



WP 43S REFERENCE MANUAL

This manual documents *WP 43S*, a free scientific software for the calculator *DM42* of *SwissMicros*. You can redistribute *WP 43S* and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

WP 43S is published and distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. Please see the GNU General Public License at <http://www.gnu.org/licenses/> for more details.

This manual is very preliminary; it will change while we develop *WP 43S* in course of this project. We reserve the right to do so at any time. The very basic principles of

WP 43S will stay constant, however. Stay informed by watching <http://sourceforge.net/p/wp43s/code/>.

Copyright © 2015 - 2019 Walter Bonin, Auf der Platte 9, 61440 Oberursel, Germany

All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without prior written permission of the author. For the time being, the locations highlighted cyan are open construction sites – information is missing there or needs further discussion and investigation to be determined. Any contributions in this matter are highly appreciated.

HP is a registered trade mark of Hewlett-Packard.

The pictures on p. 149 and bottom of p. 150 were kindly supplied by SwissMicros. The plots in Appendix J are based on material found in Wikipedia. The other photographs, pictures, diagrams, and graphics were created by the author.

Internet addresses are specified as found and verified at 2015-5-15. Please note such addresses may change without notice at any time.

This manual is published in English since it became the *lingua franca* of our time (after Greek, Latin, and French) – using it we can reach the maximum number of people without further translations. I apologize to the people of other languages and inserted some ‘translator’s notes’ where applicable.

Printed in the USA

ISBN-13: 978-172950106-1

ISBN-10: 172950106-0

WP 43S would not have been created without our love for *Classics*, *Woodstocks*, *Stings*, *Spices*, *Nuts*, *Voyagers*, and *Pioneers*. Thus we want to quote what was printed in Hewlett-Packard pocket calculator manuals until 1980, so it will not fade:

"The success and prosperity of our company will be assured only if we offer our customers superior products that fill real needs and provide lasting value, and that are supported by a wide variety of useful services, both before and after sales."

*Statement of Corporate Objectives
Hewlett-Packard*

DRAFT

TABLE OF CONTENTS

Welcome!	9
Print Conventions and Common Abbreviations	10
Section 1: Index of Items (<i>IOI</i>)	12
0 - 9	16
A	16
B	20
C	22
D	28
E	32
F	35
G	38
H	41
I	41
J	44
K	45
L	47
M	51
N	57
O	59
P	59
Q	63
R	63
S	71
T	79
U	81
V	82
W	83
X	84
Y	87
Z	88

A, α	88
β	90
Γ, γ	90
Δ, δ	90
ε	91
ζ	91
λ	91
μ	92
Π, π	92
Σ, σ	92
Φ, ϕ	94
X	94
ω	94
(, +, -, \times , /, \pm , \rightarrow	95
The Rest	99
Nonprogrammable Commands and Keys	104
Command Parameter Input and Closing It	105
Alphanumeric Input in X and Closing It	107
Section 2: Menus and Catalogs	111
One to Rule Them All – the CATALOG	111
Accessing Cataloged Items Rapidly	115
Menus and Their Contents	117
Constants	125
Unit Conversions	134
Section 3: Calling and Executing Operations	142
Using XEQ for Executing Operations	142
Operations Requiring Parameters	143
Operations Changing Data Types	145

Appendix A: Hardware	148
Appendix B: Memory Management	152
Data Types	152
Statistical Summation Registers	154
Range of Real Numbers	155
Special Results	158
Calculations with Double Precision (DP) Real Numbers	161
Appendix C: Messages and Error Codes	164
Appendix D: Display Internals	168
Segmentation	168
Character Sets	170
Appendix E: Comparison to the Function Sets of HP-42S, HP-16C, HP-21S, and WP 34S	176
Corresponding Operations on HP-42S	176
Corresponding Operations on HP-16C	182
Corresponding Operations on HP-21S	184
Corresponding Operations on WP 34S	186
New Commands on your WP 43S	190
Reference Literature	195
Appendix F: Flashing and Updating Your WP 43S	197
Appendix G: Overlays	198
Appendix H: Troubleshooting Guide	199
Appendix I: Emulating a WP 43S on Your Computer	200

Appendix J: Advanced Mathematical Functions and Tasks	201
Number Generating Functions	201
Statistical Distributions	203
More Statistical Formulas, also for Fitting	212
About Error Propagation	218
More about Curve Fitting	219
Solving Differential Equations	222
Orthogonal Polynomials	225
Even More Mathematical Functions	230
Appendix K: Background Considerations	235
Alpha Register	235
Angles	235
Display Limits	235
Plotting?	241
Stack Depth	243
Stack Lift Disabling Functions	244
Structured Programming	244
UNDO	244
Appendix L: Release Notes	245
WP 43S Quick Reference Guide	i
USING MENUS	i
MEMORY	i
DATA TYPES	ii
MODES	iii
EXECUTING FUNCTIONS AND PROGRAMS	iv
CLEARING AND DELETING	v
PROGRAMMING	vi

USING A VARIABLE MENU	vii
MATRIX OPERATIONS	vii
PROBABILITY	viii
STATISTICS	ix
ADVANCED OPERATIONS	x
OPERATIONS ON SHORT INTEGERS	xii
OPERATIONS ON ALPHANUMERIC STRINGS	xiii

DRAFT

WELCOME!

This is the reference volume of the *WP 43S* documentation. It supplements the *WP 43S Owner's Manual* with detailed information about each and every *item* (i.e. command, *menu*, *catalog*, browser, application, constant, conversion, digit, and character) provided in your *WP 43S*. The *Index of Items* in *Section 1* takes almost half of this volume.

Section 2 presents the structure and contents of all *menus* and *catalogs*. *Section 3* shows further access methods to operations and lists all operations requiring at least one parameter.

Twelve appendices cover additional special topics as listed in the *Table of Contents* above.

Enjoy!

Walter Bonin

DRAFT

Print Conventions and Common Abbreviations

- Throughout this manual, standard font is Arial. Emphasis is added by underlining or **bold** printing. Calculator COMMANDS and MENUS are called by their names, principally printed in capitals in running text (*menus* underlined). **Translator's footnotes are printed blue.** Specific terms, titles, trademarks, names or abbreviations are printed in italics, hyperlinks in blue underlined italics. The latter will beam you to its target in the original .pdf file – it cannot work in a printed copy for obvious reasons, thus either a page number or a full external address is given or the link points you to the *Table of Contents*.
- Bold italic Arial letters such as *n* are used for variables; bold normal letters for constant **sample values** (e.g. labels, numbers, or characters).
- This **KEY** font is taken for references to calculator keys, including **SOFTKEYS** in general. **Alphanumeric** and **numeric** calculator outputs (like 1.234×10^{-56} or $7,089 \cdot 10^{123}$) are printed using the respective calculator fonts.
- Courier is employed for file names and describing numeric formats.
- Times New Roman bold capitals are used for **REGISTER ADDRESSES**, lower case bold italics for *register contents*. So e.g. the value *y* lives in register **Y** and *r45* in **R45**. Overall stack contents are generally quoted in the order [*x, y, z, ...*]. We keep the term *register* for the space where an individual object is stored, although the actual size of such a *register* may vary widely following the size of the object.
- Lower case Times New Roman regular letters are for unit symbols (except in formulas - since the formula editor refuses to do this). Italics of this font are for *unit names*.
- We will use decimal points in most parts of this manual (but you may set your WP 43S to commas as well, of course). Although that point is less visible than a comma, ‘comma people’ seem to be more

tolerant against points used as radix marks than vice versa (based on the number of complaints read).

All this holds unless stated otherwise locally.

The following abbreviations are used throughout this manual:

ADM = angular display mode (see *Section 2* of the OM).

AIM = alpha input mode (see *Section 2* of the OM).

CDF = cumulated distribution function (see *Section 2* of the OM).

DP = double precision.

FM = flash memory (a special kind of *RAM*, s. *Sect. 3* of the OM).

HP = Hewlett-Packard.

IOI = *Index of Items* (see pp. 12ff).

LCD = liquid crystal display.

PDF = probability density function (see *Section 2* of the OM).

OM = Owner's Manual.

PEM = program-entry mode (see *Section 3* of the OM).

PMF = probability mass function (see *Section 2* of the OM).

px = pixels.

RAM = random access memory, allowing read and write operations.

RPN = reverse Polish notation (see *Section 1* of the OM).

SP = single precision.

SRS = subroutine return stack (see *App. B* on pp. 152ff).

TVM = *Time Value of Money* – a preprogrammed application for dealing with investments and credits, featured by all financial *HP* calculators since 1972 (s. *Sect. 5* of the OM).

Some more abbreviations may be used and explained locally.

SECTION 1: INDEX OF ITEMS (IOI)

All the *items* provided on your *WP 43S* (more than 850) are listed below with their reserved *names* (as they are printed in routines) and the key-strokes necessary to call them. Most *items* shall be picked from *menus* (see pp. 111ff) or *catalogs*.¹

Sorting in this index, in CATALOG, and CNST works in the following order:

_ 0...9 Aα...Zz Aα...Ωω () + - × / ±
, . ! ? : ; ' " · * @ _ ~ → ← ↑ ↓ ⚡
< ≤ = ≈ ≠ ≥ > % \$ € £ ¥ √ ∫ ∞
& \ ^ | [] { } ☎ #

Superscripts and subscripts are handled like normal characters in sorting. The ☎ above is the printer symbol heading all print commands.

In principle, *WP 43S* operations work as the corresponding ones did on the *WP 34S* where applicable (cf. App. E). Referring to vintage calculators, most functions and keystroke-programming will work as they did on the *HP-42S*, bit and integer functions as on the *HP-16C*, unless specified otherwise. Also for functions inspired by other vintage calculators as mentioned in the index below, their manuals may contain helpful additional information.

Operations working with the accumulated statistical data are marked light blue. Those operating on complex parameters are marked light yellow. Operations asking you for confirmation are printed red.

Some 300 functions featured in your *WP 43S* are new compared to HP's *RPN* pocket calculators.² Operations carrying familiar names but

¹ For commands stored in *menus*, we list the keys calling the respective *menu*, the *prefix(es)* of the respective *menu* row (if applicable), and the command as shown therein. We are confident you will find the corresponding *softkey*. *Items* stored in CNST are listed with their reserved names only, however, since they will be explained in detail in a separate chapter below.

² We did not compare the *RPL* calculators of the last three decades or the *HP Prime*.

deviating in their functionality from previous *HP RPN* calculators or the *WP 34S* are marked light red.

For the vast majority of operations, remarks start with a number:

- (0) represents functions without any effects on the *stack* (e.g. mode setting functions);
- (1) is for *monadic functions*,
- (2) for *dyadic functions*, and
- (3) for *triadic functions* as defined in *Section 1* of the *OM*;
 - (-1) stands for functions pushing one object on the *stack* and
 - (-2) for functions pushing two objects on the *stack*.

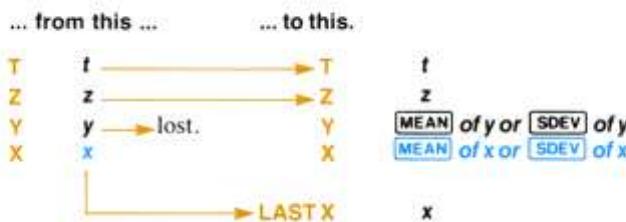
Note some functions overwrite two *stack* levels instead of pushing two values on it: e.g. →POL and →REC, as you may have expected.³

Operation or function **parameters** will be taken from the lowest *stack register(s)* unless mentioned explicitly in second column of the *IOI* –

They are exceeding the realm of shirt pocket calculators.

³ On the *HP-42S*, however, also statistical functions returning two values do that – while the *Spices* (e.g. *HP-34C*) and *Voyagers* (e.g. *HP-15C*) push both results on the stack instead as you expect from *RPN* calculators. For your information, the picture below shows what the *HP-55*, *HP-19C/29C*, *HP-67/97*, *HP-41C*, and *HP-42S* do there:

The illustration below shows what happens in the stack when you execute **[MEAN]** or **[SDEV]**. The contents of the stack registers are changed...



Alas, *HP* does not give any reason for this deviation from simple logic until today. In our opinion this is not reasonable, so for the *WP 43S* we stick to the paradigm as implemented on the *Voyagers* in this matter (as we did for the *WP 34S / 31S* before).

then they have to trail the command. Some parameters of statistical distributions shall be given in *registers I, J, and K* as specified.

Three examples of the parameter notation used throughout the *IOI* are shown below; assume **R12** contains **15.67** generally here, i.e. **r12 = 15.67**.

1. **n** represents an arbitrary integer number which must be keyed in directly, while
 - n** represents such a number which may be specified indirectly via a *register* or variable as well (as shown in the tables in *Section 1* of the *OM*); and
 - n** stands for the respective number itself;

Example:

RSD 12

rounds x to 12 significant digits, while

RSD →12

rounds x to 15 significant digits.

2. **r** (or **s**) represents an arbitrary *register* address or variable name which must be keyed in directly or picked from a *menu*, while
 - r** (or **s**) represents such an address or name which may be specified indirectly as well; and
 - r** (or **s**) stands for the contents of the address specified – **r** or **s** may be used as an address itself;

Example:

STO 12

stores x into **R12**, while

STO →12

stores x into **R15**.

3. **label** represents an arbitrary label which must be keyed in directly or picked from a *menu*, while
 - label** represents such a label which may be specified indirectly (as shown in the addressing table in *Section 3* of the *OM*); and
 - label** stands for the respective label itself, regardless of the way it was specified.

Example:

GTO 12

goes to local label **12**, while

GTO →12

goes to local label **15**.

Note that for any command XYZ requiring one trailing parameter, you can enter XYZ → ST.X and it will take its parameter from X instead.

Automatic stack lift is enabled after each command – except CLX, ENTER↑, Σ+, and Σ- (cf. Section 1 of the OM); numeric input immediately following one of these four operations will overwrite x instead of pushing it on the stack as usual.⁴

The *data types* a particular function operates on are listed in { } under “remarks” if there are restrictions – cf. App. B on pp. 152ff. Most bit and integer functions operate on *short integers* only (*data type* 10). The other functions typically work with more kinds of objects. Functions stating *data types* 8* or 9* instead of 8 or 9 operate on each matrix element instead of the entire matrix (as explained in Section 2 of the OM). Wherever operations return *data types* differing from their input, the output types are listed as well.⁵

All operations may be entered in PEM as well unless stated otherwise – many functions contained in P.FN and TEST will make most sense in PEM.

Below, the functions checked already are highlighted green, those which didn't work (for whatever reason) are marked red. Green highlighting doesn't necessarily mean the function works correctly but its results look like in the right ballpark. What wasn't checked so far isn't highlighted at all. This applies to the respective *data types*.

⁴ Some reasoning why *automatic stack lift* is disabled for these four:

- a) CLX is for clearing X to make room for a corrected value.
- b) ENTER↑ is a *stack lift* manually initiated by the user. An additional *automatic stack lift* immediately after this command would not make sense.
- c) Σ+ and Σ- are dedicated commands for adding or subtracting data points (see the chapter about *Statistical Calculations* in the OM). They were exclusively designed for data input since their first appearance on the HP-45 and are not really meant to be mixed with calculations.

⁵ E.g. for °C→°F: For single (double) precision real input, output stays single (double) precision real. For integer input, however, output is single precision real.

Some functions operating on *long integers* will return either such integers or reals, depending on the input value. See the OM, Sect. 2, *Integers: Summary of Functions*. $\sqrt[3]{x}$ is an example: it will return 3 for an input of 27, i.e. for a proper cube.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	etc.	(1) {2, 11}; {1} → {2} Convert temperatures. See pp. 133ff.
$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$		
10^x		(1) {1, 2, 3, 8*, 9*, 10, 11, 12} Returns 10^x , the inverse of $\lg(x)$.
1COMPL	▼ 1COMPL ▲ 1COMPL	(0) Sets 1's complement mode for operations on <i>short integers</i> . See Sect. 2 of the OM.
$\frac{1}{2}$		(-1) {} → {2} Trivial but helpful constant for iterations.
$\frac{1}{x}$		(1) {2, 3, 8*, 9*, 11, 12}; {1} → {2} Inverts the number x or all elements of the matrix x .
2COMPL	▼ 2COMPL ▲ 2COMPL	(0) Sets 2's complement mode for operations on <i>short integers</i> . See Sect. 2 of the OM.
2^x		(1) {1, 2, 3, 8*, 9*, 10, 11, 12} Returns 2^x .
$\sqrt[3]{x}$		(1) {1, 2, 3, 8*, 9*, 10, 11, 12}; {{1}} → {2} Returns the cube root of x .
A		Reserved variable for <i>register A</i> .
a		(-1) {} → {2} <i>Gregorian year in days</i> .
a_0		(-1) {} → {2} <i>Bohr radius in meter</i> .

Item	Keystrokes	Remarks (see pp. 12ff for general information)
ABS	f CATALOG FCNS ABS	Works like $ x $ on p. 101. ABS is featured for backward compatibility only.
ACC	f ADV ∫fdx ACC	Reserved real variable for the accuracy of integration (see Sect. 4 of the OM).
ACOS	f CATALOG FCNS ACOS	Works like arccos below. ACOS is featured for backward compatibility only.
ac→m ²	g U f A: acre → m ²	(1) {2, 11}; {1} → {2}
ac _{us} →m ²	g U f A: acre _{us} → m ²	Convert areas. See pp. 133ff.
ADV	f ADV	Menu. See p. 117.
AGM	g X.FN AGM	(2) {2, 3, 11, 12}; {1} → {2} Returns the <i>arithmetic-geometric mean</i> of x and y . Will throw an error for x or y being negative. See p. 230 for more.
AGRAPH	g P.FN P.FN2 g AGRAPH s	(0) Alpha graphics. Displays a graphics image. Each character in the source s specifies an 8-dot column pattern. The X- and Y-registers specify the pixel location of the bottom left point of this column. $1 \leq x \leq 400$ and $1 \leq y \leq 232$ are possible (but cf. App. K on pp. 241ff). So one row (8 px high) starting in column 1 may need up to 400 characters to specify – the more blank space is found therein the less characters may be required for describing it entirely. Cf. <i>HP-42S Owner's Manual</i> , pp. 135 – 140, and <i>HP-42S Programming Examples and Techniques</i> , pp. 214 – 223.
ALL	g DISP ALL n	(0) Sets the numeric display format to show all decimals of real or complex numbers whenever possible. ALL 00 works like ALL in HP-42S almost.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
		For $x \geq 10^{10}$ (or earlier for complex numbers), however, display will switch to SCI or ENG with the maximum number of decimals necessary and displayable using the <u>large</u> font (see SCIOVR and ENGOVR). The same will happen if $x < 10^{-n}$ and more than 16 digits are required to show x entirely.
a_M	g [CNST] a_M	(-1) {} → {2} Semi-major axis of the Moon's orbit around the earth in <i>meter</i> .
AND	f [BITS] AND	(2) {10} Works bitwise as in HP-16C (see Sect. 2 of the OM).
		(2) {1, 2, 11} → {1} Works like AND in HP-28S, i.e. x and y are interpreted before executing this operation. Zero is 'false'; any other real number is 'true'.
\arccos	[TRI] arccos	(1) {3, 8*, 9*, 12}; {1, 2, 11} → {4}; Returns the tagged angle $\arccos(x)$. ⁶
arcosh	g [EXP] g arcosh	(1) {2, 3, 8*, 9*, 11, 12} Returns $\text{arcosh}(x)$.
\arcsin	[TRI] arcsin	(1) {3, 8*, 9*, 12}; {1, 2, 11} → {4}; Returns the tagged angle $\arcsin(x)$. ⁷

⁶ Precisely, ARCCOS returns the principal value of $\arccos(x)$, i.e. a real part $\in [0, \pi]$ in $\frac{\pi}{4}r$, or $\in [0^\circ, 180^\circ]$ in $\frac{\pi}{4}^\circ$ or $\frac{\pi}{4}''$, or $\in [0^g, 200^g]$ in $\frac{\pi}{4}^g$, or $\in [0, 1]$ in $\frac{\pi}{4}\pi$. Cf. ISO/IEC 9899.

