SB	h X.FCN SB ***n***	Integer	Sets the specified bit in ***x***.
	h P.FCN SB ***n***		
SCI	f SCI ***n***	FLOAT	Selects scientific display format.
SEED	h STAT SEED	FLOAT	Stores a seed for random number generation.
SERR	h STAT SERR	FLOAT	Calculates $s/\sqrt{n}$ and pushes the results on the stack like the function s does. A similar calculation is found in the HP 30b now.
SETDAT	h X.FCN SETDAT	FLOAT^H	Sets the date or time, respectively, for the real time clock.
SETTIM	h X.FCN SETTIM		
SF	h SF ***n***	\α	Sets the flag specified.
SIGN	h X.FCN SIGN	\α	Returns 1 for ***x*** > 0, –1 for ***x*** < 0, and 0 for ***x*** = 0 or non-numbers.
	CPX h ...	FLOAT	Returns the unit vector of ***x*** + *i* ***y*** in **X** and **Y**.
SIGNMT	h X.FCN SIGNMT	Integer	Sets sign-and-mantissa mode for integers.
SIN	f SIN	FLOAT^H	

Name	Keys to press	in modes	Remarks
SINC	h X.FCN SINC	FLOAT	Calculates $\frac{\sin(x)}{x}$.
SINH	f HYP SIN	FLOAT	
SL	h X.FCN SL ***n***	Integer	Works like ***n*** (up to 63) consecutive SLs on HP-16C. SL 0 executes as NOP.
SLV	h SOLVE ***label***	FLOAT	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.
SR	h X.FCN SR ***n***	Integer	Works like ***n*** consecutive SRs on HP-16C. SR 0 executes as NOP.
SSIZE4	h X.FCN SSIZE4	All	Sets 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. The same will happen if the stack size is changed via STOM.
SSIZE8	h X.FCN SSIZE8		
SSIZE?	h X.FCN SSIZE?	All	Returns the number of stack levels accessible.
STO	STO ***d***	\α	
STOM	h P.FCN STOM	PRG	Sets selected modes as encoded in ***x***. See the paragraph about *indicators* above.
STOP	R/S	PRG	Stops program execution.
STO+	STO + ***d***	\α	Executes the specified operation on the content of address ***d*** and stores the result into said address. E.g. STO –12 subtracts ***x*** from the contents of **R12**, and stores the result there again. STO↑ (↓) takes the maximum (minimum) of the values in ***d*** and **X** and stores it. See the *addressing table above* for complex STO.
STO–	STO – ***d***		
STO×	STO × ***d***		
STO/	STO / ***d***		
STO↑	STO ▲ ***d***		
STO↓	STO ▼ ***d***		
SUM	h STAT SUM	FLOAT	Recalls the linear sums Σy and Σx. Useful for basic vector algebra.
TAN	f TAN	FLOAT^H	
TANH	f HYP TAN	FLOAT	
TIME	h X.FCN TIME	FLOAT^H	Recalls the time from the real time clock.

Name	Keys to press	in modes	Remarks
t(x)	h STAT t(x)	FLOAT	t works like Q(t) , t^{-1} like tp in HP-21S. The degree of freedom is stored in **J**. Similar functions are found in the HP 30b now.
t^{-1}(p)	h STAT t^{-1}(p)		
UNSIGN	h X.FCN UNSIGN	Integer	Sets unsigned mode for integers.
VIEW	h VIEW ***s***	All	Views the contents of address ***s*** .
W	h X.FCN W	FLOAT	W returns Lambert's W for given ***x*** ≥ -1/e , while W^{-1} returns ***x*** for given W (≥ -1).
W^{-1}	h X.FCN W^{-1}		
Wb(t)	h STAT Wb(t)	FLOAT	= WEIBULL(***x; j; k;* 1**) in Excel, with the *shape parameter* ***j*** and the *characteristic lifetime* ***k***. Wb^{-1} returns the survival time t_s for given probability ***p*** .
Wb^{-1}(p)	h STAT Wb^{-1}(p)		
WSIZE	h X.FCN WSIZE ***n***	Integer	Works like WSIZE on HP-16C, but with the parameter following the command instead of taken from **X**. WSIZE 0 sets the word size to maximum, i.e. 64 bits.
WSIZE?	h X.FCN WSIZE?	Integer	Recalls the word size set.
x^2	g x^2	\α	
XEQ	XEQ ***label***	PRG	Calls the respective subroutine.
		\PRG, \α	Executes the respective program.
	B , C , or D (f may be needed for accessing these hotkeys in integer bases >10.)	PRG	Calls the respective subroutine, so e.g. XEQ C will be inserted when C is pressed.
		\PRG, \α	Executes the respective program if defined.
XNOR	h X.FCN XNOR	\α	Works in analogy to AND.
XOR	h XOR	\α	Works in analogy to AND.
x !	h !	FLOAT	
***x*⇔**	h x⇄ ***r***	\α	Swaps the contents of **X** and ***r*** . See above for complex x⇔ .
x*⇔*y	x⇄y	\α	Swaps ***x*** and ***y*** , performing Re⇔Im if a complex operation was executed immediately before. See above for ^{C}x⇔y .
x → α	h X.FCN X→α	All	Interprets ***x*** as a code of up to 6 characters. Appends these characters to ***alpha***, similar to XTOA in HP-42S.