⁷ Precisely, ARCSIN returns the principal value of $\arcsin(x)$, i.e. a real part $\in [-\pi/2, \pi/2]$ in $\frac{\pi}{4}r$, or $\in [-90^\circ, 90^\circ]$ in $\frac{\pi}{4}^\circ$ or $\frac{\pi}{4}''$, or $\in [-100^g, 100^g]$ in $\frac{\pi}{4}^g$, or $\in [-0.5, 0.5]$ in $\frac{\pi}{4}\pi$. Cf. ISO/IEC 9899.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
arctan	TRI arctan	(1) {3, 8*, 9*, 12}; {1, 2, 11} → {4}; Returns the tagged angle $\text{arctan}(x)$. ⁸
arsinh	g EXP g arsinh	(1) {2, 3, 8*, 9*, 11, 12} Returns $\text{arsinh}(x)$.
artanh	g EXP g artanh	(1) {2, 3, 8*, 9*, 11, 12} Returns $\text{artanh}(x)$.
ASIN	f CATALOG FCNS ASIN	Works like arcsin above. ASIN is featured for backward compatibility only.
ASR	f BITS f ASR n	(1) {10} Works like n (≤ 63) consecutive ASR commands in HP-16C, corresponding to a division of x by 2^n . ASR 0 executes as NOP, but loads L. See Sect. 2 of the OM.
ASSIGN	f ASN item, location	(0) Assigns an <i>item</i> (i.e. a function, a <i>menu</i> , a label, or a character) to a specified sequence of keystrokes, corresponding to a specific location on the keyboard or in a <i>menu</i> . See Section 6 of the OM. Not programmable.
ATAN	f CATALOG FCNS ATAN	Works like arctan above. ATAN is featured for backward compatibility only.
atm→Pa	g U F&p; f atm→Pa	(1) {2, 11}; {1} → {2}
au→m	g U x: au→m	Convert pressures and distances. See pp. 133ff.
A...Z	f CATALOG CHARS A...Z	Submenu of Latin letters. See pp. 111ff.
A:	g U f A:	Submenu. See p. 133.

⁸ Precisely, ARCTAN returns the principal value of $\text{arctan}(x)$, i.e. a real part $\in [-\pi/2, \pi/2]$ in \mathbb{R} , for example (cf. ASIN), if flag D is set. Else the result interval for ATAN becomes $(-\pi/2, \pi/2)$ in \mathbb{R} , for example. Cf. ISO/IEC 9899.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
a_{\oplus}		(-1) {} → {2} Semi-major axis of the Earth's orbit around the sun in <i>meter</i> .
B		Reserved variable for <i>register B</i> .
BACK	 	(0) Jumps <i>n</i> steps backwards ($0 \leq n \leq 255$) in a program. E.g. BACK 1 goes to the previous program step. If BACK attempts to cross an END, an error is thrown. Reaching step 000 stops program execution and lights  . Cf. SKIP. ATTENTION: If you edit a section of your routine crossed by one or more BACK, SKIP, or CASE jumps, this may well result in a need to manually maintain all those statements individually .
bar→Pa	 	(1) {2, 11}; {1} → {2} Converts pressures. See pp. 133ff.
BATT?		(-1) {} → {2} Measures the battery voltage in the range between 1.9V and 3.4V and returns this value.
bbl→m ³	 	(1) {2, 11}; {1} → {2} Converts volumes. See pp. 133ff.
BC?		(-1) {10} Tests if the specified bit in <i>x</i> is clear.
BEEP		(0) Sounds a sequence of four tones. See also TONE and QUIET.
BeginP	 	(0) Sets "Begin" mode in TVM: payments occur at the beginning of each period. Typical for savings plans and leasing. Cf. ENDP.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
BestF	g STAT ▲ g BestF	(0) Selects the 'best' curve fit model for the current statistical data by picking the one with maximum <i>correlation</i> (like BEST in HP-42S). Relevant for L.R., CORR, COV, s_{xy} , \hat{x} , and \hat{y} . Note ORTHOF is <u>not</u> part of the set of models under investigation. See pp. 203ff for more.
Binom	f PROB g Binom: Binom	(1) {2, 11}
Binom _e	etc.	<i>Binomial distribution</i> with the number of successes g in X , the probability of a success p₀ in I , and the sample size n in J . See p. 203 for more.
Binom _p		Binom ⁻¹ returns the maximum number of successes m for a given probability p in X , p₀ in I and n in J .
Binom:	f PROB g Binom:	Submenu. See p. 120.
BITS	f BITS	Menu. See p. 115.
B _n	g X.FN B_n	(1) {1, 2, 11}
B _n *	g X.FN B_n*	B _n and B _n * return the Bernoulli number for an integer n > 0 given in X , working with different definitions (see both formulas on p. 201).
BS?	f BITS g BS? n	(0) {10}
		Tests if the specified bit in x is set.
Btu→J	g U→ E: Btu→J	(1) {2, 11}; {1} → {2} Converts energies. See pp. 133ff.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
C		Reserved variable for <i>register C</i> .
c	g [CNST] c	(-1) { } → {2}
c ₁	g [CNST] c ₁	Speed of light in vacuum in <i>meter per second</i> ; first and second radiation constants in <i>Planck's Law</i> (see p. 127).
c ₂	g [CNST] c ₂	
cal→J	g [U-] E cal→J	(1) {2, 11}; {1} → {2} Converts energies. See pp. 133ff.
CASE	g [P.FN] P.FN2 g CASE s	(0) Works like SKIP below but takes the number of steps to skip from s. Example: Assume a program section: <pre> ... 100 CASE 12 101 GTO 01 102 GTO 02 103 GTO 07 104 GTO 05 105 LBL 01 ... 132 LBL 02 ... 153 LBL 05 ... 234 LBL 07 ... </pre> <p>In execution of this program, <i>r12</i> will be checked in step 100: if $r12 \leq 1$ then the program will proceed to step 101 and continue with a jump to step 105, for $r12 = 2$ the program will go to step 102, etc., resulting in a nice controlled dispatcher for $1 \leq r12 \leq 4$.</p>

Item	Keystrokes	Remarks (see pp. 12ff for general information)
		<p>ATTENTION: CASE might surprise you for $r12 > 4$ in the example above. Take care of the input you provide!</p> <p>If you edit a section of your routine crossed by one or more BACK, SKIP, or CASE jumps, this may well result in a need to manually maintain all those statements individually.</p>
CATALOG	f CATALOG	Catalog of everything. See pp. 111ff.
Cauch	f PROB	(1) {2, 11}
Cauch _e	f Cauch: Cauch etc.	Cauchy-Lorentz distribution (also known as Lorentz or Breit-Wigner distribution) with the location x_0 specified in I and the shape γ in J. See p. 206 for more.
Cauch _p		
Cauch ⁻¹		Cauch ⁻¹ returns x for a given probability p in X, with x_0 in I and γ in J.
Cauch:	f PROB f Cauch:	Submenu. See p. 120.
CB	f BITS g CB n	(1) {10} Clears the specified bit in x , i.e. sets it to 0.
CEIL	g INTS g CEIL	(1) {8*}; {1, 2, 11} → {1} Returns the smallest integer $\geq x$. Cf. FLOOR.
CF	g FLAGS CF n	(0) Clears the flag specified, i.e. sets it to 0.
	g CLR CF n	
CHARS	f CATALOG CHARS	Submenu of characters. See pp. 111ff.
CLALL	g CLR f CLall	(0) Clears all registers, user flags, variables, and programs in RAM. Modes will stay as they are. Not programmable. Cf. CLCVAR, CLFALL, CLPALL, CLREGS, CLSTK, and RESET.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
CLCVAR	 	(0) Clears all variables used in the <i>current program</i> , i.e. sets all such real and complex variables to 0., all integer ones to 0, all <i>time</i> variables to 0:00:00, all <i>date</i> variables to January 1 st of year 0, all character strings to zero length, and all the elements of all matrix variables used to zero.
CLFALL	 	(0) Clears all global and local user <i>flags</i> . See CF.
CLK		Menu. See p. 111.
CLK12	 	(0) Sets 12h time display mode: e.g. 1:23 will become 1:23am, 23:45 will become 11:45pm. Shortens the date display in the <i>status bar</i> to two digits for the year. Cf. CLK24.
CLK24	 	(0) Sets international 24h time display mode. Expands the date display in the <i>status bar</i> to four digits for the year. Cf. CLK12.
CLLCD	 	(0) Clears the <i>LCD</i> in the rectangular window north and west of the point <i>x, y</i> . I.e. all pixels $\geq x$ and $\geq y$ are cleared.
CLMENU	CLMENU	(0) Clears all <i>menu</i> key definitions for the programmable <i>menu</i> . See MENU.
	P.FN2 ...	
CLP	CLP	(0) Clears the <i>current program</i> in <i>RAM</i> or <i>FM</i> . Freed memory is returned to the pool of free space. Not programmable.
CLPALL	 	(0) Clears all programs in <i>RAM</i> . Cf. CLP.
CLR		Menu. See p. 117.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
CLREGS	g CLR f CLREGS	(0) Clears all global and local general purpose <i>registers</i> allocated (see also LOCR), i.e. sets all these registers to 0 . The contents of the stack and L are kept.
CLSTK	g CLR CLSTK 0 g FILL	Clears all <i>stack registers</i> currently allocated (i.e. either X ... T or X ... D). All other <i>register</i> contents are kept. Cf. CLREGS.
CLX	g CLR CLX ⬅	(1) Clears <i>register X</i> , disabling <i>automatic stack lift</i> . The shortcut works for closed <i>x</i> only. Cf. CLREGS.
CLΣ	g STAT g CLΣ g CLR CLΣ	(0) Clears the statistical summation <i>registers</i> and releases the memory allocated for them (see p. 154).
CNST	g CNST	Menu. See CONST below and pp. 133ff.
COMB	f PROB C_{yx}	(2) {1} Returns the number of possible <u>subsets</u> of <i>x</i> items taken out of a set of <i>y</i> items (i.e. choose <i>x</i> out of <i>y</i>). No item occurs more than once in a subset, and <u>different orders</u> of the same <i>x</i> items are <u>not counted</u> separately. Cf. PERM. (2) {2, 3, 11, 12} See pp. 201ff for the formula.
CONJ	g CPX conj	(1) {3, 9*, 12} Returns the complex conjugate of <i>x</i> .
CONST	g P.FN f CONST n	(-1) {} → {2} Returns the constant stored at position <i>n</i> in CONST (see pp. 133ff and the OM, Section 5). Allows for indirectly addressing these contents.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
CONVG?	 	(0) {2, 11} Checks for convergence by comparing x and y as determined by the lowest five bits of r . a) The very lowest two bits set the tolerance limit: $0 = 10^{-14}$, $1 = 10^{-24}$, $2 = 10^{-32}$. b) The next two bits determine the comparison mode using the tolerance limit set: $0 = \text{compare the numbers } x \text{ and } y \text{ relatively}$, $1 = \text{compare them absolutely}$. c) The top bit tells how special numbers are handled: $0 = \text{NaN and } \pm\infty \text{ are considered converged}$, $1 = \text{they are not considered converged}$. Now, $r = a + 4b + 16c$.
CORR	 	(-1) {} → {2} Returns the coefficient of correlation for the current statistical data and curve fit model. See pp. 203ff for more.
cos	 	(1) {2, 3, 8*, 9*, 11, 12}; {1, 4} → {2} Returns the cosine of the angle in X (see Section 2 of the OM for details).
cosh	 	(1) {2, 3, 8*, 9*, 11, 12} Returns the hyperbolic cosine of x .
COV	 	(-1) {} → {2} Returns the population covariance for the two data sets entered via depending on the curve fit model selected. See s_{XY} for the sample covariance and pp. 203ff for more.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
CPX	g CPX	Menu. See p. 118.
CPXi	g CPX g CPXi etc.	(0) Selects either the letter <i>i</i> or <i>j</i> for displaying the imaginary number <i>i</i> .
CPXRES	g MODE f CPXRES g FLAGS SF	(0) Allows for complex results of real number calculations. Cf. REALRES.
CPXS	f CATALOG VARS CPXS	Submenu of complex variables defined at execution time. See pp. 111f.
CPX?	g TEST ▲ CPX?	(0) Checks if <i>x</i> is complex. Returns true if X contains data of type 3, 9, or 12 with nonzero imaginary part.
CROSS	f MATX f cross	(2) {8} Requires two real 2D or 3D vectors in X and Y and returns their cross product. Crossing of 2D vectors works as for complex numbers.
	g CPX cross	(2) {3} → {2}; {12} → {11} When two complex numbers are crossed, your WP 43S simply returns a real number that is equal to the signed magnitude of the resulting moment vector.
ct→kg	g U→ m: f carat → kg	(1) {2, 11}; {1} → {2}
cwt→kg	g U→ m: cwt→kg	Convert masses. See pp. 133ff.
CX→RE	CC	(-1) {3} → {2}; {9} → {8}; {12} → {11} Cuts a closed complex object <i>x</i> , putting either <ul style="list-style-type: none"> • (for L) its real part in X and its imaginary part in Y, or • (for ⊕) magnitude in X and angle in Y.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
D		Reserved variable for <i>register D</i> .
DATE	g CLK DATE	(-1) {} → {6} Recalls the <i>date</i> from the real-time clock and displays it in the format selected. See D.MY, M.DY, and Y.MD. Furthermore, DATE shows the day of week (see Sect. 2 of the OM).
DATES	f CATALOG VARS f DATES	Submenu of date variables defined at execution time. See pp. 111f.
DATE→	g CLK DATE→	(-2) {2, 6, 11} → {1} Assumes <i>x</i> containing a <i>date</i> in the format selected (or a real number in corresponding format) and pushes its three components as integers on the <i>stack</i> . Reversible by →DATE.
DAY	g CLK f DAY	(1) {2, 6, 11} → {1} Assumes <i>x</i> containing a <i>date</i> in the format selected (or a real number in corresponding format) and extracts the day.
DBL?	g TEST ▲ f DBL?	(0) Checks if <i>x</i> contains a double precision real or complex number.
DBLR	g INTS g DBLR etc.	{10} Double <i>word</i> length commands for remainder, multiplication and division. ⁹
DBL×		DBLR and DBL / accept a double size dividend in Y and Z (most significant bits in Y), the divisor in X as usual, and return the result in X .
DBL/		DBL× takes <i>x</i> and <i>y</i> as factors as usual but returns the product in X and Y (most significant bits in X).

⁹ See the HP-16C Owner's Handbook, Section 4 (pp. 52ff).

Item	Keystrokes	Remarks (see pp. 12ff for general information)
dB→fr	[g] [U→] [◀] dB → field ratio	(1) {2, 11}; {1} → {2}
dB→pr	[g] [U→] [◀] dB → power ratio	Convert ratios. See pp. 133ff.
DEC	[f] [LOOP] [f] DEC <i>r</i>	(0) {1, 2, 10, 11} Decrements <i>r</i> by 1. Does not load L even for target address X.
DECOMP	[f] [PARTS] [f] DECOMP	(-1) {1, 2, 11} → {1} Decomposes <i>x</i> (after converting it to an <i>improper fraction</i> , if applicable), returning a stack [<i>denominator(x)</i> , <i>numerator(x)</i> , ...]. Reversible by division. Example: If X contains 2.25 then DECOMP will return <i>x</i> = 4 and <i>y</i> = 9.
DEG	[g] [MODE] DEG	(0) Sets the ADM to <i>decimal degrees</i> .
DEG→	[f] [L→] [f] $\frac{x}{y} \rightarrow \frac{x}{y}$	(1) {1, 2, 11} → {4} Converts angles as described on p. 140.
DENANY	[g] [MODE] [g] DENANY	(0) Sets default fraction display format like in HP-35S – any denominator up to the value set by DENMAX may appear. This is the most precise way of displaying a decimal number as a fraction with DENMAX given. Example: If DENMAX = 5 then DENANY allows denominators 1, 2, 3, 4, and 5.
DENFAC	[g] [MODE] [g] DENFAC	(0) Sets 'factors of the maximum denominator', i.e. the denominator may be an integer factor of DENMAX only. Example: If DENMAX = 60 then DENFAC will allow for denominators 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, and 60. Now you know why 60 was a holy number in ancient Babylon.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
DENFIX	g MODE g DENFIX	(0) Sets fixed denominator format, i.e. the one and only denominator allowed is the value set by DENMAX.
DENMAX	g MODE g DENMAX	(1) Works like /c on HP-35S, but the maximum legal denominator is 9 999. For $x < 1$ or $x > 9\ 999$, DENMAX will be set to 9 999. For $x = 1$, the current DENMAX setting is recalled, replacing x .
DET	f CATALOG FCNS DET	Works like M on p. 101. DET is featured for backward compatibility only.
DIGITS	f CATALOG DIGITS	Submenu of digits. See pp. 111f.
DISP	g DISP	Menu. See p. 118.
DOT	g CPX dot	(2) {3} → {2}; {12} → {11} Returns $Re(x) \cdot Re(y) + Im(x) \cdot Im(y)$
	f MATX f dot	(2) {8} → {2}; {9} → {3} Requires two matrices in x and y and returns their dot (scalar) product. The dot product is defined as the sum of the products of the corresponding elements in both matrices. Note both matrices must be of the same size; else DOT will throw an error. See Sect. 2 of the OM.
DROP	g DROP↓	Drops x ... from the stack. See Sect. 1 of the OM for details.
	g STK DROP	
DROPy	g STK f DROPy	Drops y

Item	Keystrokes	Remarks (see pp. 12ff for general information)
DSE	DSE	(0) {1, 2, 10, 11} Given $cccccc.ffffii$ in the source, DSE decrements r by ii , skipping next program step if then $cccccc \leq fff$. If r features no fractional part then fff is 0. If $ii = 0$, $cccccc$ will be decremented by 1. DSE does not load L even for destination address X . Note that neither fff nor ii can be negative, and DSE is only sensible with $cccccc > 0$.
DSL	DSL	(0) {1, 2, 10, 11} Works like DSE but skips if $cccccc < fff$.
DSTACK	 DSTACK	(0) Sets the maximum number of <i>stack registers</i> displayed. For an input of 1, only x will be shown directly above the <i>menu section</i> ; for 2, x and y will be displayed; maximum input is 4. Expanded views of e.g. matrices and multi-level returns like SUM will work as described in the OM regardless of the number chosen for DSTACK. In any case, command input will be echoed directly below the <i>status bar</i> . This command is for old-school calculator users who may feel distracted by a multitude of <i>stack registers</i> displayed changing simultaneously.
DSZ	DSZ	(0) {1, 2, 10, 11} Decrement r by 1 and skips the next step if $ r < 1$ thereafter. Does not load L even for target address X . Known from HP-29C, HP-67, and HP-16C.
D.MS	(for closed input)	(0) Sets the <i>ADM</i> to <i>sexagesimal degrees</i> .

Item	Keystrokes	Remarks (see pp. 12ff for general information)
D.MS→		(1) {1, 2, 11} → {4} Converts angles as described on pp. 140f.
D.MY		(0) Sets the format dd.mm.yyyy for date display.
D→J		(1) {2, 6, 11} → {1} Assumes x containing a date in the format selected (or a real number in corresponding format) and converts it to a Julian day number ¹⁰ according to J/G setting.
D→R		(1) {1, 2, 11} → {4}
D→D.MS		Convert angles as described on p. 140.
e		(-1) {} → {2}
e _E		Elementary charge in coulomb and Euler's e .
EIGVAL		(-1) {8, 9} Evaluates the matrix x and pushes a diagonal matrix containing its eigenvalues on the stack.
EIGVEC		(-1) {8, 9} Evaluates the matrix x and pushes a matrix containing its eigenvectors on the stack.
END		(0) Last command in a program and terminal for searching local labels as described in the OM Sect. 3. Works like RTN in all other aspects.
ENDP		(0) Sets "End" mode in TVM: payments occur at the end of each period. Typical for loans and investments. Cf. BEGINP.

¹⁰ Translator's note: Julian day number translates to "Julianisches Datum" in German and «jour Julien» in French. See the corresponding articles in Wikipedia for more.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
ENG	g DISP ENG n	(0) Sets engineer's display format (see <i>Section 2</i> of the OM).
ENGOVR	g DISP ▲ ENGOVR	(0) Defines that numbers exceeding the range displayable in ALL or FIX will be shown in engineer's format. Cf. SCIOVR.
ENORM	f MATX g ENORM	(1) {8, 9} → {2} Calculates the Euclidean norm of the matrix in X . The Euclidean norm is defined as the square root of the sum of squares of all matrix elements. Works like FNRM on HP-42S. For a vector, ENORM returns its length. Cf. $ x $ on p. 101.
ENTER↑	ENTER↑	(-1) Separates two entries in input. Copies x into Y , disabling <i>automatic stack lift</i> . See p. 108 and the OM (<i>Section 1</i>) for details.
ENTRY?	g TEST g ENTRY?	(0) Checks the (internal) entry <i>flag</i> . It is set if: <ul style="list-style-type: none">• any character is entered in A/M, or• any command is accepted for entry (be it via ENTER↑, a function key, or R/S with a partial command line). Useful in routines, e.g. after PAUSE.
EQN	g EQN	Menu . See p. 118.
EQ.DEL	g EQN f DELETE	Deletes an equation.
EQ.EDI	g EQN EDIT	Opens the <i>Equation Editor</i> to edit an existing equation.
EQ.NEW	g EQN NEW	Opens the <i>Equation Editor</i> to enter a new equation.

See *Section 4* of the OM.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
erf	g [X.FN] erf etc.	(1) {2, 11}; {1} → {2}
erfc		Returns the error function or its complement. See pp. 230ff for more.
ERR	g [P.FN] ERR <i>n</i>	(0) Raises the error specified. The consequences are the same as if the corresponding error really occurred, so e.g. a running routine will be stopped and the message will be thrown. See App. C on pp. 164ff for the respective error codes. Cf. MSG.
EVEN?	g [TEST] f EVEN?	(0) Checks if <i>x</i> is integer and even.
e^x	e^x	(1) {2, 3, 8*, 9*, 11, 12}; {1, 10} → {2} Returns e^x .
EXITALL	g [P.FN] P.FN2 EXITall	(0) Exits all menus.
EXP	g EXP	Menu. See p. 118.
ExpF	g STAT ▲ g ExpF	(0) Selects the exponential curve fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 203ff for more.
Expon	f PROB f Expon: Expon etc.	(1) {2, 11}
Expon _e		Exponential distribution with the rate λ in J. See pp. 203ff for more.
Expon _p		Expon ⁻¹ returns the survival time t_s for a given probability <i>p</i> in X, with λ in J.
Expon ⁻¹		
Expon:	f PROB f Expon:	Submenu. See p. 120.
EXPT	f PARTS EXPT	(1) {1, 2, 11} → {1} Returns the exponent <i>h</i> of the number $x = m \cdot 10^h$ displayed. Cf. MANT.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
e^x-1	g EXP f e^x-1	(1) {2, 3, 8*, 9*, 11, 12} For $x \approx 0$, this returns a more accurate result for the fractional part than e^x does.
e/m_e	g CNST e/m_e	(-1) {} → {2} Electron charge to mass ratio in <i>coulomb per kilogram</i> .
E:	g U→ E:	<i>_submenu</i> . See p. 133.
F	g CNST F	(-1) {} → {2} <i>Faraday constant in coulomb per mol.</i>
FAST	g MODE f FAST	(0) Sets the processor speed to 'fast'. This is <i>startup default</i> and is kept for fresh batteries. Cf. SLOW.
FB	f BITS g FB <i>n</i>	(1) {10} Inverts ('flips') the specified bit in x .
FBR	f a.FN g FBR	(0) Font browser. See <i>App. D</i> .
FCNS	f CATALOG FCNS	<i>_submenu</i> of provided functions. See pp. 111ff.
FC?	g FLAGS FC? <i>n</i>	(0) Tests if the specified <i>flag</i> is clear.
FC?C	g FLAGS f FC?C <i>n</i> etc.	(0) Tests if the specified <i>flag</i> is clear. Clears, flips, or sets this <i>flag</i> after testing, respectively.
FC?F		
FC?S		
FF	g FLAGS FF <i>n</i>	(0) Flips the <i>flag</i> specified.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
FIB	g X.FN f FIB	(1) {1} Returns the Fibonacci number (see pp. 201f).
		(1) {2, 3, 11, 12} Returns the extended Fibonacci number.
FILL	g FILL	Copies x to all stack registers.
FIN	g FIN	Menu. See p. 118.
FIX	g DISP FIX n	(0) Sets fixed point display format (see the OM, Sect. 2).
FLAGS	g FLAGS	Menu. See p. 118.
FLASH	f CATALOG PROGS FLASH	Submenu of global labels defined at execution time. See pp. 111f.
FLASH?	g INFO f FLASH?	(-1) {} → {1} Returns the number of free words in FM (1 word = 2 bytes).
FLOOR	g INTS g FL00R	(1) {8*}; {1, 2, 11} → {1} Returns the greatest integer $\leq x$. Cf. CEIL.
fm.→m	g U- x: ▲ fathom → m	(1) {2, 11}; {1} → {2} Converts distances. See pp. 133ff.
FP	f PARTS FP	(1) {1, 2, 8*, 10, 11} Returns the fractional part of x . Cf. IP.
FP?	g TEST f FP?	(0) Tests x for having a fractional part $\neq 0$.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
$F_e(x)$	f PROB F: $F_e(x)$ etc.	(1) {2, 11}
$F_p(x)$		<i>Fisher's F distribution.</i> $F_u(x)$ equals $Q(F)$ on HP-21S. The degrees of freedom are specified in I and J . See pp. 203ff for more.
$F(x)$		
$F^{-1}(p)$		
$fr \rightarrow dB$	g U Δ field ratio $\rightarrow dB$	(1) {2, 11}; {1} \rightarrow {2} Converts ratios. See pp. 133ff.
FS?	g FLAGS FS? n	(0) Tests if the specified flag is set.
FS?C	g FLAGS f FS?C n etc.	
FS?F		
FS?S		
$ft \rightarrow m$	g U $x: f ft \rightarrow m$	(1) {2, 11}; {1} \rightarrow {2}
$ft_{us} \rightarrow m$	g U $x: \Delta$ survey foot _{us} $\rightarrow m$	Convert distances. See pp. 133ff.
FV	g FIN TVM FV	(1) Reserved variable for the future value of your investment or loan in TVM.
$fz_{uk} \rightarrow m^3$	g U $f V: f floz_{uk} \rightarrow m^3$	(1) {2, 11}; {1} \rightarrow {2}
$fz_{us} \rightarrow m^3$	g U $f V: f floz_{us} \rightarrow m^3$	Convert volumes. See pp. 133ff.
F_α	g CNST F_α	(-1) {} \rightarrow {2}
F_δ	g CNST F_δ	<i>Feigenbaum's</i> α and δ .
F:	f PROB F:	Submenu. See p. 120.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
f'	g EQN f	Submenus for calculating the first or second derivative of a given equation. See the OM (Sect. 4) for more.
f''	g EQN f'	
f'(x)	f ADV f' (x) <i>lbl</i>	{1, 2, 11} → {2} f(x) [f'(x)] returns the 1 st [2 nd] derivative of the function <i>f(x)</i> at position <i>x</i> . This <i>f(x)</i> must be specified in a routine starting with LBL <i>lbl</i> . On return, Y, Z, and T will be cleared and the position <i>x</i> will be in L. See Section 4 of the OM for more. ATTENTION: f(x) and f'(x) fill all stack registers with <i>x</i> before calling the routine specified.
f"(x)	f ADV f f"(x) <i>lbl</i>	
F&p:	g U- F&p:	Submenu. See p. 133.
G	g CNST G	(-1) {} → {2} Newtonian constant of gravitation in $m^3 / kg\ s^2$; also called γ by other authors.
G_0	g CNST G_0	(-1) {} → {2} Conductance quantum in <i>siemens</i> .
GAP	g DISP ▲ f GAP <i>n</i>	(0) Defines the interval for inserting digit group separators in reals. For integers, the intervals are fixed to 4 digits for binary and 3 for any other base – except 4, 8, and 16 where the interval is 2. In input, gaps will always be inserted as chosen for reals. After GAP 0, <u>no</u> group separators will be displayed neither in reals nor integers at all. See App. D and Sect. 2 of the OM.
G_c	g CNST G_c	(-1) {} → {2} Catalan's (mathematical) constant.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
GCD	g INTS g GCD	(2) {1; 10} Returns the Greatest Common Divisor of x and y . ¹¹ This will always be positive.
g_d	g X.FN f g_d etc.	(1) {2, 3, 11, 12}; {1} → {2} Returns the Gudermannian function or its inverse. See p. 231 for details.
g_e	g CNST g_e	(-1) {} → {2} <i>Landé's electron g-factor.</i>
Geom	f PROB g Geom: Geom etc.	(1) {2, 11}
$Geom_e$		<i>Geometric distribution:</i> The CDF returns the probability for a first success after $m = x$ Bernoulli experiments. The probability p_0 for a success in each such experiment must be specified in J. See pp. 203ff for more.
$Geom_p$		
$Geom^{-1}$		$Geom^{-1}$ returns the number of failures f before 1 st success for given probabilities p in X, p_0 in J.
Geom:	f PROB f Geom:	Submenu. See p. 120.
$gl_{uk} \rightarrow m^3$	g U→ f V: gl_{uk} \rightarrow m^3	(1) {2, 11}; {1} → {2}
$gl_{us} \rightarrow m^3$	g U→ f V: gl_{us} \rightarrow m^3	Convert volumes. See pp. 133ff.
GM_{\oplus}	g CNST GM_{\oplus}	(-1) {} → {2} Newtonian constant of gravitation times the Earth's mass with its atmosphere included according to WGS84. ¹² Displayed in m^3/s^2 .

¹¹ See also LCM. Remember school?

Translator's notes for French readers: GCD correspond à PGCD en français.
 Translator's notes for German readers: GCD entspricht ggT auf Deutsch.

¹² See http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html

Item	Keystrokes	Remarks (see pp. 12ff for general information)
GRAD	g MODE GRAD	(0) Sets the <i>ADM</i> to <i>grad/gon</i> . ¹³
GRAD→	f L→ f g→4	(1) {1, 2, 11} → {4} Converts angles as described on pp. 140f.
GTO	f GTO <i>labl</i>	(0) In <i>PEM</i> : inserts an unconditional branch to <i>labl</i> . Else positions the program pointer to <i>labl</i> .
GTO.	f GTO . n	to step <i>n</i> (specify up to four digits until becoming unambiguous in used program memory).
	f GTO . <i>labl</i>	to the global label specified.
	f GTO . ▲	directly after previous END, going to the top of <i>current program</i> (see Section 3 of the OM).
	f GTO . ▼	directly after next END, going to the top of next program.
	f GTO . .	to the end of used program memory in <i>RAM</i> (i.e. to the final END statement).
g⊕	g CNST g⊕	(-1) {} → {2} Standard earth acceleration in m/s^2 .

¹³ This angular unit is also known as *gradian* in the English language.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
\hbar	g CNST \hbar	(-1) {} → {2} Planck constant in joule-second.
H_n	g X.FN Orthog H_n	(2) {2, 11}; {1} → {2}
H_{nP}	... Orthog f H_{nP}	Hermite polynomials for probability (H_n) and physics (H_{nP}). See p. 218 for details.
$hp_E \rightarrow W$	g U P : $hp_E \rightarrow W$	(1) {2, 11}; {1} → {2} Convert powers. See pp. 133ff.
$hp_M \rightarrow W$	etc.	
$hp_UK \rightarrow W$		
Hyper	f PROB g Hyper: Hyper	(1) {2, 11}
Hyper _e	etc.	Hypergeometric distribution with the number of successes g in X , the probability of a success p_0 in I , the sample size n in J , and the batch size n_0 in K . See pp. 203ff for the formula.
Hyper _p		Hyper ⁻¹ returns the maximum number of successes m for a given probability p in X , p_0 in I , n in J , and n_0 in K .
Hyper ⁻¹		
Hyper:	f PROB g Hyper:	Submenu. See p. 120.
\hbar	g CNST \hbar	(-1) {} → {2} $= h / 2\pi$, so-called Dirac constant in joule-second.
I		Reserved variable for register I.
IDIV	g INTS f IDIV	(2) {1, 10}; {2, 11} → {1} Integer division, working like 7 IP .
IDIVR	f CATALOG FCNS IDIVR	{1, 10}; {2, 11} → {1} Like IDIV but also returns the remainder in Y.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
iHg→Pa		(1) {2, 11}; {1} → {2} Converts pressures. See pp. 133ff.
Im		(1) {3} → {2}; {9} → {8}; {12} → {11}
		Returns the imaginary part of x . Cf. RE.
IMPFRC		(1) {2, 11} Allows only <i>improper fractions</i> in display (e.g. $\frac{5}{3}$ instead of $1\frac{2}{3}$). Displays any reals (with $ x < 10^6$) according to the settings by DEN... as <i>improper fractions</i> . Cf. PROFRC.
INC		(0) {1, 2, 10, 11} Increments r by 1. Does not load L even for target address X .
INDEX		(1) Indexes a named matrix. You can also index a matrix by editing it (see M.EDIT or M.EDIN). After exiting the <i>Matrix Editor</i> , the matrix is no longer indexed. See also <i>Matrix Utility Functions</i> in the <i>HP-42S Owner's Manual</i> , pp. 223ff.
INFO		Menu. See p. 119.
INPUT		Works in programs only: Recalls the content of the source specified into X , displays the name of the source along with r , and halts program execution, allowing you to enter or calculate a value; pressing R/S then stores x into said destination and continues program execution – pressing EXIT instead cancels INPUT, so R/S thereafter will continue with the source content as it was. If you use an input variable name undefined at execution time, INPUT automatically creates the variable with an initial value of zero.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
INTS	g INTS	Menu. See p. 119.
INT?	g TEST f INT?	(0) Tests x for being an integer, i.e. having a fractional part equal to zero. Cf. FP?.
INVRT	f CATALOG FCNS INVRT	Works like $[M]^{-1}$ on p. 101. INVRT is featured for backward compatibility only.
in. \rightarrow m	g U\rightarrow x: g in.\rightarrowm	(1) {2, 11}; {1} \rightarrow {2} Converts distances. See pp. 133ff.
IP	f PARTS IP	(1) {1, 2, 8*, 10, 11} Returns the integer part of x . Cf. FP.
ISE	f LOOP ISE r	(0) {1, 2, 10, 11} Given ccccccc.ffffii in the source, ISE increments r by ii, skipping next program step if ccccccc \geq fff then. If r has no fractional part then ffff = 0 and ii = 0. If ii = 0, ccccccc will be incremented by 1. ISE does not load L even for target address X. Note that neither ffff nor ii can be negative, but ccccccc can.
ISG	f LOOP ISG r	(0) {1, 2, 10, 11} Works like ISE but skips if ccccccc > fff.
ISZ	f LOOP ISZ r	(0) {1, 2, 10, 11} Increments r by 1, skipping next program step if then $ r < 1$. ISZ does not load L even for target address X. Known from HP-29C, HP-67, and HP-16C.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
I _{xyz}	g X.FN f I _{xyz}	(3) {1, 2, 11} Returns the regularized beta function.
IΓ _p	g X.FN f IΓ _p etc.	(2) {1, 2, 11} Returns the regularized gamma function (one of two kinds).
I+	f MATX ▲ I+	(1) Increments or decrements the row index of the indexed matrix. See INDEX and also J+, J-, RCLEL, STOEL, RCLIJ, and STOIJ.
I-	f MATX ▲ I-	
I/O	g I/O	Menu. See p. 119.
i%/a	g FIN TVM i%/a	(1) Reserved variable for the annual interest rate of your investment or loan in TVM.
J		Reserved variable for register J.
J _y (x)	g X.FN g J _y (x)	(2) {2, 11}; {1} → {2} J _y (x) returns the <i>Bessel function of first kind</i> and order <i>y</i> . See p. 232 for details.
J+	f MATX ▲ J+	(1) Increments or decrements the column index of the indexed matrix. If M.GROW is set and the pointers I and J are at the last element of the matrix, executing J+ creates a new row at the end of the matrix. See INDEX and also I+, I-, RCLEL, STOEL, RCLIJ, and STOIJ.
J-	f MATX ▲ J-	
J/G	g CLK ▲ f J/G	(0) {2, 6} Sets the date the Gregorian calendar was introduced in the region you are interested in. See Dates in Section 2 of the OM.
J→Btu	g U→ E: J→Btu etc.	(1) {2, 11}; {1} → {2}
J→cal		Convert energies. See pp. 133ff.

See p. 231 for more.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
J→D	g CLK f J→D	(1) {1} → {6} Takes <i>x</i> as a <i>Julian day number</i> ¹⁴ and converts it to a common <i>date</i> according to J/G (s. above) and the date format selected.
J→Wh	g U E: J→Wh	(1) {2, 11}; {1} → {2} Converts energies. See pp. 133ff.
K		Reserved variable for <i>register K</i> .
k	g CNST k	(-1) {} → {2} <i>Boltzmann constant in joule per kelvin.</i>
KEY		See KEYG and KEYX below.
KEYG	g P.FN P.FN2 KEYG key#, labl	Defines the label to be branched to (KEYG) or called (KEYX) when a particular softkey is pressed. KEYG and KEYX work in PEM only and will be translated to a program step
KEYX	g P.FN P.FN2 KEYX key#, labl	KEY key# GTO labl or KEY key# XEQ labl , respectively. Key numbers go from 1 to 18 with 1 corresponding to F1 , 9 to f F3 , and 14 to g F2 , for example.
KEY?	g TEST g KEY? r	(0) Tests if a key was pressed while a routine was running or paused. If <u>no</u> key was pressed in that interval, the next program step after KEY? will be executed; else it will be skipped and the code of said key will be stored in <i>r</i> . Key codes reflect the rows and columns on the keyboard (see the OM, Sect. 3).