Name	Keys to press	in modes	Remarks
***x* < … ?**	f x < ? ***a***	\α	Compares ***x*** with ***a*** and executes the next program line if true, else skips this line. The three dots will be replaced in the listing by ***a*** according to the examples given in the *addressing table above*. CPX f x = ? ***a*** and CPX g x ≠ ? ***a*** compare the complex number ***x+iy*** as explained in the *addressing table above*.
x ≤ … ?	h TEST x ≤ ? ***a***		
***x* = … ?**	f x = ? ***a***		
***x* ≠ … ?**	g x ≠ ? ***a***		
x ≥ … ?	h TEST x ≥ ? ***a***		
x > … ?	h TEST x > ? ***a***		
x̄, ȳ	f x̄	FLOAT	Recalls $\frac{1}{n}\sum y$ and $\frac{1}{n}\sum x$ and pushes them on the stack like the function s does.
x̄w	h STAT x̄w	FLOAT	Recalls the weighted mean $\sum xy \big/ \sum y$.
x̂	h STAT x̂	FLOAT	Returns a forecast ***x*** for a given *y* according to the fit model chosen. See L.R. for more.
y y^x	g y^x	\α	In integer modes ***x*** must be ≥ 0.
	C	\(α, -3, -4, -5, h)	Shortcut as long as label C is not defined yet.
Y.MD	h X.FCN Y.MD	FLOAT	Sets the format for date calculations.
ŷ	f ŷ	FLOAT	Returns a forecast *y* for a given ***x*** according to the fit model chosen. See L.R. for more.
αDATE	h X.FCN αDATE	\integer	Assumes **X** containing a date and appends it to ***alpha*** in the format selected.
αDAY	h X.FCN αDAY	\integer	Assumes **X** containing a date, recalls the name of the respective day and appends its first 3 letters to ***alpha***.
αIP	h X.FCN αIP	All	Appends the integer part of ***x*** to ***alpha***, similar to AIP in HP-42S.
αLENG	h X.FCN αLENG	All	Returns in the numeric display the number of characters found in ***alpha***, like ALENG in HP-42S.
αMONTH	h X.FCN αMONTH	\integer	Works like αDAY, but processing the month.
αOFF	h P.FCN αOFF	PRG & α	Work like AOFF and AON in HP-42S.
αON	h P.FCN αON	PRG & \α	

Name	Keys to press	in modes	Remarks
αRCL	f RCL ***s***	α	Interprets the content of the source ***s*** as characters and appends them to ***alpha***.
	h X.FCN αRCL ***s***	\α	
αRC#	h X.FCN αRC# ***s***	All	Interprets the content of ***s*** as a number, converts it to a string in the format selected, and appends this to ***alpha***.
αRL	h X.FCN αRL ***n***	All	Rotates ***alpha*** by ***n*** characters like AROT in HP-42S, but with a positive parameter trailing the command instead of taken from **X**. αRL 0 executes as NOP. αRR works like αRL but rotates to the right.
αRR	h X.FCN αRR ***n***		
αSL	h X.FCN αSL ***n***	All	Shifts the ***n*** left-most characters out of ***alpha***, similar to ASHF in HP-42S. αSL 0 executes as NOP. αSR works like αSL but takes the ***n*** right-most characters instead.
αSR	h X.FCN αSR ***n***		
αSTO	f STO ***d***	α	Stores the first 6 characters in the alpha register into destination ***d*** .
	h X.FCN αSTO ***d***	\α	
αTIME	h X.FCN αTIME	FLOAT^H	Assumes **X** containing a time and appends it to ***alpha*** in the format selected
αVIEW	h X.FCN αVIEW	\α	Displays ***alpha***. In programs, use αVIEW followed by PAUSE for message output. Use PROMPT for parameter prompting – it equals αVIEW followed by STOP actually.
α → x	h X.FCN α→X	All	Returns the character code of the left-most character in ***alpha*** and deletes this character, like ATOX in HP-42S.
β(x,y)	h X.FCN β(x,y)	FLOAT	Calculates Euler's Beta $B(x,y)=\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ with $\operatorname{Re}(x)>0$, $\operatorname{Re}(y)>0$.
Γ(x)	h STAT Γ(x)	FLOAT	
	h X.FCN Γ(x)		
ΔDAYS	h X.FCN ΔDAYS	FLOAT	Assumes **X** and **Y** containing dates in the format chosen and calculates the number of days between them. Works like in HP-12C.
Δ%	f Δ %	FLOAT	Calculates $100\cdot\frac{x-y}{y}$ like %CH in HP-42S.

Name	Keys to press	in modes	Remarks
π	h π	FLOAT	Complex version copies π in **X** and clears **Y**.
Π	g π ***label***	FLOAT	Computes a product. At the beginning, **X** contains the loop control number in the format `cccccc.fffii` and the product is set to 1. Each run through the routine specified computes a factor. At the end of this run, this factor is multiplied with said product; the operation decrements `ccccccc` by `ii` and runs said routine again if `ccccccc` is then > `fff` , else returns with the resulting product in **X**.
Σ	g Σ ***label***	FLOAT	Computes a sum. The sum is cleared at the beginning. Each run through the routine specified computes a summand. At its end, this is added to said sum. Loop control is as in **Π** .
σ	h STAT σ	FLOAT	Calculates the standard deviation of the population and pushes the results on the stack like the function s does.
Σln^{2}x	h STAT Σln^{2}x	FLOAT	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.
Σln^{2}y	h STAT Σln^{2}y		
Σlnx	h STAT Σlnx		
Σlnxy	h STAT Σlnxy		
Σlny	h STAT Σlny		
Σxlny	h STAT Σxlny		
Σylnx	h STAT Σylnx		
Σx	h STAT Σx	FLOAT	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	h STAT Σx^2		
Σxy	h STAT Σxy		
Σy	h STAT Σy		
Σy^2	h STAT Σy^2		
Σ+	**Σ+**	FLOAT	
Σ–	h Σ–		
χ^2(x)	h STAT χ^2(x)	FLOAT	χ^2 works like Q(χ^2) , the inverse like χ^2_p in HP-21S. The degree of freedom is in **J**. Similar functions are found in the HP 30b now.
χ^2INV	h STAT χ^2INV		

Name	Keys to press	in modes	Remarks
+, −, ×, /	+ , − , × , /	\α	
+/−	+/−		
//	g //	FLOAT	Calculates $\left(1/x + 1/y\right)^{-1}$.
[.] or [,]	.	FLOAT	Inserts the radix mark as selected.
		α	Inserts a point.
[.]	.	D.MY etc.	Separates the leading unit in date modes. The user may decide where a number represents a date or not.
[ ] or [/]	.	/c	First . is interpreted as a space, 2nd as a fraction mark, e.g. 2 . 3 . 4 results in 2 ¾ in the dot matrix display. Proper fractions are entered starting with a . .
[°]	.	H.MS	Separates degrees (hours) from minutes and seconds, so input format is `hhhh.mmssdd`.
%	g %	FLOAT	Calculates $\frac{x \cdot y}{100}$.
%MG	h X.FCN h % MG	FLOAT	Calculates the margin[16] $100 \cdot \frac{x-y}{x}$ in percent for a price ***x*** and cost ***y*** like %MU-Price does in HP-17B.
%MRR	h X.FCN h % MRR	FLOAT	Calculates the mean rate of return in % per period, i.e. $100 \cdot \left[\left(\frac{x}{y}\right)^{1/z} - 1\right]$ with ***x*** = FV = future value after ***z*** periods, ***y*** = PV = present value. For ***z*** = 1 , Δ% returns the same result easier.
%T	h X.FCN h % T	FLOAT	Calculates $100 \cdot \frac{x}{y}$, i.e. percent of total FWIW.
%Σ	h STAT h % Σ	FLOAT	Calculates $100 \cdot \frac{x}{\sum x}$.
%+	h %+	FLOAT	Adds a markup of ***x*** % to a price ***y*** , calculating $y \cdot \left(1 + \frac{x}{100}\right)$ like in %MU-Cost of HP-17B.

[16] Margin corresponds to „Handelsspanne“ in German.