¹⁴ Translator's note: *Julian day number* translates to "Julianisches Datum" in German and «jour Julien» in French. See the corresponding articles in Wikipedia for more information about these numbers.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
kg→ct	[g] [U→] [m:] [f] kg → carat	
kg→cwt	[g] [U→] [m:] kg→cwt	
kg→lb.	etc.	
kg→oz		
kg→scw	[g] [U→] [m:] [f] kg → sh.cwt	(1) {2, 11}; {1} → {2} Convert masses. See pp. 133ff.
kg→sto	[g] [U→] [m:] [f] kg → stone	
kg→s.t	[g] [U→] [m:] [▲] kg → short ton	
kg→ton	[g] [U→] [m:] [▲] kg→ton	
kg→trz	[g] [U→] [m:] [f] kg → tr.oz	
K _J	[g] [CNST] K _J	(-1) {} → {2} Josephson constant in hertz per volt.
KTYP?	[g] [INFO] KTYP? ↵	(-1) {} → {1} Assumes a key code in the address specified (see KEY?), checks it, and returns the key type: <ul style="list-style-type: none">• 0 ... 9 if it corresponds to a digit 0 ... 9,• 10 if it corresponds to [, E, or +/>,• 11 if it corresponds to f or g,• 13 if it corresponds to a softkey, and• 12 if it corresponds to any other key. May help in user interaction with routines.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
L		Reserved variable for register L.
LASTx	RCL L	(-1) See Section 1 of the OM. Actually, this command will be recorded as RCL L in routines.
lbf→N	g U F&p: lbf→N	(1) {2, 11}; {1} → {2}
lb.→kg	g U m: lb.→kg	Convert forces and masses. See pp. 133ff.
LBL	g LBL labl	(0) Identifies programs and routines for execution and branching. Read more about labels and specifying them in Sect. 3 of the OM.
LBL?	g TEST g LBL? labl	(0) Tests for existence of the label specified, anywhere in program memory. See LBL for more.
LCM	g INTS f LCM	(2) {1; 10} Returns the Least Common Multiple of x and y . ¹⁵ This will always be positive.
LEAP?	g TEST f LEAP?	(0) {2, 6, 11} Assumes x containing a date in the format selected (or a real number in corresponding format), extracts the year, and tests for a leap year.
LgNrm	f PROB	(1) {2, 11}
LgNrm _e	f LgNrm: LgNrm etc.	<i>Log-normal distribution with $\mu = \ln \bar{x}_g$ specified in I and $\sigma = \ln \varepsilon$ in J. See \bar{x}_g and ε below and pp. 203ff for more.</i>
LgNrm _p		LgNrm^{-1} returns x for a given probability p in X, with μ in I and σ in J.
LgNrm ⁻¹		
LgNrm:	f PROB f LgNrm:	Submenu. See p. 120.

¹⁵ See also GCD. Remember school?

Translator's notes for French readers: LCM correspond à PPCM en français.
 Translator's notes for German readers: LCM entspricht kgV auf Deutsch..

Item	Keystrokes	Remarks (see pp. 12ff for general information)
LinF	g STAT ▲ g LinF	(0) Selects the linear fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 203ff for more.
LJ	f BITS ▲ g LJ	(-1) {10} Left justifies a bit pattern within its word size as in HP-16C: The stack will lift, placing the left-justified word in Y and the count (of bit-shifts necessary to left justify the word) in X. Example for word size 8: 1 0110 ₂ LJ returns $x = 3$ and $y = 1011\ 0000_2$.
LN	In	(1) {2, 3, 8*, 9*, 11, 12}; {1, 10} → {2} Returns the natural logarithm of x .
L_n	g X.FN Orthog L_n	(2) {2, 11}; {1} → {2} <i>Laguerre polynomials.</i> See pp. 218f for more.
LN(1+x)	g EXP f ln 1+x	(1) {2, 3, 8*, 9*, 11, 12} For $x \approx 0$, this returns a more accurate result for the fractional part than $\ln(x)$ does.
$L_{n\alpha}$	g X.FN Orthog L_{n\alpha}	(3) {2, 11}; {1} → {2} <i>Laguerre's generalized polynomials.</i> See pp. 218f for more.
$LN\beta$	g X.FN g lnβ	(2) {2, 3, 11, 12}; {1} → {2} Returns the natural logarithm of <i>Euler's Beta function</i> (see p. 90).
$LN\Gamma$	g X.FN g lnΓ	(1) {2, 3, 11, 12}; {1} → {2} Returns the natural logarithm of $\Gamma(x)$ (see p. 90). Allows also for calculating really great factorials.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
LOAD	g I/O LOAD	Restores the entire backup from <i>FM</i> and returns Backup restored . Thus, LOAD = LOADP + LOADR + LOADSS + LOADΣ. Not programmable. ¹⁶
LOADP	g I/O LOADP	(0) Loads the complete program memory from backup and appends it to the programs already in <i>RAM</i> . This will only work if there is enough space – else an error will be thrown. Not programmable. ¹⁶
LOADR	g I/O LOADR	(0) Recovers the numbered general purpose <i>registers</i> from backup. Lettered <i>registers</i> will not be recalled. ¹⁶
LOADSS	g I/O LOADSS	(0) Recovers the system state from backup. ¹⁶
LOADΣ	g I/O LOADΣ	(0) Recovers the statistical summation <i>registers</i> from backup. Throws an error if there are none. ¹⁶
LocR	g P.FN g LocR n	(0) Allocates <i>n local registers</i> (≤ 100) and 16 <i>local flags</i> for the <i>current routine</i> . See the OM, Sect. 3.
LocR?	g INFO f LocR?	$(-1) \{ \} \rightarrow \{1\}$ Returns the number of <i>local registers</i> currently allocated.
LOG ₁₀	g lg	(1) {1, 2, 3, 8*, 9*, 10, 11, 12} $\{ \{1\} \rightarrow \{2\} \}$ Returns the logarithm of <i>x</i> for base 10.
LOG ₂	g EXP lb <i>x</i>	(1) {1, 2, 3, 8*, 9*, 10, 11, 12} $\{ \{1\} \rightarrow \{2\} \}$ Returns the logarithm of <i>x</i> for base 2.

¹⁶ See SAVE on p. 73 and App. A on pp. 143f for more.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
LogF	g STAT ▲ g LogF	(0) Selects the logarithmic curve fit model. Relevant for CORR, COV, L.R., s_{xy} , \hat{x} , and \hat{y} . See pp. 203ff for more.
Logis	f PROB	
Logis _e	f Logis: Logis etc.	(1) {2, 11} <i>Logistic distribution with μ given in I and s in J.</i> See pp. 203ff for details.
Logis _p		
Logis ⁻¹		
Logis:	f PROB f Logis:	Submenu. See p. 120.
LOG _x y	g EXP log _x y	(2) {1, 2, 3, 8*, 9*, 10, 11, 12}; ({1} → {2}) Returns the logarithm of y for the base x .
LOOP	f LOOP	Menu. See p. 119.
l _{PL}	g CNST l _{PL}	(-1) {} → {2} <i>Planck length in meter.</i>
ly→m	g U→ x: ly→m	(1) {2, 11}; {1} → {2} Converts distances. See pp. 133ff.
LZ0FF	f BITS ▼ f LZ0FF	(0) Toggle leading zeros in <i>short integers</i> like flag 3 does in HP-16C. Works in bases 2, 4, 8, and 16 only.
	g MODE ▲ f ...	
LZON	f BITS ▼ LZON	
	g MODE ▲ ...	
I.INTS	f CATALOG VARS I.INTS	Submenu of <i>long integer</i> variables defined at execution time. See pp. 111ff.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
L.R.	g STAT ▲ L.R.	(-2) {} → {2} Pushes the parameters a_1 (in Y) and a_0 (in X) of the fit curve through the data points accumulated in the statistical summation <i>registers</i> on the <i>stack</i> , according to the curve fit model selected (see LINF, ORTHOF, EXPF, POWERF, LOGF). For a straight line, a_0 is its y-intercept and a_1 is its slope. See pp. 203ff for more.
$m^2 \rightarrow ac$	g [U→] f A: $m^2 \rightarrow acre$	(1) {2, 11}; {1} → {2}
$m^2 \rightarrow ac_{US}$	g [U→] f A: $m^2 \rightarrow acre_{US}$	Convert areas. See pp. 133ff.
$m^3 \rightarrow bbl$	g [U→] f V: f $m^3 \rightarrow barrel$	
$m^3 \rightarrow fz_{UK}$	g [U→] f V: f $m^3 \rightarrow fz_{UK}$	
$m^3 \rightarrow fz_{US}$	etc. g [U→] f V: $m^3 \rightarrow fz_{US}$	(1) {2, 11}; {1} → {2}
$m^3 \rightarrow gl_{UK}$	g [U→] f V: $m^3 \rightarrow gl_{UK}$	Convert volumes. See pp. 133ff.
$m^3 \rightarrow gl_{US}$	etc. g [U→] f V: $m^3 \rightarrow gl_{US}$	
MANT	f PARTS MANT	(1) {2, 11}; {1} → {2} Returns the mantissa m of the number $x = m \cdot 10^h$ displayed. Cf. EXPT.
MASKL	f BITS MASKL n	(-1) {} → {10} Work like MASKL and MASKR on HP-16C, but with the mask length (or its address) following the command instead of taken from X. Thus, the mask is pushed on the stack.
MASKR	f BITS MASKR n	Example: For WSIZE 8, MASKL 3 returns a mask word 1110 0000 ₂ . Use it e.g. for extracting the three most significant bits of an arbitrary byte via AND.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
MATRS	f [CATALOG] VARS MATRS	Submenu of matrix variables defined at execution time. See pp. 111f.
MATR?	g [TEST] ▲ MATR?	(0) Checks if x is a real or complex matrix.
MATX	f [MATX]	Menu. See p. 119.
Mat_A	Mat A	Reserved variables for solving systems of linear equations (<i>SLE</i> , see Section 2 of the OM).
Mat_B	Mat B	
Mat_X	Mat X	(-1) Returns the solution vector of an <i>SLE</i> .
max	g [X.FN] g max	(2) {1, 2, 4, 5, 6, 10, 11} Returns the maximum of x and y .
m_e	g [CNST] m_e	(-1) {} → {2} Electron mass in <i>kilogram</i> .
MEM?	g [INFO] MEM?	(-1) {} → {1} Returns the number of free <i>words</i> in program memory (1 <i>word</i> = 2 <i>bytes</i>), also taking into account the local <i>registers</i> allocated.
MENU	g [P.FN] P.FN2 MENU	Displays the programmable menu. See the HP-42S OM, Part 2, Section 10, p. 146.
MENUS	f [CATALOG] MENUS	Submenu of menus defined at execution time. See pp. 111ff.
min	g [X.FN] g min	(2) {1, 2, 4, 5, 6, 10, 11} Returns the minimum of x and y .
MIRROR	f [BITS] f MIRROR	(1) {10} Reflects the bit pattern in x (e.g. 0001 0111 ₂ would become 1110 1000 ₂ for word size 8).

Item	Keystrokes	Remarks (see pp. 12ff for general information)
mi. \rightarrow m	g U x: f mi. \rightarrow m	(1) {2, 11}; {1} \rightarrow {2} Converts distances. See pp. 133ff.
M _M	g CNST M _M etc.	(-1) {} \rightarrow {2} Masses of the Moon and the neutron in <i>kilogram</i> ; neutron to proton mass ratio.
m _n		
m _n /m _p		
MOD	g INTS f MOD	(2) {1, 2, 3, 10, 11, 12} Returns $y \bmod x$ (see Section 2 of the OM for examples). Cf. RMDR.
MODE	g MODE	Menu. See p. 120.
MONTH	g CLK f MONTH	(1) {2, 6, 11} \rightarrow {1} Assumes x containing a <i>date</i> in the format selected (or a real number in corresponding format) and extracts the month.
m _p	g CNST m _p etc.	(-1) {} \rightarrow {2} Proton mass and <i>Planck</i> mass in <i>kilogram</i> ; proton to electron mass ratio.
M _{PL}		
m _p /m _e		
MSG	g P.FN P.FN2 f MSG	(0) {1, 2, 11} Throws the (<i>temporary</i>) error message specified by the integer part of x . Cf. ERR. See App. C on pp. 164ff for the respective error codes.
m _u	g CNST m _u	(-1) {} \rightarrow {2}
m _u c ²	g CNST m _u c ²	Atomic mass constant in <i>kilogram</i> and its energy equivalent in <i>joule</i> .
MULTx	g DISP ▲ MULTx	(0) Select the symbol for multiplication display.
MULT-	g DISP ▲ MULT-	

Item	Keystrokes	Remarks (see pp. 12ff for general information)
MUL π	g MODE MULπ	(0) Sets the ADM to <i>multiples of π.</i>
MUL $\pi\rightarrow$	f L\rightarrow f $\cancel{\pi \rightarrow \pi}$	(1) {1, 2, 11} \rightarrow {4} Converts angles as described on pp. 140f.
MVAR	g P.FN P.FN2 f MVAR name	(0) Defines a <i>menu variable</i> . Such variables are required for VARMNU. Works in PEM only.
MyMenu	f CATALOG MENUS MyMenu	User menu. See the OM, Section 6.
My α	f CATALOG CHARS Myα	User menu in AIM.
m_μ	g CNST m_μ	(-1) {} \rightarrow {2} Muon mass in <i>kilogram</i> .
M.DELR	f DELR while editing a matrix	(0) {8, 9} Deletes the current row of elements (where the cursor is in). Will not work if the matrix has only one row.
M.DIM	f MATX f DIM name	(0) {1, 2} Creates a new named matrix or re-dimensions an existing matrix to IP(y) rows and IP(x) columns. See DIM in the HP-42S Owner's Manual, p. 217.
M.DIM?	g INFO g DIM? f MATX ▲ f DIM?	{8, 9} \rightarrow {1} Returns the dimensions of the matrix x (rows to Y, columns to X). Note the matrix is saved in L. Previous y goes into Z, previous z into T, etc.
M.DY	g CLK ▲ M.DY	(0) Selects the format mm/dd/yyyy for <i>date</i> display.
M.EDI	f MATX EDIT	(2) {8, 9} Opens x using the <i>Matrix Editor</i> (like MATRIX EDIT in HP-42S). See Section 2 of the OM.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
M.EDIN	f MATX f EDITN <i>name</i>	(2) Opens a named matrix using the <i>Matrix Editor</i> (like MATRIX EDITN in <i>HP-42S</i>). See <i>Section 2</i> of the OM.
M.EDIT		Submenu for matrix editing, called by M.EDI or M.EDIN. See p. 119.
M.GET	f MATX g GETM	(0) {1, 2} → {8, 9} Gets a sub-matrix with IP(y) rows and IP(x) columns out of the indexed matrix into X (like GETM in <i>HP-42S</i>). Cf. M.PUT.
M.GOTO	f GOTO with M.EDIT displayed	(0) Asks for target row and column and moves to this matrix element.
M.GROW	f GROW with M.EDIT displayed	(0) Allows the indexed matrix to grow automatically (see J+ above and <i>Section 2</i> of the OM; see also GROW in the <i>HP-42S Owner's Manual</i> , p. 213.). Cf. M.WRAP.
M.INSR	f INSR with M.EDIT displayed	(0) {8, 9} Inserts a new row of elements containing zero, left of the current cursor position in the matrix.
M.LU	f MATX ▲ f M.LU	(1) Takes a <i>descriptor</i> of a square matrix in X. Transforms (X) into its LU decomposition in-situ. The value in X is replaced by a <i>descriptor</i> that defines the pivots that were required to calculate the decomposition. The most significant digit is the pivot for the first diagonal entry, the next most significant for the second and so forth.
M.NEW	f MATX NEW	(2) {1, 2} → {8} Creates a new matrix (like NEW in <i>HP-42S</i>). Its number of rows shall be supplied in Y and its number of columns in X. M.NEW returns a clear matrix in X – all its elements are set to zero.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
M.OLD	OLD with M.EDIT displayed	(0) Recalls the old element content (like OLD in HP-42S). See Section 2 of the OM.
M.PUT	f [MATX] g PUTM	(0) {8, 9} Puts the matrix x as is into the indexed matrix (like PUTM in HP-42S). Cf. M.GET.
M.R \leftrightarrow R	f [MATX] ▲ f R \leftrightarrow R	(0) {8, 9} Swaps row x and row y of the indexed matrix (like R \leftrightarrow R in HP-42S).
M.SIMQ		Submenu of MATX, called by SIM_EQ.
M.SQR?	g TEST ▲ f M.SQR?	(0) Returns true if x is a square matrix.
M.WRAP	f WRAP with M.EDIT displayed	(0) Controls the index pointers (see Section 2 of the OM). Cf. M.GROW.
m:	g U \leftrightarrow m:	Submenu. See p. 133.
m \rightarrow au	g U \leftrightarrow x: m \rightarrow au	(1) {2, 11}; {1} \rightarrow {2} Convert distances or heights. See pp. 133ff.
m \rightarrow fm.	g U \leftrightarrow x: ▲ m \rightarrow fathom	
m \rightarrow ft.	g U \leftrightarrow x: f m \rightarrow ft.	
m \rightarrow ft _{us}	g U \leftrightarrow x: ▲ m \rightarrow survey foot _{us}	
m \rightarrow in.	g U \leftrightarrow x: g m \rightarrow in.	
m \rightarrow ly	g U \leftrightarrow x: m \rightarrow ly	
m \rightarrow mi.	g U \leftrightarrow x: f m \rightarrow mi.	
m \rightarrow nmi.	g U \leftrightarrow x: f m \rightarrow nmi.	
m \rightarrow pc	g U \leftrightarrow x: m \rightarrow pc	

Item	Keystrokes	Remarks (see pp. 12ff for general information)
m→pt.	[g] [U→] x: [g] m → point	
m→yd.	[g] [U→] x: [g] m→yd.	
m _⊕	[g] [CNST] m _⊕ etc.	(-1) {} → {2} Masses of the Sun and Earth in <i>kilogram</i> .
N _A	[g] [CNST] N _A	(-1) {} → {2} Avogadro's number in <i>particles per mol</i> .
NaN	[g] [CNST] NaN	(-1) Not a Number.
NAND	[f] [BITS] [f] NAND	(2) Works in analogy to AND. See p. 18.
NaN?	[g] [TEST] [▲] [f] NaN?	(0) Returns true if x is Not a Number.
NBin	[f] [PROB]	(1) {2, 11}
NBin _e	[g] NBin: NBin etc.	Negative binomial distribution with the total number of failures f in X, the probability of a success p₀ in I, and the number of draws n in J. See pp. 203ff for more information.
NBin _p		
NBin ⁻¹		
NBin:	[f] [PROB] [g] NBin:	_submenu_. See p. 120.
NEIGHB	[g] [INFO] [f] NEIGHB	(2) {1} Returns ... <ul style="list-style-type: none"> • $x + 1$ for $x < y$; • x for $x = y$; • $x - 1$ for $x > y$. (2) {2, 3, 11, 12} Returns the nearest machine-representable number to x in the direction towards y in the mode set. For