Name	Keys to press	in modes	Remarks
%+MG	h X.FCN h % +MG	FLOAT	Adds a margin of ***x*** % to the cost ***y*** , calculating a sales price $y \Big/ \left(1-\frac{x}{100}\right)$ like %MU-Price does in HP-17B.
%–	h %–	FLOAT	Subtracts a discount of ***x*** % from the price ***y*** , calculating $y \cdot \left(1-\frac{x}{100}\right)$.
√‾	f √x	\α	
	D	\(α, -4, -5, h)	Shortcut as long as label D is not defined yet.
∫	f ∫ ***label***	FLOAT	Integrates the function given in the routine specified. Lower and upper integration limits must be supplied in **Y** and **X**, respectively. Otherwise, the user interface is as in HP-15C.
∞?	h TEST ∞?	PRG	Tests ***x*** for infinity and executes the next program line if true, else skips this line.
→DEG	➡ g DEG	FLOAT	Assumes **X** containing an angle in current angular mode and converts it to degrees. Angular mode is kept.
→GRAD	➡ g GRAD	FLOAT	Works like →DEG, but converts to grads.
→HR	➡ f H	H.MS	Assumes **X** containing hours or degrees in the format `hhhh.mmssdd` and converts them into decimal numbers.
→H.MS	➡ g H.MS	FLOAT	Assumes **X** containing *decimal* hours or degrees and converts them into the format `hhhh.mmssdd`.
→POL	g ▶P	FLOAT	Assumes **X** and **Y** containing the coordinates *(x, y)* and converts them to *(r , ϑ)* .
→RAD	➡ g RAD	FLOAT^{H}	Works like →DEG, but converts to radians.
→REC	f ▶R	FLOAT	Assumes **X** and **Y** containing the coordinates *(r , ϑ)* and converts them to *(x, y)* .

Non-programmable control, clearing and information commands:

Name	Keys to press	in modes	Remarks
	g OFF	All	Turns calculator off.
	ON	Calc. off	Turns calculator on.
	▲ / ▼	Status open	Go to previous / next set of flags.
		Cat. open	Go to previous / next item in this catalogue.
		α	Shifts the display window to the left / right in alpha register if applicable. Useful for longer text strings.
		PRG	Like BST / SST in HP-42S.
		Else	
	f ◀ and g ▶	Integer	Shifts the display window like in HP-16C. Useful for integers with small bases.
	f ↑	α	Toggles upper and lower case.
	←	Input pending	Deletes last digit or character put in.
		PRG	Deletes current step if no input is pending.
	f α	\α	Toggle alpha mode for keyboard entry.
	ENTER↑	α	
CLALL	h X.FCN CLALL	\PRG	Clears all registers and programs if confirmed.
CLPR	h CLPR	\α	Clears the current program after confirmation. This program is the one the program pointer is in.
EXIT	EXIT	All	Exits catalogues and other operations with pending input, canceling the execution of said operation.
PRGOFF	h P/R	PRG	Leaves programming mode.
PRGON	h P/R	\PRG, \α	Enters programming mode.
RESET	h X.FCN RESET	All	Clears all registers and programs, and resets all modes to start-up default if confirmed. This default is FIX 4, DEG, Y.MD, LinF, 24h, 2COMPL, DENANY, DENMAX 9999, FLOAT, PROFRC, SSIZE4, WSIZE 64.
	h P.FCN RESET		
R/S	R/S	\PRG, \α	Runs a program (starting with the current program line) or stops a running program.

Name	Keys to press	in modes	Remarks
SHOW	h SHOW	FLOAT & \PRG	Shows the full mantissa until this key is released.
		\FLOAT & \PRG	= NOP
		PRG	Displays a CRC-32 checksum of program memory contents (8 hex digits), allowing validation of program integrity.
STATUS	h STATUS	\PRG	Shows the status of user flags, similar to STATUS on HP-16C. See the paragraph above.
VERS	h X.FCN VERS	\PRG	Shows the firmware version.
→BIN	➡ f 2	\α, \h, \-5	These commands show ***x*** in target integer representation until the next key is pressed. Mode is kept. If used in bases 15 and 16, prefix f must precede the key ➡
→DEC	➡ f 10		
→HEX	➡ g 16		
→OCT	➡ g 8		

Catalogues (not programmable):

Calling a catalogue will set temporary alpha mode to allow for typing the first 1 or 2 characters of the item wanted. ▲ and ▼ browse the catalogue, ENTER↑ selects the item displayed and exits, while EXIT leaves the catalogue without executing anything, returning to the mode as set before. See the *table above about addressing catalogued items*, and the *next paragraph* for detailed item lists.

Name	Keys to press	in modes	Contents
CONST	h CONS	FLOAT	Constants like in HP35s. See them listed in a separate *table below*. CPX h CONS will clear **Y** in recalling the constant selected.
CONV	h CONV	FLOAT	Conversions as listed in a separate *table below*.
CPX	f CPX	α	"Complex" letters mandatory for languages beyond English. Upper or lower case will be displayed according to setting (see f ↑ above).
P.FCN	h P.FCN	\α	Extra programming functions.
STAT	h STAT	FLOAT	Extra statistical functions.
		α	Some special letters for statistics.
TEST	h TEST	PRG	Contains all tests except those on the keyboard.
		α	Comparison symbols and brackets. Parentheses are called by f (and g) , respectively.
X.FCN	h X.FCN	FLOAT	Extra real functions.
		Integer	Extra integer functions.
		α	Extra alpha functions.
	CPX h …	FLOAT	Extra complex functions.
. / ,	h ./,	α	Punctuation marks and text symbols.
→	f →	α	Arrows and mathematical symbols.

DETAILED CATALOGUE CONTENTS

Here the contents of the catalogues X.FCN, P.FCN, STAT, and TEST are listed. A single operation, e.g. BASE, may be contained in more than one catalogue. The characters necessary to access a specific function in the respective catalogue are printed bold in this table – ▼ has to be pressed once for each character printed red – if even the last letter of a function name is red, one may need more strokes of ▼ to access this function. The alpha catalogues are found further below. See also the catalogues CONST and CONV in separate paragraphs.

Content of X.FCN in …		
… FLOAT	… integer modes	… alpha mode
12h	**1**COMPL	**CL**ALL
24h	**2**COMPL	**R**ESET
ALL	**A**SR	**V**ERS
ANGLE	**B**ASE	**x**→α
BASE	**C**B	α**D**ATE
CEIL	**CL**ALL	α**DA**Y
CLALL	**CLR**EG	α**I**P
CLREG	**D**BLR	α**L**ENG
DATE	**DB**L*	α**M**ONTH
DAY	**DBL**/	α**R**C#
DAYS+	**F**B	α**RL**
DECOMP	**FF**	α**RR**
DENANY	**FI**B	α**S**L
DENFAC	**G**CD	α**SR**
DENFIX	**L**CM	α**T**IME
DENMAX	**LJ**	α→x
DISP	**M**ASKL	
D.MY	**MA**SKR	
D→J	**MAX**	
D→R	**MI**N	
E3OFF	**MIR**ROR	
E3ON	**N**AND	
e^{x} -1	**nB**ITS	
…	…	

Content of CPX X.FCN
^{c}**e**^{x} -1
^{c}**F**IB
^{c}**L**N1+x
^{c}**LN**β
^{c}**LNΓ**
^{c}**S**IGN
^{c}**SI**NC
^{c}**W**
^{c}**W**$^{-1}$
^{c}**β**(x,y)
^{c}**Γ**(x)

Content of P.FCN
CB
CLFLAG
DROP
^{c}**DR**OP
DSZ
FB
FF
ISZ
NOP
OFF
ON
PROMPT
RCLM
R-CLR
R-COPY
R-SORT
R-SWAP
SB
STOM
αOFF
α**O**N

Content of STAT
BestF
B(k)
B $^{-1}$**(**p)
ERF
ExpF
Ex(x)
Ex $^{-1}$**(**p)
F(x)
F $^{-1}$(p)
Ge(k)
Ge $^{-1}$(p)
LinF
LogF
nΣ
N(x)
N $^{-1}$**(**p)
PowerF
P(k)
P $^{-1}$**(**p)
RAND#
SEED
SERR
SUM
…

Content of X.FCN (continued)			
… in FLOAT		… in integer modes	
FF	XNOR	NOR	WSIZE
FIB	x→α	RAND#	XNOR
FLOOR	Y.MD	RCLWS	x→α
GCD	αBEG	RESET	αBEG
IMPFRC	αDATE	RJ	αEND
Iβ	αDAY	RL	αIP
IΓ	αEND	RLC	αLENG
J→D	αIP	RR	αRCL
LCM	αLENG	RRC	αRC#
LN1+x	αMONTH	SB	αRL
LNβ	αRCL	SIGN	αRR
LNΓ	αRC#	SIGNMT	αSL
MAX	αRL	SL	αSR
MIN	αRR	SR	αSTO
M.DY	αSL	UNSIGN	αVIEW
NAND	αSR	VERS	α→x
NOR	αSTO		
PROFRC	αTIME		
RESET	αVIEW		
ROUNDI	α→x		
R→D	β(x,y)		
SETDAT	Γ(x)		
SETTIM	ΔDAYS		
SIGN	%MG		
SINC	%MRR		
TIME	%T		
VERS	%+MG		
W			
W^{-1}			

Content of TEST
BC?
BS?
FC?
FC?C
FC?F
FC?S
FS?C
FS?F
FS?S
LEAP?
NaN?
x ≤ ?
x ≥ ?
x > ?
∞?