Item	Keystrokes	Remarks (see pp. 12ff for general information)
		<ul style="list-style-type: none"> ... $x < y$, it is the machine successor of x ; ... $x = y$, it is y ; ... $x > y$, it is the machine predecessor of x. <p>NEIGHB may be useful investigating numeric stability (see NEIGHBOR in the <i>HP-71 Math Pac</i>).</p>
NEXTP	g X.FN g NEXTP	(1) {1, 2, 11} → {1} Returns the next prime number greater than x .
nmi.→m	g U→ x: f nmi.→m	(1) {2, 11}; {1} → {2} Converts distances. See pp. 133ff.
NOP	g P.FN P.FN2 f NOP	(0) 'Empty' program step (for historical reasons only).
NOR	f BITS f NOR	(2) Works in analogy to AND. See p. 18.
Norml	f PROB	(1) {2, 11}
Norml _e	f Norml: Norml etc.	Normal distribution with an arbitrary mean μ given in I and a standard deviation σ in J. See Sect. 2 of the OM for an application example and pp. 203ff for more.
Norml _p		
Norml ⁻¹		Norml ⁻¹ returns x for a given probability p in X, with μ in I and σ in J.
Norml:	f PROB f Norml:	_submenu_. See p. 120.
NOT	f BITS NOT	<p>(1) {10}</p> <p>Inverts x bit-wise as on HP-16C.</p> <p>(1) {1, 2, 11} → {1}</p> <p>Returns 1 for $x = 0$, and 0 for $x \neq 0$.</p>
NPER	g FIN TVM n_{PER}	(1) Reserved variable for the total number of payment periods for your loan or total number of compounding periods for your investment in TVM.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
$n\Sigma$	g STAT ▼ n	(-1) {} → {1} Recalls the number of accumulated data points.
$N \rightarrow lbf$	g U→ F&p: N → lbf	(1) {2, 11}; {1} → {2} Converts forces. See pp. 133ff.
ODD?	g TEST f ODD?	(0) Checks if x is integer and odd. xxx
OFF	g OFF	(0) In PEM, inserts a step to turn your WP 43S off under program control. Else turns your WP 43S off.
OR	f BITS OR	(2) Works in analogy to AND. See p. 18.
OrthoF	g STAT ▲ f OrthoF	(0) Selects the linear orthogonal fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 212ff for more.
ORTHOG	g X.FN Orthog	Submenu. See p. 123.
$oz \rightarrow kg$	g U→ m: oz → kg	(1) {2, 11}; {1} → {2} Converts masses. See pp. 133ff.
P_0	g CNST P₀	(-1) {} → {2} Standard atmospheric pressure in <i>pascal</i> .
PAUSE	g P.FN PAUSE n	(0) With a routine running, refreshes the display and pauses program execution for n ticks (see TICKS), with $0 \leq n \leq 99$. The pause will terminate early when you press a key.
$Pa \rightarrow atm$	g U→ F&p: f Pa → atm	
$Pa \rightarrow bar$	g U→ F&p: Pa → bar	(1) {2, 11}; {1} → {2}
$Pa \rightarrow iHg$	g U→ F&p: f Pa → in.Hg	Convert pressures. See pp. 133ff.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
Pa→psi	g U→ F&p: Pa→psi	
Pa→tor	g U→ F&p: f Pa → torr	
PARTS	f PARTS	Menu. See p. 120.
pc→m	g U→ x: pc→m	(1) {2, 11}; {1} → {2} Converts distances. See pp. 133ff.
PERM	f PROB P_{yx}	(2) {1} Returns the number of possible <u>arrangements</u> (a.k.a. permutations) of x items taken out of a set of y items. No item occurs more than once in an arrangement, and <u>different orders</u> of the same x items are counted separately. Cf. COMB. (2) {2, 3, 11, 12} See pp. 201ff for the formula.
PER/a	g FIN TVM f per/a	(1) Reserved variable for the <u>annual</u> number of payments for your loan or compounding periods of your investment in <i>TVM</i> .
PGMINT	f ADV f PGMINT labl	Specifies the address of the expression to be integrated or solved, respectively.
PGMSLV	f ADV f PGMSLV labl	See Section 4 of the OM.
PIXEL	g P.FN P.FN2 g PIXEL	(0) Turns on a single pixel (dot) on the screen. The location of the pixel is given by the numbers in the X- and Y-registers. See AGRAPH on p. 17 for more.
PLOT	g STAT g PLOT	(0) See p. 241 for more.
PMT	g FIN TVM PMT	(1) Reserved variable for the payment per period for your investment or loan in <i>TVM</i> .

Item	Keystrokes	Remarks (see pp. 12ff for general information)
P_n	g X.FN Orthog P_n	(1) {2, 11}; {1} → {2} <i>Legendre polynomials.</i> See pp. 218f for more.
POINT	g P.FN P.FN2 g POINT	(1) {1, 2, 11} Turns on a square point (3×3 px □) on the screen. The location of its center is given by integer parts of the numbers in the X - and Y-registers . See AGRAPH on p. 17 for more.
Poiss	f PROB	(1) {2, 11}; {1} → {2}
Poiss _e	g Poiss: Poiss etc.	<i>Poisson distribution</i> with the number of successes g in X and the <i>Poisson parameter</i> λ in I . See pp. 203ff for details.
Poiss ⁻¹	f PROB g Poiss: Poiss ⁻¹	(1) {2, 11} Returns the maximum number of successes m for a given probability p in X and λ in I .
Poiss:	f PROB g Poiss:	Submenu. See p. 120.
POLAR	g MODE POLAR g CPX g POLAR	(0) Sets polar format for displaying complex numbers.
PopLR	g P.FN g PopLR	(0) Pops the local registers allocated to the <i>current routine</i> (see <i>Section 3</i> of the <i>OM</i>) without returning to the calling routine. See LOCR and RTN.
PowerF	g STAT ▲ g PowerF	(0) Selects the power curve fit model. Relevant for CORR, COV, L.R., s_{xy} , \hat{x} , and \hat{y} (see pp. 203ff for more).
pr→dB	g U→ ▲ power ratio → dB	(1) {2, 11}; {1} → {2} Converts ratios. See pp. 133ff.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
PRCL	g P.FN f PRCL	(0) Copies the <i>current program</i> (from <i>FM</i> or <i>RAM</i>) and appends it to <i>RAM</i> , where it can then be edited (see the OM). PRCL allows for duplicating programs in <i>RAM</i> . Will only work with enough space at destination. Recall a library routine from <i>FM</i> , edit it, and PSTO – this way you can modify this part of the <i>FM</i> library (see PSTO).
PRIME?	g TEST f PRIME?	(0) {1, 2, 11} Checks if the absolute value of $ P(x) $ is a prime. The method is believed to work for integers up to 9×10^{18} .
PROB	f PROB	Menu. See p. 120.
PROFRC	f a b/c	(1) {2, 11} Allows only <i>proper fractions</i> in display. Displays any reals (with $ x < 10^6$) according to the settings by DEN... as <i>proper fractions</i> , e.g. 1.25 or $\frac{5}{4}$ as $1\frac{1}{4}$. Cf. IMPFRC.
PROGS	f CATALOG PROGS	Submenu of global labels defined at execution time. See pp. 111f.
psi→Pa	g U→ F&p: psi→Pa	(1) {2, 11}; {1} → {2} Converts pressures. See pp. 133ff.
PSTO	g P.FN f PSTO	(0) Copies the <i>current program</i> (see the OM) from <i>RAM</i> and appends it to the <i>FM</i> library. Not programmable. Cf. PRCL. This program must include at least one LBL statement with a global label (preferably at its beginning). If a program with the same label already exists in the library it will be deleted first. Global labels may be browsed in CATALOG PROGS and called by XEQ.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
pt. \rightarrow m	g U\rightarrow x: g point \rightarrow m	(1) {2, 11}; {1} \rightarrow {2} Converts heights. See pp. 133ff.
PUTK	g P.FN f PUTK r	(0) Assumes a key code in the address specified. Stops program execution, takes said code and puts it in the keyboard buffer resulting in immediate execution of the corresponding call. R/S is required to resume program execution then. May help in user interaction with routines.
PV	g FIN TVM PV	(1) Reserved variable for the present value of your investment or loan in <i>TVM</i> .
P.FN	g P.FN	Menu . See p. 121.
P.FN2	g P.FN P.FN2	Submenu . See p. 121.
P:	g U\rightarrow P:	Submenu . See p. 133.
QUIET	g I/O f QUIET g MODE f QUIET	(0) Toggles the <i>flag</i> to disable or enable the beeper (not programmable).
R	g CNST R	(-1) {} \rightarrow {2} Molar gas constant in <i>joule per mol and kelvin</i> .
RAD	g MODE RAD	(0) Sets the <i>ADM</i> to <i>radians</i> .
RAD \rightarrow	f L\rightarrow f 4 ^r \rightarrow 4	(1) {1, 2, 11} \rightarrow {4} Converts angles as described on pp. 140f.
RAM	f CATALOG PROGS RAM	Submenu of global labels defined at execution time. See pp. 111f.
RAN#	f PROB ▲ RAN#	(-1) {} \rightarrow {2} Returns a random number between 0 and 1 like RAN does in <i>HP-42S</i> . See also SEED.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
RBR	[g] RBR	Calls the <i>register browser</i> . See Sect. 5 of the OM. You may call RBR also in PEM but it is not programmable.
RCL	[RCL] <i>r</i>	(-1) Recalls the content of a <i>register</i> or variable.
RCLCFG	[RCL] [f] Config <i>r</i>	(0) Recalls a configuration stored by STOCFG (see Section 2 of the OM).
RCLEL	[f] [MATX] [g] RCLEL [RCL] [g] ...EL	(-1) {} → {2, 3} Recalls a copy of the current element a_{ij} of the indexed matrix. Cf. STOEL.
RCLIJ	[f] [MATX] ▲ RCLIJ [RCL] [g] ...IJ	(-2) {} → {1} Recalls the current values of the matrix index pointers into X (= column number) and Y (= row number). If the pointers both equal zero, then there is currently no indexed matrix. Cf. STOIJ.
RCLS	[RCL] [f] Stack <i>r</i>	Recalls 4 or 8 values from a set of <i>registers</i> starting at address <i>r</i> , and pushes them on the <i>stack</i> . This is the converse command of STOS.
RCL+	[RCL] [+] <i>r</i>	(1) Recalls a content of a <i>register</i> or variable, executes the operation specified, and puts the result on the <i>stack</i> like a <i>monadic function</i> . ¹⁷
RCL-	[RCL] [-] <i>r</i>	
RCL×	[RCL] [x] <i>r</i>	
RCL/	[RCL] [/] <i>r</i>	
RCL↑	[RCL] [f] Max <i>r</i> [RCL] ▲ <i>r</i>	(1) {1, 2, 4, 5, 6, 10, 11} Replaces <i>x</i> with the maximum of <i>r</i> and <i>x</i> . ¹⁷

¹⁷ Only legal operations according to the matrices in Section 2 of the OM will work. See also the examples given there.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
RCL _v	[RCL] f Min r [RCL] ▼ r	(0) {1, 2, 4, 5, 6, 10, 11} Replaces x with the minimum of r and x . ¹⁷
RDP	g DISP f RDP n	(1) {2, 3, 5, 8*, 9*, 11, 12} Rounds x to n decimal places ($0 \leq n \leq 99$, think of FIX format), taking the RM setting into account. See RM and compare RSD.
RDX, RDX.	g DISP ▲ RDX, etc.	(0) Select a comma or a point as decimal radix mark.
r _e	g CNST r _e	(-1) {} → {2} Classical electron radius in meter.
Re	g CPX f Re f PARTS g Re	(1) [3] → {2}; [9] → {8}; [12] → {11} Returns the real part of x . Cf. IM.
REALRES	g MODE f REALRE g FLAGS CF	(0) Allows only real results, no complex ones. The letter S cannot be shown in the menu view for space reasons there. Cf. CPXRES.
REALS	f CATALOG VARS REALS	Submenu of real variables defined at execution time. See pp. 111f.
REAL?	g TEST ▲ REAL?	(0) Checks if x is a real number or matrix.
RECT	g MODE RECT g CPX g ...	(0) Sets rectangular (Cartesian) format for displaying complex numbers.
REGS		Reserved variable for the 100×1 matrix of registers – if required.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
RECV		(0) Prepares your WP 43S for receiving data via serial I/O. See SEND and Section 3 in the OM for more.
RESET	 	Executes CLALL and resets all modes to <i>startup default</i> , i.e. 24h, 2COMPL, ALL 0, CPXi, DEG, DENANY, DENMAX 0, DSTACK 4, GAP 3, J/G 1752-01-01, LinF, LocR 0, LZOFF, MULTx, PROFRC, RDX., REALRES, RECT, RM 0, SCIOVR, SSIZE4, TDISP -1, WSIZE 64, and Y.MD. See these individual commands for more. RESET is not programmable.
RE \rightarrow CX		<p>(2) {2} \rightarrow {3}; {11} \rightarrow {12}</p> <p>Composes a complex number out of two reals or integers x and y, taking either</p> <ul style="list-style-type: none"> • (for \sqcup) the real part from X and the imaginary part from Y, or • (for \odot) the magnitude from X and the angle from Y. <p>Sets C unless set before.</p> <p>(2) {8} \rightarrow {9}</p> <p>Works in analogy for two real matrices x and y.</p>
Re \leftrightarrow Im		(1) {3, 9*, 12} Swaps real and imaginary parts of complex objects.
RJ	 	<p>(-1) {10}</p> <p>Right justifies a bit pattern within its word size, in analogy to LJ (see there).</p> <p>Example: 10 1100₂ RJ results in y = 1011₂ and x = 2.</p>

Item	Keystrokes	Remarks (see pp. 12ff for general information)
R_K	g [CNST] R_K	(-1) {} → {2} Von Klitzing constant in <i>ohm</i> .
RL	f [BITS] ▲ RL n etc.	(1) {10}
RLC		Work like n consecutive RLs / RLCs on <i>HP-16C</i> , similar to RL _n / RLC _n there. For RL, $0 \leq n \leq 63$. For RLC, $0 \leq n \leq 64$. RL 0 / RLC 0 execute as NOP, but load L. See the OM, Sect. 2, for more.
R_M	g [CNST] R_M	(-1) {} → {2} Mean radius of the Moon in <i>meter</i> .
RM	g [MODE] f RM n	(0) Sets floating point rounding mode. This rounding mode is used only for RSD or when converting from the high precision internal format (39 digits) to packed reals. It will <u>not</u> alter the display nor change the behavior of ROUND. The following 7 modes are supported: 0: round half even: $\frac{1}{2} = 0.5$ rounds to next even number (default). ¹⁸ 1: round half up: 0.5 rounds up ('business-man's rounding'). ¹⁹ 2: round half down: 0.5 rounds down. 3: round up: rounds away from 0. 4: round down: rounds towards 0 (truncates). 5: ceiling: rounds towards $+\infty$. 6: floor: rounds towards $-\infty$.
RM?	g [INFO] RM?	(-1) {} → {1} Returns the floating point rounding mode set. See RM for more.

¹⁸ This is the way of rounding used in science.

¹⁹ Translator's notes for French and German readers: Cela correspond à l'arrondi commercial. / Das entspricht kaufmännischer Rundung.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
RMDR	g RMDR	(2) {1, 2, 3, 10, 11, 12}
	g INTS f RMDR	Returns the remainder of a division. Equals RMD on HP-16C but works for reals as well. See Section 2 of the OM for examples. Cf. MOD.
RNORM	f MATX ▲ f RNORM	(1) {8, 9} Calculates the row norm of the matrix in X, i.e. the maximum value (over all rows) of the sums of the absolute values of all elements in a row (like RNRM on HP-42S). For a vector, the row norm is the largest absolute value of any of its elements.
ROUND	g DISP ROUND	(1) {2, 3, 4, 5, 6, 8*, 9*, 11, 12}
	f PARTS f ...	Rounds x using the current display format like RND on HP-42S.
ROUNDI	g DISP ROUNDI	(1) {8}; {2, 11} → {1}
	f PARTS f ...	Rounds x to next integer. ½ rounds to 1.
RR	f BITS ▲ RR n etc.	(1) {10} Work like n consecutive RRs / RRCs on HP-16C, similar to RRn / RRCn there. For RR, $0 \leq n \leq 63$. For RRC, $0 \leq n \leq 64$. RR 0 / RRC 0 execute as NOP, but load L. See the OM, Sect. 2, for more.
RRC		
RSD	g DISP f RSD n	(1) {2, 3, 5, 8*, 9*, 11, 12} Rounds x to n significant digits ($1 \leq n \leq 34$), taking the RM setting into account. See RM, cf. RDP.
RSUM	f MATX ▲ f RSUM	(1) {8, 9} Calculates the row sum of the matrix in X, returning an $m \times 1$ matrix filled with the row sums of the $m \times n$ input matrix.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
RTN	g RTN	(0) In <i>PEM</i> : RTN is the logically last command in a routine (see <i>Section 3</i> of the OM). In a routine executing: RTN pops local data (cf. PopLR) and returns to the caller, i.e. moves the program pointer one step behind the XEQ instruction that called said routine. If there is none (i.e. this routine is top level), program execution halts, the program pointer is set to step 0000, and ¶ is lit. If RTN is pressed in <i>run mode</i> with no routine executing, it resets the program pointer to the start of <i>current program</i> (see <i>Section 3</i> of the OM). If the program is in <i>FM</i> , the pointer is set to step 0000 in <i>RAM</i> , and ¶ is lit.
RTN+1	g P.FN P.FN2 RTN+1	(0) Works like RTN, but moves the program pointer <u>two</u> steps behind the XEQ instruction that called said routine.
R-CLR	g P.FN g R-CLR	(0) {2, 11} Interprets <i>x</i> in the form <i>sss.nn</i> . Clears <i>nn</i> registers starting with address <i>sss</i> . Example: For <i>x</i> = 34.567 , R-CLR will clear R34 through R89. ATTENTION: For <i>nn</i> = 0 , clearing will cover the maximum available: <ul style="list-style-type: none">• For <i>sss</i> ∈ [0; 99], it will stop at R99.• For <i>sss</i> ∈ [100; 111] , it will stop at K.• For <i>sss</i> ≥ 112, it will stop at the highest allocated local register.
R-COPY	g P.FN g R-COPY	(0) {2, 11} Interprets <i>x</i> in the form <i>sss.nnddd</i> . Takes <i>nn</i> registers starting with address <i>sss</i> and copies their contents to <i>ddd</i> etc.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
		<p>Example: For $x = 7.0304567$, $r07$, $r08$, $r09$ will be copied into R45, R46, R47, respectively.</p> <p>For $x < 0$, R-COPY will take nn registers from <i>FM</i> instead, starting with <i>register number sss </i>. Destination will be in <i>RAM</i> always.</p> <p>ATTENTION: For nn = 0, copying will cover the maximum available as explained with R-CLR. Then x must be negative.</p>
R-SORT		<p>(0) {2, 11}</p> <p>Interprets x in the form sss.nn. Sorts the contents of nn registers starting with address sss.</p> <p>Example: Assume $x = 49.036\ 9$, $r49 = 1.2$, $r50 = -3.4$, and $r51 = 0$; then R-SORT will return $r49 = -3.4$, $r50 = 0$, and $r51 = 1.2$.</p> <p>ATTENTION: For nn = 0, sorting will cover the maximum available as explained with R-CLR.</p>
R-SWAP		<p>(0) {2, 11}</p> <p>Works like R-COPY but <u>swaps</u> the contents of source and destination <i>registers</i>.</p>
R→D		<p>(1) {1, 2, 11} → {4}</p> <p>Converts angles as described on pp. 140f.</p>
R↑		<p>Rotates the <i>stack</i> contents one level up or down, respectively. See <i>Section 1</i> of the OM for details.</p>
R↓		
R _∞		<p>(-1) {} → {2}</p>
R _⊖		<p>Rydberg constant (see p. 130); mean radii of the Sun and Earth in <i>meter</i>.</p>
R _⊕		

Item	Keystrokes	Remarks (see pp. 12ff for general information)
s	g STAT s	(-2) {} → {2} Takes the statistical sums accumulated, calculates the <i>sample standard deviations</i> s_y and s_x and pushes them on the stack. See Sect. 2 of the OM for the output format and pp. 203ff for the formula.
Sa	g CNST Sa	(-1) {} → {2} Semi-major axis in <i>meter</i> of the Earth model WGS84. ²⁰
SAVE	f SAVE	(0) Saves user program space, <i>registers</i> and system state to FM, and returns Saved. Recall your backup using the different flavors of LOAD. Not programmable. See App. A for more.
SB	f BITS g SB n	(1) {10} Sets the specified bit in x .
Sb	g CNST Sb	(-1) {} → {2} Semi-minor axis in <i>meter</i> of WGS84. ²⁰
SCI	g DISP SCI n	(0) Sets scientific display format (see Section 2 of the OM).
scw→kg	g U m f short cwt → kg	(1) {2, 11}; {1} → {2} Converts masses. See pp. 133ff.
SCIOVR	g DISP ▲ SCIOVR	(0) Defines that numbers exceeding the range displayable in ALL or FIX will be shown in scientific format. Cf. ENGOVR, see RESET.
SDIGS?	g INFO f SDIGS?	(-1) {} → {1} Returns the number of significant digits set by SETSIG.