Content of STAT (continued)
t(x)
t $^{-1}$(p)
Wb(t)
Wb $^{-1}$(p)
$\bar{x}$w
$\bar{\hat{x}}$
Γ(x)
σ
Σln^{2}x
Σln^{2}y
Σlnx
Σlnxy
Σlny
Σx
Σx^2
Σxlny
Σxy
Σy
Σy^2
Σylnx
χ^2(x)
χ^2INV
%Σ

Here are the contents of the alpha catalogues:

STAT
x̄
x̂
ȳ
ŷ

→
→
←
∫
°
▯$^{-1}$
▯^{2}
ħ
▯^{x}
^
∞

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

TEST
<
≤
=
≥
>
[
]
{
}

. / ,
,
“
#
‘
*
:
;
@
_
~

TABLE OF CONSTANTS

This lists all constants contained in the catalogue CONST. Values of physical constants (incl. their relative standard uncertainty given in parentheses below) are from CODATA 2006, copied in August 2010. Commas are used as radix marks for better visibility in this table. Green background denotes exact values. The more the background turns to red, the less precise the values are known. The characters necessary to get to a specific function in the catalogue are printed bold in this index – ▼ has to be pressed once for each character printed red.

For the units, please remember $1T = 1\frac{Wb}{m^2} = 1\frac{V \cdot s}{m^2}$.

	Numeric value	Unit	Remarks
a	365,2425	d	Gregorian year (per definition)
a₀	5,2917720859E-11 (6,8E-10)	m	Bohr radius $= \frac{\alpha}{4\pi \cdot R_\infty}$
c	2,99792458E8	${}^{m}/_{s}$	Vacuum speed of light (per definition)
c₁	3,74177118E-16 (5,0E-8)	$m^2 \cdot W$	First radiation constant $= 2\pi \cdot h \cdot c^2$
c₂	0,014387752 (1,7E-6)	$m \cdot K$	Second radiation constant $= {}^{hc}/_{k}$
e	1,602176487E-19 (2,5E-8)	C	Electron charge $= \frac{2}{K_J R_K} = \Phi_0 G_0$
eE	2,718281828459045	1	Euler's e. Please note the character *e* is used for the electron charge elsewhere in this table.
F	96485,3399 (2,5E-8)	$\frac{C}{mol}$	Faraday's constant = *e N_A*
g	9,80665	${}^{m}/_{s^2}$	Standard earth acceleration (per definition)
G	6,67428E-11 (1,0E-4)	$\frac{m^3}{kg \cdot s^2}$	Newton's gravitation constant
G₀	7,7480917004E-5 (6,8E-10)	${}^{1}/_{\Omega}$	Conductance quantum $= {}^{2e^2}/_{h} = {}^{2}/_{R_K}$ with the von Klitzing constant R_K = 25812,807557 Ω
g_e	2,0023193043622 (7,4E-13)	1	Landé's g-factor
h	6,62606896E-34 (5,0E-8)	$J\,s$	Planck constant
ħ	1,054571628E-34 (5,0E-8)		$= {}^{h}/_{2\pi}$

	Numeric value	Unit	Remarks
k	1,3806504E-23 (1,7E-6)	J/K	Boltzmann constant $= R/N_A$
$\mathbf{m_e}$	9,10938215E-31 (5,0E-8)	*kg*	Electron mass
$\mathbf{m_n}$	1,674927211E-27 (5,0E-8)		Neutron mass
$\mathbf{m_p}$	1,672621637E-27 (5,0E-8)		Proton mass
$\mathbf{m_u}$	1,660538782E-27 (5,0E-8)		Atomic unit mass $= 10^{-3}$ *kg / N*$_A$
$\mathbf{m_\mu}$	1,88353103E-28 (5,6E-8)		Muon mass
$\mathbf{N_A}$	6,02214179E23 (5,0E-8)	$1/mol$	Avogadro's number
$\mathbf{p_o}$	101325	*Pa*	standard atmospheric pressure (per definition)
R	8,314472 (1,7E-6)	$\frac{J}{mol \cdot K}$	Molar gas constant
$\mathbf{r_e}$	2,8179402894E-15 (2,1E-9)	*m*	Classical electron radius $= \alpha^2 \cdot a_0$
$\mathbf{R_\infty}$	1,0973731568527E7 (6,6E-12)	$1/m$	Rydberg constant $= \alpha^2 m_e c / 2h$
$\mathbf{T_o}$	273,15	*K*	= 0°C, standard temperature (per definition)
$\mathbf{t_p}$	5,39124E-44 (5,0E-5)	*s*	Planck time $= \sqrt{\hbar G / c^5}$
$\mathbf{V_m}$	0,022413996 (1,7E-6)	m^3/mol	Molar volume of an ideal gas at standard conditions $= RT_0 / p_0$
$\mathbf{Z_o}$	376,730313461…	*Ω*	Characteristic impedance of vacuum $= \sqrt{\frac{\mu_0}{\varepsilon_0}} = \mu_0 c$
α	7,2973525376E-3 (6,8E-10)	1	Fine-structure constant $= \frac{e^2}{4\pi\varepsilon_0 \hbar c} \approx \frac{1}{137}$
γEM	0,57721566490153286…	1	Euler-Mascheroni constant
$\mathbf{\gamma_p}$	2,675222099E8 (2,6E-8)	$\frac{1}{s \cdot T}$	Proton gyromagnetic ratio $= 2\mu_P / \hbar$
$\mathbf{\varepsilon_o}$	8,854187817…E-12	$\frac{A \cdot s}{V \cdot m}$	Electric constant, vacuum permittivity $= \frac{1}{\mu_0 c^2}$

	Numeric value	Unit	Remarks
λ_c	2,4263102175E-12 (1,4E-9)	m	Compton wavelength of the electron $= h/m_e c$, of the neutron $= h/m_n c$, and of the proton $= h/m_p c$
λ_{cn}	1,3195908951E-15 (1,5E-9)		
λ_{cp}	1,3214098446E-15 (1,9E-9)		
μ_o	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	9,27400915E-24 (2,5E-8)	J/T	Bohr's magneton $= e\hbar/2m_e$
μ_e	-9,28476377E-24 (2,5E-8)		Electron magnetic moment
μ_n	-9,6623641E-27 (2,4E-7)		Neutron magnetic moment
μ_p	1,410606662E-26 (2,6E-8)		Proton magnetic moment
μ_u	5,05078324E-27 (2,5E-8)		Nuclear magneton $= e\hbar/2m_p$
μ_μ	-4,49044786E-26 (3,6E-8)		Muon magnetic moment
π	3,141592653589793	1	
σ_B	5,6704E-8 (7,0E-6)	$\frac{W}{m^2 K^4}$	Stefan-Boltzmann constant $= \frac{2\pi^5 k^4}{15 h^3 c^2}$
Φ	1,61803398874989485…	1	Golden ratio $= \frac{1+\sqrt{5}}{2}$
Φ_o	2,067833667E-15 (2,5E-8)	$V\,s$	Magnetic flux quantum $= h/2e = 1/K_J$ with the Josephson constant K_J = 4,83597891E14 Hz/V
∞		1	Infinity (may the Lord of Mathematics forgive us calling this a constant)

TABLE OF CONVERSIONS

These are the contents of CONV. The characters necessary to access a specific conversion there are printed bold in this index – ▼ has to be pressed once for each character printed red. The constant **T₀** may be useful for conversions, too; it is found in the *catalogue CONST*. The conversion factors or divisors listed in this table will not be seen when executing a conversion.

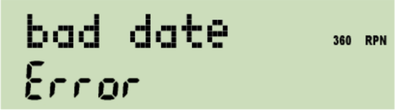
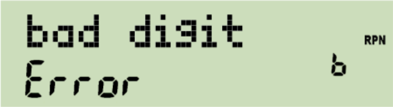
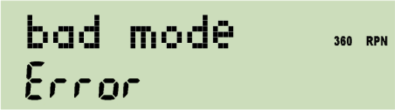
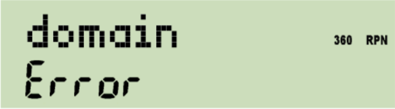
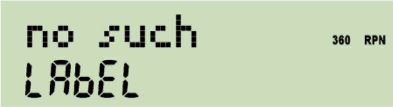
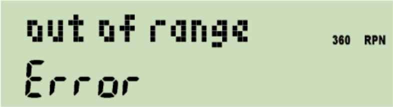
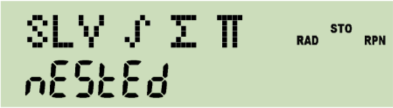
Conversion		Remarks	Class
acres→ha	* 0.4046873	Remember 1 ha = 10^4 m²	Area
atm→Pa	* 1.01325E5	Exactly	Pressure
au→km	* 1.495979E8	Astronomic units	Length
bar→Pa	* 1E5	Exactly	Pressure
bhp→W	* 745.6999	British horse power	Power
Btu→J	* 1055.056		Energy
cal→J	* 4.1868	Exactly	Energy
cm→inch	/ 2.54	Exactly	Length
feet→m	* 0.3048	Exactly	Length
flozUK→*ml*	* 28.41306	Remember 1 *ml* = 1 cm^3	Volume
flozUS→*ml*	* 29.57353		Volume
galUK→ *l*	* 4.54609		Volume
galUS→ *l*	* 3.785418		Volume
g→oz	/ 28.34952		Mass
g→tr oz	/ 31.10348		Mass
ha→acres	/ 0.4046873		Area
HP$_e$→W	* 746	Exactly	Power
inch→cm	* 2.54	Exactly	Length
J→Btu	/ 1055.056		Energy
J→cal	/ 4.1868	Exactly	Energy
J→k Wh	/ 3.6E6	Exactly, since 1 h = 3600 s	Energy
kg→lbm	/ 0.4535924		Mass
km→au	/ 1.495979E8	Astronomic units	Length
km→*l.y.*	/ 9.460730E12	Light years	Length
km→mi	/ 1.609344	Exactly	Length
km→nmi	/ 1.852	Nautical miles, exactly	Length