²⁰ This model is used to define the Earth's surface for surveying and GPS. See http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html

Item	Keystrokes	Remarks (see pp. 12ff for general information)
SDL	g P.FN P.FN2 f SDL <i>n</i> etc.	(1) {2, 11} Shifts digits left (right) by <i>n</i> decimal positions, equivalent to multiplying (dividing) <i>x</i> by 10^n . Cf. SL and SR for binary integers.
Se ²	g CNST Se²	(-1) {} → {2} First eccentricity squared of the Earth model WGS84 (see footnote 20 on previous page).
SEED	f PROB ▲ SEED	(0) {2, 11} Stores a seed for random number generation. If <i>x</i> = 0, the seed is taken from the real-time clock.
SEND	g I/O f SEND	(0) Sends all RAM data to the device connected via serial I/O. See RECV and Section 3 in the OM for more.
SETCHN	g DISP ▲ g CHINA	(0) Sets regional preferences (see Section 2 of the OM).
SETDAT	g CLK ▲ SETDAT	(0) Sets the date for the real-time clock (the emulator takes this information from the PC clock).
SETEUR	g DISP ▲ g EUROPE	(0) Set some regional preferences (see Section 2 of the OM).
SETIND	g DISP ▲ g INDIA	
SETJPN	g DISP ▲ g JAPAN	
SETSIG	g MODE ▲ f SETSIG	(0) {1} Sets the number of significant digits (0 ... 34) for rounding after each operation. SETSIG 0 sets maximum precision.
SETTIM	g CLK ▲ SETTIM	(0) Sets the time for the real-time clock (the emulator takes this information from the PC clock).

Item	Keystrokes	Remarks (see pp. 12ff for general information)
SETUK	g DISP ▲ g UK etc.	(0) Set some regional preferences (see Section 2 of the OM).
SETUSA		
Se ^{'2}	g CNST Se' ²	(-1) {} → {2} Second eccentricity squared of the Earth model WGS84 (see footnote 20 on p. 47).
SF	g FLAGS SF n	(0) Sets the flag specified.
Sf ⁻¹	g CNST Sf ⁻¹	(-1) {} → {2} Flattening parameter of the Earth model WGS84 (see footnote 20 on p. 47).
SHOW	g SHOW	(0) {2, 11, 12}
	g X.FN ▲ SHOW	Shows all digits stored in a number.
SIGN	f PARTS sign	(1) {8}, {1, 2, 10, 11} → {1} Returns 1 for $x > 0$, -1 for $x < 0$, and 0 for $x = 0$ or non-numeric data. Corresponds to the mathematical function signum(x).
	g CPX f sign	(1) {3, 12} Returns the unit vector of the complex number x (cf. UNITV). Maintained for backward compatibility only.
SIGNMT	g BITS ▼ SIGNMT	(0) Sets sign-and-mantissa mode for operations on <i>short integers</i> . See Section 2 of the OM.
	g MODE ▲ ...	
SIM_EQ	f MATX SIM EQ n	(0) Solves a system of n linear equations $(\text{MATA}) \cdot \overrightarrow{\text{MATX}} = \overrightarrow{\text{MATB}}$. If these matrices are not defined before, they will be created automatically at execution time. See Sect. 2 of the OM for more.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
sin	TRI sin	(1) {2, 3, 8*, 9*, 11, 12}; {1, 4} → {2} Returns the sine of the angle in X.
sinc	g X.FN ▲ sinc	(1) {2, 3, 8*, 9*, 11, 12} Returns $\frac{\sin(x)}{x}$ for $x \neq 0$ and 1 for $x = 0$. Note input has to be supplied in radians.
sinh	g EXP g sinh	(1) {2, 3, 8*, 9*, 11, 12} Returns the hyperbolic sine of x.
SKIP	g P.FN P.FN2 g SKIP n	(0) Skips n program steps forwards ($0 \leq n \leq 255$). So e.g. SKIP 2 skips over the next two steps, going e.g. from step 123 to step 126. If SKIP attempts to cross an END, an error is thrown. ATTENTION: If you edit a section of your routine crossed by one or more BACK, SKIP, or CASE jumps, this may well result in a need to manually maintain all those statements individually .
SL	f BITS ▲ SL n	(1) {10} Works like n (≤ 63) consecutive SLs on HP-16C. SL 0 executes as NOP, but loads L. See Sect. 2 of the OM for more.
SLOW	g MODE f SLOW	(0) Sets the processor speed to 'slow', about 50% of 'fast'. This is also automatically set for low battery voltage (s. the OM, Sect. 2). Cf. FAST.
SLVQ	f ADV SLVQ	{1, 2, 3} → {2 or 3} Solves the quadratic equation $ax^2 + bx + c = 0$ with its parameters on the input stack [c, b, a, ...], and tests the result. <ul style="list-style-type: none">• If $r := b^2 - 4ac \geq 0$, SLVQ returns $-\frac{b \pm \sqrt{r}}{2a}$ in Y and X. In a routine, the step after SLVQ will be executed.

Item	Keystrokes	Remarks (see pp. 12ff for general information)												
		<ul style="list-style-type: none"> Else, SLVQ returns the first complex root in X and the second in Y (the complex conjugate of the first). In a routine, the step after SLVQ will be skipped. <p>In either case, SLVQ returns r in Z. Higher stack registers are kept unchanged. L will contain equation parameter c.</p>												
s_m	g STAT s_m	$(-2) \{ \} \rightarrow \{2\}$ Takes the statistical data accumulated and pushes the <i>standard errors</i> (i.e. <i>std. deviations</i> of the means \bar{y} and \bar{x}) on the stack. Output format will be like the one of s (see the OM, Sect. 2).												
SMODE?	g INFO SMODE?	$(-1) \{ \} \rightarrow \{1\}$ Returns the <i>integer sign mode</i> set for <i>short integers</i> , i.e. <table> <tr> <td>true</td> <td>2</td> <td>for 2's complement,</td> </tr> <tr> <td>true</td> <td>1</td> <td>for 1's complement,</td> </tr> <tr> <td>false</td> <td>0</td> <td>for unsigned, or</td> </tr> <tr> <td>true</td> <td>-1</td> <td>for sign & mantissa mode.</td> </tr> </table>	true	2	for 2's complement,	true	1	for 1's complement,	false	0	for unsigned, or	true	-1	for sign & mantissa mode.
true	2	for 2's complement,												
true	1	for 1's complement,												
false	0	for unsigned, or												
true	-1	for sign & mantissa mode.												
s_{mw}	g STAT f s_{mw}	$(-1) \{ \} \rightarrow \{2\}$ Returns the <i>standard error</i> for weighted data, i.e. the <i>standard deviation</i> of the mean \bar{x}_w .												
SOLVE	f ADV SOLVE var	$\{2, 3\}$ Solves the equation $f(var) = 0$, with f calculated by the equation specified (in <i>PEM</i> by PGMSLV). Two initial estimates of the root must be supplied in X and Y when calling SOLVE. It returns var_{root} in X , the second last var -value tested in Y , then $f(var_{root})$ in Z , and 0 in T . Additionally, SOLVE acts as test in programs, so the next program step will be skipped if SOLVE fails to find a root. See												

Item	Keystrokes	Remarks (see pp. 12ff for general information)
		<i>Section 4 of the OM</i> for more. ATTENTION: SOLVE fills all stack registers with x before calling the routine specified.
Solver	g EQN Solver	Submenu for solving a given equation. See the OM, Sect. 4, for more.
SPEC?	g TEST ▲ f SPEC?	(0) True if x is ‘special’ ($\pm\infty$ or NaN).
SR	f BITS ▲ SR n	(1) {10} Works like n (≤ 63) consecutive SRs on HP-16C. SR 0 executes as NOP, but loads L. See Section 2 of the OM for more.
SSIZE4	g STK g SSIZE4 etc. or g MODE g SSIZE4	Set the stack size to 4 or 8 registers, respectively (see Section 1 of the OM). Note register contents will remain unchanged in this operation (as well as if stack size is modified by any other operation – e.g. by RCLCFG).
SSIZE8	g INFO SSIZE?	(-1) {} → {1} Returns the number of stack registers currently allocated, 4 or 8.
STAT	g STAT	Menu. See p. 121.
STATUS	g FLAGS STATUS	Flag browser. See Section 5 of the OM.
STK	g STK	Menu. See p. 121.
STO	STO r	(0) Stores x into destination.
STOCFG	STO f Config r	(0) Stores the current configuration for later use as described in Section 2 of the OM. RCLCFG recalls such data.
STOEL	f MATX g STOEL	(1) {1, 2, 3}
	STO g ...EL	Stores a copy of x into the indexed matrix at the current element, a_{ij} . Cf. RCLEL.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
STOIJ	f MATX ▲ STOIJ	(1) {1} Sets the index pointers to IP(x) (= column number) and IP(y) (= row number). Cf. RCLIJ.
	[STO] g ...IJ	
STOP	[R/S]	(0) Stops program execution. May be inserted in programs to wait for input, for example.
STOS	[STO] f Stack <i>r</i>	(0) Stores the entire stack in a set of 4 or 8 <i>registers</i> , starting at the destination address specified. See RCLS.
STO+	[STO] + <i>r</i>	(0) Executes the specified operation on <i>r</i> and stores the result (e.g. $r - x$) at the address specified. ²¹
STO-	[STO] - <i>r</i>	
STOx	[STO] × <i>r</i>	
STO/	[STO] / <i>r</i>	
STO↑	[STO] f Max <i>r</i>	(0) {1, 2, 4, 5, 6, 10, 11} Stores the maximum of <i>r</i> and <i>x</i> in the address specified. ²¹
	[STO] ▲ <i>r</i>	
STO↓	[STO] f Min <i>r</i>	(0) {1, 2, 4, 5, 6, 10, 11} Stores the minimum of <i>r</i> and <i>x</i> in the address specified. ²¹
	[STO] ▼ <i>r</i>	
sto→kg	g [U→] <i>m</i> : f stone → kg	(1) {2, 11}; {1} → {2} Converts masses. See pp. 133ff.
STRI?	g [TEST] g STRI?	(0) True if <i>x</i> is an alphanumeric string (like STR? in HP-42S).

²¹ Only legal operations according to the matrices in Section 2 of the OM will work. See also the examples given there.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
STRING	f CATALOG VARS STRING	Submenu of alpha string variables defined at execution time. See pp. 111f.
ST.A		
ST.B		
ST.C		
ST.D		Reserved variables for stack registers A ... Z
ST.T		
ST.X		
ST.Y		
ST.Z		
SUM	g STAT f SUM	(-2) {} → {2} Recalls the linear sums Σy and Σx . Useful in basic 2D vector algebra. Output is labeled in analogy to s.
s_w	g STAT f s_w	(-1) {} → {2} Calculates the standard deviation for weighted data (where the weight y of each data point x was entered via $\Sigma+$). See pp. 203ff for the formula.
s_xy	g STAT ▲ s_xy	(-1) {} → {2} Calculates the sample covariance for the two data sets entered via $\Sigma+$, depending on the curve fit model selected. See pp. 203ff for the formula and COV for the population covariance.
s.INTS	f CATALOG VARS s.INTS	Submenu of short integer variables defined at execution time. See pp. 111f.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
s.t→kg	g [U→] m: ▲ short ton → t	(1) {2, 11}; {1} → {2}
s→year	g [U→] f s→year	Convert masses and times. See pp. 133ff.
T ₀	g [CNST] T₀	(-1) {} → {2} = 0°C, standard temperature in <i>kelvin</i> .
tan	[TRI] tan	(1) {2, 3, 8*, 9*, 11, 12}; {1, 4} → {2} Returns the tangent of the angle in X. Returns “Not a Number” for $x = \pm 90^\circ$ or equivalents if flag D is set.
tanh	g [EXP] g tanh	(1) {2, 3, 8*, 9*, 11, 12} Returns the hyperbolic tangent of x.
TDISP	g [CLK] ▲ TDISP n	(0) Sets time display format. TDISP 0 allows for displaying just <i>hours</i> and <i>minutes</i> , TDISP 2 for <i>seconds</i> , too, and $n \geq 3$ also for $n - 2$ digits showing decimal fractions of <i>seconds</i> . TDISP 1 is interpreted as TDISP 0. TDISP -1 allows for displaying all digits.
TEST	g [TEST]	Menu. See p. 122.
TICKS	g [P.FN] TICKS	(-1) {} → {1} Returns the number of ticks from the real-time clock at execution time. 1 tick = 0.1 s. Counting starts when the calculator is turned on.
TIME	g [CLK] TIME	(-1) {} → {5} Recalls the time from the real-time clock at execution (see Sect. 2 of the OM for the output format).
TIMER	f [TIMER]	Starts the timer application based on the real-time clock and following the timer of HP-55. See Sect. 5 of the OM for a detailed description.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
TIMES	f [CATALOG] VARS f TIMES	Submenu of time variables defined at execution time. See pp. 111f.
T_n	g [X.FN] Orthog T _n	(2) {2, 11}; {1} → {2} <i>Chebyshev polynomials of first kind.</i> See pp. 218f for details.
TONE	g [I/O] f TONE n	(0) Sounds a tone according to n (= 1 ... 9).
ton→kg	g [U→] m: ▲ ton→kg	(1) {2, 11}; {1} → {2} Converts masses. See pp. 133ff.
TOP?	g [TEST] g TOP?	(0) Returns ... <ul style="list-style-type: none"> • false if called with the program pointer being in a subroutine; • true if called in the top routine (i.e. if the program-running flag is set and the SRS pointer is clear).
tor→Pa	g [U→] F&p: f torr → Pa	(1) {2, 11}; {1} → {2} Converts pressures. See pp. 133ff.
T_p	g [CNST] T _p	(-1) {} → {2} <i>Planck temperature in kelvin.</i>
t_{PL}	g [CNST] t _{PL}	(-1) {} → {2} <i>Planck time in seconds.</i>
$t_e(x)$	f [PROB]	(1) {2, 11}
$t_p(x)$	t: t _e (x) etc.	Student's <i>t distribution</i> . The degrees of freedom are stored in J. $t_e(x)$ equals Q(t) on HP-21S. See Section 2 of the OM for an application example and pp. 203ff for more mathematical details.
$t(x)$		
$t^{-1}(p)$		

Item	Keystrokes	Remarks (see pp. 12ff for general information)
TRANS	f CATALOG FCNS TRANS	Works like $[M]^T$ on p. 101. TRANS is featured for backward compatibility only.
TRI	TRI	Menu. See p. 122.
trz→kg	g U→ m: f tr.oz → kg	(1) {2, 11}; {1} → {2} Converts masses. See pp. 133ff.
TVM	g FIN TVM	Application. See Section 5 of the OM.
t:	f PROB t:	Submenu. See p. 120.
t↔	g STK t↔ r	Swaps <i>t</i> and <i>r</i> , in analogy to <i>x</i> ↔
ULP?	g INFO f ULP?	(1) {1, 2, 11} Returns 1 times the smallest power of ten which can be added to <i>x</i> or subtracted from <i>x</i> to actually change the value of <i>x</i> in your WP 43S in the mode set. Thus 1 is returned for integers.
U _n	g X.FN Orthog U _n	(2) {2, 11}; {1} → {2} Chebyshev polynomials of second kind. See pp. 218f for details.
UNITV	f MATX f UNITY	(1) {8, 9} Returns the unit vector for the matrix <i>x</i> (like UVEC in HP-42S). Each element of the matrix is adjusted so its overall Euclidean norm becomes 1 (see ENORM); for a vector, its magnitude will become 1.
	g CPX UNITV	(1) {3, 12} Returns a complex number with magnitude $ r = 1$ in direction of <i>x</i> .