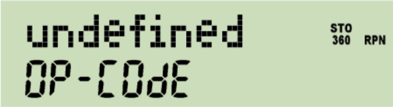
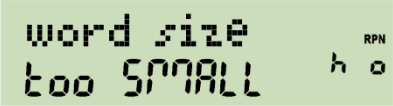
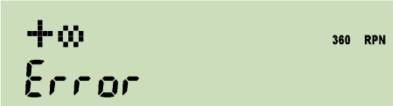
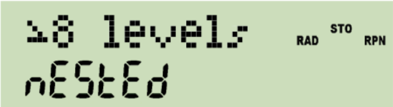
Conversion		Remarks	Class
km→**pc**	/ 3.085678E16	Parsec	Length
kWh→J	* 3.6E6	Exactly	Energy
lbf→N	* 4.448222		Force
lbm→kg	* 0.4535924		Mass
l.y.→km	* 9.460730E12	Light years	Length
l →galUK	/ 4.54609		Volume
l →**g**alUS	/ 3.785418		Volume
mi→km	* 1.609344	Exactly	Length
ml→flozUK	/ 28.41306		Volume
ml→flozUS	/ 29.57353		Volume
mmHg→Pa	* 133.3224	1 mm Hg = 1 torr	Pressure
m→feet	/ 0.3048	Exactly	Length
nmi→km	* 1.852	Nautical miles, exactly	Length
N→lbf	/ 4.448222		Force
oz→g	* 28.34952		Mass
Pa→atm	/ 1.01325E5	Exactly	Pressure
Pa→bar	/ 1E5	Exactly	Pressure
Pa→mmHg	/ 133.3224		Pressure
pc→km	* 3.085678E16	Parsec	Length
PS(hp)→W	* 735.4988		Power
sh ton→t	* 0.9071847	Remember 1 t = 10^3 kg	Mass
ton→t	* 1.016047		Mass
tr oz→g	* 31.10348		Mass
t→sh ton	/ 0.9071847		Mass
t→**t**on	/ 1.016047		Mass
W→bhp	/ 745.6999		Power
W→HP$_e$	/ 746	Exactly	Power
W→**P**S(hp)	* 735.4988		Power
°C→°F		Exactly	Temperature
°F→°C		Exactly	Temperature

MESSAGES

There are a few commands generating messages in the display. Two of them, STATUS and VERS, were introduced above in the *paragraph about display* already.

Furthermore, there are a number of error messages. Depending on error conditions, the following messages will be displayed:

Message	May appear in ...	Explanation and Examples
 bad date Error 	FLOAT	Invalid date format or incorrect date in input, e.g. month >12, day >31 etc.
 bad digit Error 	Integer	Invalid digit in integer input, e.g. 2 in binary, 9 in octal, or +/- in unsigned mode.
 bad mode Error 	All	Caused by calling an operation in a mode where it is not defined, e.g. SIN in hexadecimal.
 domain Error 	\α	An argument exceeds the domain of this mathematical function. May be caused by roots or logs of negative numbers (if not preceded by CPX), by 0 / 0, LN(0), Γ(0), TAN(90°) and equivalents, ATANH(x) for $\left\|\mathrm{Re}(x)\right\| \geq 1$, or ACOSH(x) for $\mathrm{Re}(x) < 1$, etc.
 no such LAbEL 	All	Attempt to address an undefined label.
 out of range Error 	All	• A number exceeds the valid range. Caused e.g. by specifying decimals >11, word size >64, negative flag numbers, integers ≥2^{64}, hours or degrees >9000, invalid times, denominators ≥9999 etc. • A register address exceeds the valid range. May also happen in indirect addressing. • A block register operation (e.g. R-COPY) attempts exceeding valid register numbers (0 … 99).
 SLV ∫ Σ Π nEStEd 	PRG	Nested use of solve, integrate, sum or product is not allowed.

Message	May appear in ...	Explanation and Examples
 undefined OP-CODE	All	An instruction with an undefined op-code occurred (should never happen, but who knows).
 word size too SMALL	Integer, \PRG	Stack or register content is too big for the word size set.
 +∞ Error (or –∞)	\α, \PRG	• Division of a number > 0 (or < 0) by zero. • Divergent sum or product or integral. • Positive (or negative) overflow in FLOAT.
 ≥8 levels nESTEd	PRG	Subroutine nesting exceeds 8 levels.

Any key pressed will wipe out the error message displayed and execute with the stack contents present.

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