Item	Keystrokes	Remarks (see pp. 12ff for general information)
UNSIGN	[g] [BITS] ▼ UNSIGN [g] [MODE] ▲ ...	(0) Sets unsigned mode for mode for operations on <i>short integers</i> . Cf. UNSGN on HP-16C. See Section 2 of the OM.
U→	[g] [U→]	Menu . See p. 122.
VANGLE	[f] [MATX] [g] [VANGLE]	(2) {8} Returns the angle between two 2D or 3D vectors: $\vartheta = \arccos \left(\frac{\vec{v}_1 \cdot \vec{v}_2}{ \vec{v}_1 \vec{v}_2 } \right)$
VARMNU	[g] [P.FN] P.FN2 [f] [VARMNU] <i>label</i>	Creates a variable menu using MVAR instructions following the global label specified. Cf. the HP-42S Owner's Manual.
VARS	[f] [CATALOG] VARS	Submenu of variables defined at execution time. See pp. 111f.
VERS?	[g] [INFO] [g] VERS?	(0) Shows your firmware version and build number (see Section 2 of the OM).
VIEW	[f] [VIEW] <i>r</i>	(0) Shows <i>r</i> until the next key is pressed. Example: If <i>r</i> is e.g. a variable called Test12 containing -123.45, VIEW Test12 ENTER↑ will display <i>Test12 =</i> -123.45
<i>V_m</i>	[g] [CNST] <i>V_m</i>	(-1) {} → {2} Molar volume of an ideal gas at standard conditions in <i>cubic meter per mol</i> .
V:	[g] [U→] [f] [V:]	Submenu . See p. 133.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
WDAY	g CLK WDAY	(1) {2, 6, 11} → {1} Assumes x containing a date in the format selected (or a real number in corresponding format) and returns the name of the respective day and a corresponding integer (Monday = 1). ²²
Weibl	f PROB	(1) {2, 11}
Weibl _e	f Weibl: Weibl	Weibull distribution with its shape parameter b in I and its characteristic lifetime T in J. See pp. 203ff for details.
Weibl _p	etc.	
Weibl ⁻¹		Weibl ⁻¹ returns the survival time t_s for a given probability p in X, with b in I and T in J.
Weibl:	f PROB f Weibl:	Submenu. See p. 120.
WHO?	g INFO g WHO?	(0) Displays credits to the brave men who made this project work.
Wh→J	g U→ E: Wh→J	(1) {2, 11}; {1} → {2} Converts energies. See pp. 133ff.
W _m	g X.FN ▲ W_m	(1) {2, 3, 11, 12}; {1} → {2}
W _p	etc.	W _p returns the principal branch of Lambert's W for given $x \geq -1/e$. W _m returns its negative branch (works for $x \in \mathbb{R}$ only). W ⁻¹ returns x for a given W _p (≥ -1). See pp. 230ff for more.
W ⁻¹		

²² Translator's note: These day numbers correspond to Chinese weekdays 1 to 6 directly. For Portuguese weekdays ('segunda-feira' etc.), add 1 to days 1 to 5.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
WSIZE	 	<p>(0) Works almost like on <i>HP-16C</i>, but with the parameter $1 \leq n \leq 64$ trailing the command instead of taken from X.</p> <p>Reducing word size truncates the values in the stack as allocated and in L. All other memory content stays as is (see App. B on pp. 152ff).</p> <p>Increasing the word size will add empty bits to each stack register. WSIZE 0 sets the word size to maximum, i.e. 64 bits.</p>
WSIZE?	 	<p>(-1) {} → {1}</p> <p>Recalls the word size set.</p>
W→hp _E	 etc.	
W→hp _M		
W→hp _{UK}		
x ²	 	<p>(1) {1, 2, 3, 8*, 9*, 10, 11, 12}</p> <p>Return the square or cube of x, respectively.</p>
x ³		
XEQ		(0) Executes the function or routine with the label specified. – In PEM, inserts a call to the subroutine with the label specified.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
xIN	[XEQ] type	with type = NILADIC, MONADIC, DYADIC, TRIADIC, or ..._COMPLEX defines how many <i>stack</i> levels are used for parameter input to the function under consideration. Furthermore it does some initialization work (e.g. SSIZE8 and DBLON). xIN is the recommended way to start an <i>XROM routine</i> . Thereafter, SSIZE4 is legal but DBLOFF is not. Note xIN cannot nest and <i>XROM routines</i> using xIN cannot call user code.
XNOR	[f] BITS [f] XNOR	(2) Work in analogy to AND. See p. 18.
XOR	[f] BITS XOR	
xOUT	[XEQ] way	Cleans and reverts the settings of xIN, taking care of a proper return including the correct setting of <i>I</i> and the <i>stack</i> . Typically, way = xOUT_NORMAL . Generally, xOUT shall be the last command of an <i>XROM routine</i> .
\bar{x}	[g] STAT \bar{x}	(-2) {} \rightarrow {2} Calculates the <i>arithmetic means</i> of the <i>y</i> - and <i>x</i> -data accumulated and pushes them on the <i>stack</i> . See also s, s _m , and σ .
\bar{x}_G	[g] STAT [g] \bar{x}_G	(-2) {} \rightarrow {2} Calculates the <i>geometric means</i> of the <i>y</i> - and <i>x</i> -data accumulated and pushes them on the <i>stack</i> . See pp. 212ff for the formula. Output format will be similar to the one of \bar{x} . See also ε , ε_m , and ε_p .
\bar{x}_w	[g] STAT [f] \bar{x}_w	(-1) {} \rightarrow {2} Returns the <i>arithmetic mean</i> for weighted data (where the weight <i>y</i> of each data point <i>x</i> was entered via $\Sigma+$). See pp. 212ff for the formula. See also s _w and s _{mw} .

Item	Keystrokes	Remarks (see pp. 12ff for general information)
\hat{x}		(1) {2, 11}; {1} → {2} Returns a forecast \hat{x} for a given y (in X) according to the curve fit model chosen. See L.R. for more.
X.FN		Menu. See p. 123.
$x!$		(1) {1, 10} Returns the factorial $n!$. Note this only defined for positive integers. $20!$ is the biggest factorial $< 2^{64}$.
		(1) {2, 3, 11, 12} Returns $\Gamma(x + 1)$.
$x:$		Submenu. See p. 133.
$x \rightarrow DATE$		(1) {2} → {6} Interprets the real number x as a date coded in the date format selected (Y.MD, D.MY, or M.DY) and converts it to a proper date.
$x \rightarrow \alpha$		(1) {1, 2, 10, 11} → {7} Interprets x as a character code and converts the integer part of x to the respective character x , similar to XTOA in the HP-42S.
$x \leftrightarrow r$		Swaps x and r , analogous to $x \leftarrow y$. Will be listed like $x \leftrightarrow J$, $x \leftrightarrow .12$, $x \leftrightarrow \rightarrow 12$, etc.
$x \leftrightarrow y$		Swaps the stack contents x and y .
$x = ?$		(0) Compare x with r . See $x < ?$ for more.
$x \neq ?$		

Item	Keystrokes	Remarks (see pp. 12ff for general information)
$x = +0?$	g TEST ▲ x=+0? etc.	(0) {1, 2, 3, 10, 11, 12} These tests are for comparing <i>short integers</i> in modes 1COMPL and SIGNMT, and for <i>long integers</i> , real or complex numbers if flag D is set. Then e.g. $0 / (-7)$ will display -0.
$x = -0?$		
$x \approx ?$	g TEST ▲ x≈? r	(0) {2, 3, 4, 5, 8*, 9*, 11, 12} Will be true if the <u>rounded</u> values of x and r are equal (see ROUND). See $x < ?$ for more.
$x < ?$	g TEST $x < ? r$ etc.	(0) {1, 2, 4, 5, 6, 10, 11} Compare x with r .
$x \leq ?$		
$x \geq ?$		
$x > ?$		
$\sqrt[x]{y}$	g EXP $\sqrt[x]{y}$	(2) {1, 2, 3, 10, 11, 12}; (1) → {2}) Returns the x^{th} root of y . Roots of negative integers or reals may return complex numbers if CPXRES is set.
$\text{yd.} \rightarrow \text{m}$	g U→ $x:$ g yd.} \rightarrow \text{m}	(1) {2, 11}; (1) → {2}) Converts distances. See pp. 133ff.
YEAR	g CLK f YEAR	(1) {2, 6, 11} → {1}) Assumes x containing a <i>date</i> in the format selected (or a real number in corresponding format) and extracts the year.
$\text{year} \rightarrow \text{s}$	g U→ f year} \rightarrow \text{s}	(1) {2, 11}; (1) → {2}) Converts times. See pp. 133ff.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
y^x	y^x	(2) {1, 2, 3, 10, 11, 12} Returns the x^{th} power of y . It allows for raising any positive real number to an arbitrary real power, as well as any negative real number to an arbitrary integer power, all returning real results. Exceeding these boundaries may produce complex results (or errors if CPXRES is not set).
\hat{y}	$\text{g STAT } \Delta \hat{y}$	(1) {2, 11}; {1} → {2} Returns a forecast \hat{y} (in X) for a given x according to the curve fit model chosen. See L.R. for more.
Y.MD	$\text{g CLK } \Delta Y.MD$	(0) Sets the format yyyy-mm-dd for date display.
$y \leftrightarrow r$	$\text{g STK } y \leftrightarrow r$	Swaps y and r , in analogy to $x \leftrightarrow$.
Z_0	$\text{g CNST } Z_0$	(-1) {} → {2} Characteristic impedance of vacuum in ohm.
$z \leftrightarrow r$	$\text{g STK } z \leftrightarrow r$	Swaps z and r , in analogy to $x \leftrightarrow$.
α	$\text{g CNST } \alpha$	(-1) {} → {2} Fine-structure constant.
αINTL	$f \text{ CATALOG } \text{CHARS } \alpha\text{INTL}$	Submenu. See pp. 111ff.
	$\text{g } +$	Menu in AIM (see p. 123).
$\alpha\text{LENG?}$	$\text{g INFO } \text{g } \alpha\text{LENG? } r$	(-1) {} → {1}
	$f \text{ a.FN } f \alpha\text{LENG? } r$	Returns the number of characters found in r , similar to ALENG in HP-42S. ²³

²³ This command will throw an error if there is no string in r at execution time.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
αMATH	f CATALOG CHARS αMATH	Submenu. See pp. 111ff.
	g -	Menu in AIM. See p. 124.
αOFF	g P.FN	(0) Turn AIM off and on, like AOFF and AON in HP-42S. 
	f αOFF etc.	
αPOS?	g INFO g αPOS? r	(-1) {} → {1} Looks in <i>r</i> for the target given in X . If a match is found, αPOS returns the position number where the target was found (counting the left-most character as position 0). If a match is not found, αPOS returns -1. ²³
	f α.FN αPOS? r	The target may be an individual character code or an <i>alpha string</i> . αPOS saves a copy of the target in L . It works similar to POSA in HP-42S.
αRL	f α.FN αRL r	(0) Rotates <i>r</i> by <i>x</i> characters like AROT in HP-42S, but with $x \geq 0$. αRL 0 executes as NOP, but loads L . ²³
αRR	f α.FN αRR r	(0) Works like αRL but rotates to the right.
αSL	f α.FN αSL r	(0) Shifts the <i>x</i> leftmost characters out of <i>r</i> , like ASHF in HP-42S. This allows for deleting the first <i>x</i> characters in the string. αSL 0 executes as NOP, but loads L . ²³
αSR	f α.FN αSR r	(0) Shifts the <i>x</i> rightmost characters out of <i>r</i> , like ASHF in HP-42S. This allows for deleting the last <i>x</i> characters in the string. αSR 0 executes as NOP, but loads L . ²³
α.FN	f α.FN	Menu. See p. 124.
A...Ω	f CATALOG CHARS A...Ω	Submenu of Greek letters, see pp. 111ff.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
$\alpha \bullet$	f CATALOG CHARS $\alpha \cdot$	Submenu. See pp. 111ff.
	g .	Menu in AIM. See p. 124.
$\alpha \rightarrow x$	f a.FN $\alpha \rightarrow x$ r	(-1) {} → {1} Pushes the character code of the leftmost character in r on the stack and removes this character from the string, similar to ATOX in HP-42S. ²³
$\beta(x,y)$	g X.FN ▲ $\beta(x,y)$	(2) {2, 3, 11, 12}; {1} → {2} Returns Euler's Beta $B(x,y) = \frac{\Gamma(x) \Gamma(y)}{\Gamma(x+y)}$ with $Re(x) > 0$ and $Re(y) > 0$. Called β here to avoid ambiguity. See $\Gamma(x)$ below.
γ	g CNST γ	(-1) {} → {2}
γ_{EM}	g CNST γ_{EM}	Newtonian constant of gravitation (also called G by other authors) in $m^3/kg\ s^2$;
γ_p	g CNST γ_p	Euler-Mascheroni constant (for mathematics); proton gyromagnetic ratio (see p. 131).
γ_{xy}	g X.FN ▲ f γ_{xy}	(2) {2, 11}; {1} → {2}
Γ_{xy}	etc.	Return the lower or upper incomplete gamma function, respectively. See pp. 230ff for details.
$\Gamma(x)$	f PROB ▲ $\Gamma(x)$	(1) {2, 3, 11, 12}; {1} → {2} Returns $\Gamma(x)$. Note xJ calls $\Gamma(x + 1)$. See also LNF.
δx	f CATALOG PROGS δx	Predefined global label for $f'(x)$ and $f''(x)$ – see Section 4 of the OM.
$\Delta\%$	f △%	(1) {2, 11}; {1} → {2} Returns $100 \frac{x-y}{y}$ leaving y unchanged, like %CH in HP-42S. Use it also for calculating markups or margins as explained in the OM, Sect. 2.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
ε	g STAT g ε	(-2) {} → {2} Calculates the <i>scattering factors</i> ε_y and ε_x for <i>log-normally</i> distributed sample data and pushes them on the stack. This ε_x works for the <i>geometric mean</i> \bar{x}_g in analogy to the <i>standard deviation</i> s for the <i>arithmetic mean</i> \bar{x} but <u>multiplicative</u> instead of additive. See pp. 203ff for more information.
ε_0	g CNST ε_0	(-1) {} → {2} Electric constant or vacuum permittivity in <i>ampere-second per volt-meter</i> .
ε_m	g STAT g ε_m	(-2) {} → {2} Works like ε on previous page but returns the <i>scattering factors</i> of the two <i>geometric means</i> (in analogy to the standard error for <i>arithmetic means</i>).
ε_p	g STAT g ε_p	(-2) {} → {2} Works like ε but returns the <i>scattering factors</i> of the two populations.
$\zeta(x)$	g X.FN ▲ f $\zeta(x)$	(1) {2, 3} Returns <i>Riemann's Zeta</i> . See p. 233 for more.
λ_c	g CNST λ_c	(-1) {} → {2}
λ_{cn}	etc.	<i>Compton wavelength of the electron, neutron, and proton in meter.</i>
λ_{cp}		

Item	Keystrokes	Remarks (see pp. 12ff for general information)
μ_0	g CNST μ_0 etc.	(-1) { } → {2}
μ_B		Magnetic constant or vacuum permeability in <i>volt-second per ampere-meter</i> ;
μ_e		Bohr magneton in <i>joule per tesla</i> ;
μ_e/μ_B		electron magnetic moment in <i>joule per tesla</i> ;
μ_n		ratio of electron magnetic moment to <i>Bohr</i> magnetons;
μ_p		neutron and proton magnetic moments, nuclear magneton, and
μ_u		Muon magnetic moment in <i>joule per tesla</i> .
μ_μ		
Π	f ADV Π <u>labl</u>	Computes a product using the routine specified. See Section 4 of the OM for more. ATTENTION: Π fills all <i>stack registers</i> with x before calling the routine specified.
π	g T	(-1) { } → {2} Recalls π .
Σ	f ADV Σ <u>labl</u>	Computes a sum using the routine specified. See Section 4 of the OM for more. ATTENTION: Σ fills all <i>stack registers</i> with x before calling the routine specified.
s	g STAT s	(-2) { } → {2} Works like s but returns the <i>standard deviations</i> of the two <i>populations</i> instead. See pp. 203ff.
g_B	g CNST g_B	(-1) { } → {2} <i>Stefan-Boltzmann constant</i> (see p. 132).

Item	Keystrokes	Remarks (see pp. 12ff for general information)
$\Sigma \ln^2 x$	g STAT ▼ f $\Sigma \ln^2 x$ etc.	(-1) {} → {2} Recall the corresponding statistical sums, necessary for curve fit models beyond pure linear. Calling the sums by name significantly improves program readability. Note they are stored in dedicated <i>registers</i> of your WP 43S (see App. B on pp. 152ff.).
$\Sigma \ln^2 y$		
$\Sigma \ln x$		
$\Sigma \ln xy$		ATTENTION: Depending on input data, some of the logarithmic sums may become non-numeric. If this happens no error will be thrown, however, regardless of flag D.
σ_w		(-1) {} → {2} Works like s_w but returns the <i>standard deviation of the population</i> instead. See pp. 203ff.
Σx	g STAT ▼ Σx etc.	(1) {} → {2} Recall the corresponding statistical sums, necessary for basic statistics and fits (see above).
Σx^2		
$\Sigma x^2 y$		
$\Sigma x \ln y$		
Σxy		
Σy		
Σy^2		
$\Sigma y \ln x$	g STAT ▼ g $\Sigma y \ln x$	
$\Sigma +$	g STAT Σ+	(8) → {2} If X contains an $n \times 2$ matrix then $\Sigma +$ adds n 2D data points to the statistical sums. The <i>temporary information</i> mentioned in the footnote will show the last data point added. ²⁴ {1, 2, 11} Adds one 2D data point to the statistical sums. ²⁴

Item	Keystrokes	Remarks (see pp. 12ff for general information)
$\Sigma-$	 	{1, 2, 11} Subtracts one 2D data point from the statistical sums. ²⁴
Φ	 	(-1) {} → {2}
Φ_0	 	Golden ratio and magnetic flux quantum, the latter in <i>volt-second</i> .
$\Phi_e(x)$	 	(1) {2, 11}
$\Phi(x)$...	Standard normal (a.k.a. Gaussian) error probability with $\mu=0$ and $\sigma=1$ (equal to Q in HP-32E and Q(z) in HP-21S), <i>CDF</i> , <i>PDF</i> , and <i>quantile function</i> . See pp. 203ff for the formulas. – Take NORM1 for arbitrary means and standard deviations.
$\varphi(x)$...	
$\Phi^{-1}(p)$...	
$\Phi:$	 	Submenu. See p. 120.
$\chi^2(x)$	 	(1) {2, 11} <i>Chi-square distribution</i> (with its degrees of freedom given in I). $\chi^2_u(x)$ equals $Q(\chi^2)$ on HP-21S. See Section 2 of the OM for an application example and pp. 203ff for more.
$\chi^2_e(x)$	 etc.	
$\chi^2_p(x)$		
$(\chi^2)^{-1}$		
$\chi^2:$	 	Submenu. See p. 120.
ω	 	(-1) {} → {2}
		Angular velocity of the Earth in <i>radian per second</i> according to WGS84 (see footnote 20 on p. 47).

²⁴ $\Sigma+$ and $\Sigma-$ return *temporary information* as shown in Section 2 of the OM and disable *automatic stack lift*. Both may also be used for 2D vector adding and subtracting (see SUM and the corresponding example in Section 2 of the OM).

Item	Keystrokes	Remarks (see pp. 12ff for general information)									
$(-1)^x$	g X.FN ▲ f (-1)^x	(1) {1, 2, 3, 8*, 9*, 10, 11, 12} If x is non-integer, returns $\cos(\pi x)$.									
+	[+]	(2) Returns $y + x$ for compatible objects. See the tables in the OM, Sect. 2, for details.									
+/-	[+/-] (for closed input)	(1) ‘Unary minus’, returns $x \times (-1)$. See the tables in the OM, Sect. 2, for details.									
-	[-]	(2) Returns $y - x$ for compatible numeric objects. See the tables in the OM, Sect. 2, for details.									
$-\infty$	g CNST $-\infty$	(-1) {} → {2} Minus infinity. May the Lord of Mathematics forgive us calling this a constant, but it is counted as a numeric value in your WP 43S.									
\times	[x]	(2) Returns $y \times x$ for compatible numeric objects. See the tables in Section 2 of the OM for details.									
$\times\text{MOD}$	g INTS f $\times\text{MOD}$	(3) {1, 2, 10, 11} Returns $(z \times y) \bmod x$ for $x > 1$, $y > 0$, $z > 0$.									
/	[/]	(2) Returns $y \times x^{-1}$ for compatible numeric objects. See the tables in the OM, Sect. 2, for details. Returns $y \div x$ if both y and x are of data type 1 or 10. Cf. IDIV.									
$\pm\infty?$	g INFO g $\pm\infty?$	(0) {2, 11} → {1} Tests x for infinity. Returns <table style="margin-left: 20px;"> <tr> <td>true</td> <td>1</td> <td>for $x = +\infty$,</td> </tr> <tr> <td>true</td> <td>-1</td> <td>for $x = -\infty$, and</td> </tr> <tr> <td>false</td> <td>0</td> <td>else.</td> </tr> </table>	true	1	for $x = +\infty$,	true	-1	for $x = -\infty$, and	false	0	else.
true	1	for $x = +\infty$,									
true	-1	for $x = -\infty$, and									
false	0	else.									
\rightarrow		Reserved symbol for indirect addressing.									

Item	Keystrokes	Remarks (see pp. 12ff for general information)
→DATE		(3) {1, 2} → {6} Assumes the three components of a date (year, month, and day) supplied on the stack in proper order for the date format selected and converts them to a single <i>date</i> in x . Thus inverts DATE→.
→DEG		(1) {1, 2, 4, 11} → {4} Converts angles as described on pp. 140f.
→DP		(1) {1, 2, 11} → {11}; {3, 12} → {12} Converts x into a DP number. Numbers shown as fractions will be displayed as decimal numbers (cf. IMPFRC and PROFRC). Compare →SP.
→D.MS		(1) {1, 2, 4, 11} → {4}
→GRAD		Convert angles as described on pp. 140f.
→HR		(1) {5} → {2} Operates on times like →REAL below. →HR is featured for backward compatibility only..
→H.MS	 (for closed input)	(1) {1, 2, 5, 11} → {5} Converts x to a sexagesimal <i>time</i> – cf. p. 107.
→INT	 (for closed input)	(1) {1, 2, 10, 11} → {10} Converts the integer part of x to a <i>short integer</i> of the base specified – cf. p. 107.
→MULπ		(1) {1, 2, 4, 11} → {4} Converts angles as described on pp. 140f.

Item	Keystrokes	Remarks (see pp. 12ff for general information)									
→POL	g →P	<p>{2, 11}; {1} → {2}</p> <p>Assumes X and Y containing 2D <i>Cartesian</i> coordinates of a point or components of a vector (x, y). Converts them to the respective polar coordinates or components (r, θ). See the picture and cf. →REC.</p> <table border="1"> <tr> <td></td> <td>x</td> <td>y</td> </tr> <tr> <td>→P</td> <td>r</td> <td>θ</td> </tr> <tr> <td>→R</td> <td>x</td> <td>y</td> </tr> </table> <p>For switching the format of complex numbers, choose POLAR.</p>		x	y	→P	r	θ	→R	x	y
	x	y									
→P	r	θ									
→R	x	y									
→RAD	f L → 4→4 ^r	<p>(1) {1, 2, 4, 11} → {4}</p> <p>Converts angles as described on pp. 140f</p>									
→REAL	f .d (for closed input)	<p>(1) {1, 2, 4, 5, 6, 10, 11} → {2}</p> <p>Converts x to an SP real number. For {6}, the date format chosen is taken into account. Numbers shown as fractions will be displayed as decimal numbers (cf. IMPFRC and PROFRC).</p> <p>For returning the real part of a complex number, choose RE. For cutting a complex number into its parts, use CC.</p>									
→REC	f R →	<p>{2, 11}; {1, 4} → {2}</p> <p>Assumes X and Y containing 2D polar coordinates of a point or components of a vector (r, θ). Converts them to the respective Cartesian coordinates or components (x, y). See →POL.</p> <p>For switching the format of complex numbers, choose RECT.</p>									

Item	Keystrokes	Remarks (see pp. 12ff for general information)
→SP		(1) {1, 2, 11} → {2}; {3, 12} → {3} Converts x to an SP real number. Numbers shown as fractions will be displayed as decimal numbers (cf. IMPFRC and PROFRC). Compare →DP.
↑Lim		Reserved real variables for the upper and lower limit of integration (see Section 4 of the OM).
⇄		<p>Shuffles the contents of the bottom four <i>stack registers</i> at execution time.</p> <p>Examples:</p> <ul style="list-style-type: none"> ⇄xyz works like ENTER↑ (but does not disable <i>automatic stack lift!</i>), ⇄yxzt works like $x \leftarrow y$, ⇄yztx works like R↓ in a 4-level <i>stack</i>, ⇄txyz works like R↑ in a 4-level <i>stack</i>, but also ⇄yytt or ⇄zzzx is possible. <p>ATTENTION: This is a very powerful command although it does not look it. Note it will only affect the <u>lowest four stack registers</u> regardless of <i>stack size</i>. There is <u>no</u> connection to <i>registers A ... D</i>. Playing with ⇄, you may quickly lose some <i>stack</i> contents and easily make a mess of the <i>stack</i>.</p>

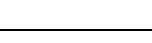
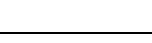
Item	Keystrokes	Remarks (see pp. 12ff for general information)
%		(1) {2, 11}; {1} → {2} Returns $\frac{xy}{100}$, leaving y unchanged.
%MRR		(3) {2, 11}; {1} → {2} Returns the mean rate of return in percent per period, i.e. $100 \cdot \left(\sqrt[z]{x/y} - 1 \right)$ with $x = FV$ = future value after z periods, $y = PV$ = present value. For $z = 1$, Δ% returns the same result easier.
%T		(2) {2, 11}; {1} → {2} Returns $\frac{100 x}{y}$, interpreted as % of total.
%Σ		(1) {2, 11}; {1} → {2} Returns $\frac{100 x}{\sum x}$.
%+MG		(2) {2, 11}; {1} → {2} Calculates a sales price by adding a margin of x % to the cost y , as %MU-Price in HP-17B. Formula: $p_{sale} = \frac{y}{1 - \frac{x}{100}}$ You may use %+MG for calculating net amounts as well; just enter a negative percentage in x .
\sqrt{x}		(1) {1, 2, 3, 8*, 9*, 10, 11, 12}; {(1)} → {2}) Returns the square root of x . Square roots of negative integers or reals will return complex numbers if CPXRES is set.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
\int	f [ADV] \int_{fdx} f var (listed in programs as $\int fd$ trailed by the integration variable)	<p>{2}</p> <p>Integrates the function given in the routine specified by PGMINT over the variable specified. Lower and upper integration limits must be supplied by the corresponding variables \downarrowLim and \uparrowLim, accuracy by ACC. \int returns the (approximated) integral in X and an upper limit of its uncertainty in Y.</p> <p>ATTENTION: \int fills all <i>stack registers</i> with x before calling the routine specified.</p>
	g [EQN] $\int f$ f	Integrates the current equation. Not programmable.
$\int f$	g [EQN] $\int f$	Submenus. See pp. 117f.
$\int f dx$	f [ADV] $\int f dx$	
∞	g [CNST] ∞	<p>(-1) {} → {2}</p> <p>Infinity. May the Lord of Mathematics forgive us calling this a constant, but it is counted as a numeric value in your WP 43S.</p>
${}^{\text{MOD}}$	g [INTS] g ${}^{\text{MOD}}$	<p>(3) {1, 2, 10, 11}</p> <p>Returns $(z^y) \bmod x$ for $x > 1$, $y > 0$, $z > 0$.</p>
$ M $	f [MATX] M 	<p>(1) {8} → {2}; {9} → {3}</p> <p>Requires a square matrix in X and returns its determinant. The matrix itself is stored in L unmodified.</p>

Item	Keystrokes	Remarks (see pp. 12ff for general information)
x	f x or f PARTS x or g CPX x 	<p>(1) {1, 2, 4, 10, 11}</p> <p>Returns the absolute (unsigned) value of x.</p> <p>(1) {3} → {2}; {12} → {11}</p> <p>Returns the magnitude $\sqrt{\operatorname{Re}(x)^2 + \operatorname{Im}(x)^2}$ in X.</p> <p>(1) {8*, 9*} → {8}</p> <p>Returns a real matrix with the absolute (unsigned) value of each input matrix element. Complex matrix elements will become real this way. Cf. ENORM.</p>
	g X.FN ▲ g 	<p>(2) {2, 3, 11, 12}; {1} → {2}</p> <p>Returns $\left(\frac{1}{x} + \frac{1}{y}\right)^{-1}$, being useful in electrical engineering especially. Returns 0. for x or y being zero.</p>
[M] ^T	f MATX [M]^T	<p>(1) {8, 9}</p> <p>Returns the transpose of the matrix x (like TRANS in HP-42S). The transpose is another matrix with rows changed by columns. If A is an $n \times m$ matrix and a_{ij} is an element of it then A^T will be an $m \times n$ matrix B with $b_{ij} = a_{ji}$. The transpose is done in-situ and does not require any additional memory.</p>
[M] ⁻¹	f MATX [M]⁻¹	<p>(0) {8, 9}</p> <p>Takes the square matrix in X and inverts it in-situ (like INVRT on HP-42S).</p>
↳	g CPX f ↳ <hr/> f PARTS g ↳	<p>(1) {3} → {2}; {9*} → {8}; {12} → {11}</p> <p>Returns $\arctan\left(\frac{\operatorname{Im}(x)}{\operatorname{Re}(x)}\right)$ for $x \geq 1$, else $\arctan\left(\frac{\operatorname{Im}(x)}{\operatorname{Re}(x)}\right) + \pi \operatorname{sign}(\operatorname{Im}(x))$. Cf. SIGN and x .</p>
↳→	f L→	<p>Menu. See p. 125.</p>

Item	Keystrokes	Remarks (see pp. 12ff for general information)
■ADV	g I/O ▲ ■ADV	(0) Prints the current contents of the print buffer and a linefeed. ATTENTION: The printer will actually print only when a linefeed is sent to it.
■CHAR	g I/O ▲ ■CHAR n	(0) Sends a single character (with the code specified) to the printer. Character codes n > 127 can only be specified indirectly. ■MODE setting will be honored. See ■ADV.
■DLAY	g I/O ▲ ■DLAY n	(0) Sets a delay of n ticks (see TICKS) to be used with each linefeed on the printer.
■LCD	g I/O ▲ ■LCD	(0) Sends the contents of the entire <i>LCD</i> to the printer.
■MODE	g I/O ▲ ■MODE n	(0) Sets print mode. Legal print modes are: 0: Use the printer font and character set wherever possible (default). All characters feature the same width (5 columns + 2 columns spacing). 1: Use the variable pitch display font, resulting in some jitter on the printout but packing more characters in a row. 2: Use the small display font, which allows for packing even more info in a row. 3: Send the output to the serial channel. Works for plain ASCII only – no characters will be translated. Line setup is the same as for serial communication: 9600 baud, 8 bits, no parity.
■PROG	g I/O ▲ ■PROG	(0) Prints the listing of the <i>current program</i> (see Sect. 3 of the OM), 1 row per step. See ■ADV.
■r	g I/O ▲ f ■r r	(0) Prints the <i>register</i> specified, right adjusted, <u>without</u> labeling the output. Note ■X is on the keyboard. If you want a heading label, compose the string in X first or use ■REGS. See ■ADV.
	f ■X	

Item	Keystrokes	Remarks (see pp. 12ff for general information)
REGS		(1) Interprets x in the form $sss.nn$. Prints the contents of nn registers starting with number sss . Each register takes one row starting with a label. See also \blacksquare ADV. ATTENTION for $nn = 0$: <ul style="list-style-type: none"> • For $sss \in [0; 99]$, printing will stop at R99. • For $sss \in [100; 111]$, printing stops at K. • For $ss \geq 112$, printing will stop at the highest allocated local register.
STK		(0) Prints the stack contents. Each register prints in a separate row starting with a label indicating said register. See \blacksquare ADV.
TAB		(0) Positions the print head to print column n (0 to 165, where $n > 127$ can only be specified indirectly). Useful in formatting (in \blacksquare MODE 1 or 2 in particular). Allows also for printer plots. If n is less than current print head position, a linefeed will be entered to reach the new position. See \blacksquare ADV.
USER		(0) Prints all variable names and global program labels. The variable names are printed first, so if you are not interested in the program labels, press R/S to stop the listing.
WIDTH		(-1) Returns the number of print columns that x would take in the print mode set. See \blacksquare ADV and \blacksquare MODE. Second use: in \blacksquare MODE 1 or 2, \blacksquare WIDTH returns the width of x in pixels (including the last column being always blank) in the specified font.
Σ		(0) Prints the summation registers. Each register prints in one row starting with its label. See \blacksquare ADV.

Item	Keystrokes	Remarks (see pp. 12ff for general information)
¤#		(0) Sends a single <i>byte</i> , without translation, to the printer (e.g. a control code). <i>n</i> > 127 can only be specified indirectly. ¤MODE setting will not be honored. See ¤ADV.
#		For inserting an integer $0 \leq n \leq 255$ in a single program step. Maintained for backward compatibility to WP 34S only.
#B		(1) {10} Counts the bits set in <i>x</i> (like on HP-16C).

Nonprogrammable Commands and Keys

The commands ASSIGN, CLALL, CLP, CLPALL, CLREGS, FBR, GTO., LOAD, LOADP, PSTO, RBR, RESET, SAVE, STATUS, TIMER, and TVM cannot be programmed. The same applies to the operations of the *Matrix Editor* and *Equation Editor*, as well as answers to questions your WP 43S asks.

Furthermore, all *catalog* and *menu* calls themselves as well as the operations called by **EXIT**, **P/R**, **USER**, **α** , **\leftarrow** , **$\equiv\Delta$** , **\blacktriangle** , **$\equiv\triangledown$** , and **\blacktriangledown** are neither programmable nor will they show any input echo in the top numeric row as the other commands do (cf. the OM, Sect. 2). See also *Section 2: Menus and Catalogs* (on pp. 111ff) for more about this topic.

The *browsers* RBR and STATUS as well as the *application* TIMER use some keys for particular control purposes (e.g. **STO**, **RCL**, **.**, and numeric keys – see the *OM, Section 5*).

Command Parameter Input and Closing It

The following table shows what will happen when particular keys are pressed while command parameter input is not finished yet (see pp. 107ff for input in X instead). Note that characters include also digits and punctuation marks. The table lists the respective keys beginning top left on the keyboard:

Keystrokes	Situation	Remarks
[A] ... [Z]	addressing	Register or flag input. See the <i>virtual keyboard</i> in Section 1 of the OM for the letters applicable.
[A] ... [Z]	entering a label or a variable name	Appends the corresponding Latin or Greek letter or digit to the label or variable name pending. Use [▼] and [▲] to switch cases. See the picture overleaf (and cf. Section 2 of the OM).
[g] [A] ... [g] [O]		
[f] [0] ... [f] [9]		
[ENTER↑]	arbitrary parameter input pending	If there is no input yet, assumes the default, if applicable. Closes pending input, interprets it as a <i>register</i> or <i>flag</i> address or a variable name or a label or alike, and executes the command. Cf. Section 1 of the OM.
[◀]	arbitrary parameter input pending	Deletes the rightmost character keyed in. If there is nothing left, cancels the pending command, returning to the status of your WP 43S as it was before that input was started.
[0] ... [9]	addressing	Numeric parameter input. See Section 1 of the OM for the valid number ranges.
[.]	addressing	Header for <i>local registers</i> or <i>flags</i> .
[EXIT]	arbitrary parameter input pending	If there is an open <i>menu</i> , closes it. Else cancels pending command input, returning to the status of your WP 43S as it was before that command was called.



Alphanumeric Input in X and Closing It

The following table shows what will happen when particular keys are pressed while alphanumeric or numeric input in X is open still (see pp. 105f for command parameter input instead). The table lists the respective keystrokes top left to bottom right on the keyboard:

Keystrokes in mode(s)	Remarks
f # base	$\neg(A, \alpha)$ Closes input of a <i>short integer</i> in X. ²⁵
f d-ms	$\neg(A, \alpha)$ Closes input of a sexagesimal angle in X. ²⁵
f .d	$\neg(A, \alpha)$ Closes input of a <i>date</i> in X. ²⁵
f h-ms	$\neg(A, \alpha)$ Closes input of a sexagesimal <i>time</i> in X. ²⁵
A ... Z	$\neg(A, \alpha)$ Appends the corresponding Latin or Greek letter to the <i>alpha string</i> x . Use ▼ and ▲ to switch cases. See the picture on previous page and cf. <i>Section 2</i> of the OM.
g A ... g O	
CC	$\neg(A, \alpha)$ Closes input of the first part of a complex number in X and waits for input of its second part (see the <i>Key Response Table</i> and <i>Section 2</i> of the OM).
f (R↓) (▼)	$\neg(A, \alpha)$ Prefix for the next character becoming a subscript, if applicable.

²⁵ See *Section 2* of the OM. At closure, input will be checked – illegal digits (e.g. 8 in octal input or C in decimal) or bases or numbers (e.g. 72 *minutes* in a *time*) or characters found, or out-of-range conditions detected will cause an error thrown (see also the description of **ENTER↑** on next page and the error messages in *App. C*).

Keystrokes	in mode(s)	Remarks
ENTER↑	arbitrary input pending	<p>If there was no input before, sets input to zero.</p> <p>Else closes input (in X) and checks the following conditions top-down:</p> <ul style="list-style-type: none"> • If this input is <u>alphanumeric</u> (i.e. if it contains at least one non-numeric character except .), takes it as an <i>alpha string</i>. • Else (i.e. if this input is purely numeric) if it contains one CC, takes it as a complex number. • Else if it contains two ., takes it as a <i>fraction</i>. • Else if it contains one . or one E, takes it as a <i>real number</i>. • Else (i.e. if it contains neither a CC nor a . nor an E), tests it for #: <ul style="list-style-type: none"> ◦ If it contains one # and a valid base trailing it then takes it as a <i>short integer</i>; ◦ else looks up if <u>previous entry</u> was a <i>short integer</i>: if true then takes the new input as another <i>short integer</i> of the same base; else takes the new input as a <i>long integer</i>. <p>Then checks the new input (according to the condition met) as outlined in footnote 25 and interprets it. Finally, unless an error had to be thrown, copies <i>x</i> into Y.</p>
f [x]	A, α	Appends x to the <i>alpha string</i> <i>x</i> .
+/-	$\neg(A, \alpha)$	Changes the sign of the mantissa or exponent in numeric input as explained in <i>Section 1</i> of the OM.
f [+/-]	A, α	Appends ± to the <i>alpha string</i> <i>x</i> .
E	$\neg(A, \alpha)$	Closes input of the mantissa and waits for input of the exponent (see <i>Section 1</i> of the OM).
f [E] (^)	A, α	Prefix for the next character becoming a superscript, if applicable.

Keystrokes	in mode(s)	Remarks
	arbitrary input pending	Deletes the last (rightmost) character keyed in. If there is nothing left, cancels the pending input, returning to the status of your WP 43S as it was before that input was started.
	A, α	Appends to the <i>alpha string</i> x .
	A, α	Appends to the <i>alpha string</i> x .
	$\neg(A, \alpha)$	Standard numeric input, appending the corresponding digit to x . Note you can enter ... <ul style="list-style-type: none"> up to 16 digits plus a sign in the mantissa and up to three digits plus a sign in the exponent for a real number or any part of a complex number, an arbitrary number of digits plus a sign for a <i>long integer</i>, up to 64 bits for a <i>short integer</i>, or up to 16 digits for the nominator and up to 4 digits for the denominator of a fraction.
	A, α	Appends the respective digit to the <i>alpha string</i> x .
 	A, α	Turns to upper (or lower) case for the following letter(s).
	A, α	Appends to the <i>alpha string</i> x .
 A ... F	$\neg(A, \alpha)$	Numeric input for bases >10, appending the corresponding digit to x . See Section 2 of the OM for more. Digits will be checked when input is closed (see the description of above).
	A, α	Appends to the <i>alpha string</i> x .
	A, α	Appends to the <i>alpha string</i> x .

Keystrokes in mode(s)		Remarks
	\neg (A, α)	Inserts a radix mark as selected. Separates <i>degrees</i> from <i>minutes</i> , <i>seconds</i> , and <i>hundredths of seconds</i> in angular input, so input format is dddd.dddd.mmsshh d.ms for sexagesimal angles. Separates <i>hours</i> from <i>minutes</i> , <i>seconds</i> , and fractions of <i>seconds</i> in <i>time</i> input, so input format is hhhh.ffffmmssffff h.ms for sexagesimal <i>times</i> .
Second	\neg (A, α)	A second in input indicates a fraction. See <i>Section 2</i> of the OM for examples. The second just separates the nominator and the denominator in input. Note you cannot enter E after you entered twice – but you may delete the second dot while editing the input row.
	A, α	Appends , to the <i>alpha string x</i> .
	A, α	Appends . to the <i>alpha string x</i> .
	program waiting for arbitrary input	Closes input and starts its checks and interpretation like ENTER↑ above.
	A, α	Appends a blank space to the <i>alpha string x</i> .
EXIT	arbitrary input pending	If there is an open <i>menu</i> , closes it. Else closes pending numeric or alphanumeric input and releases it for interpretation.

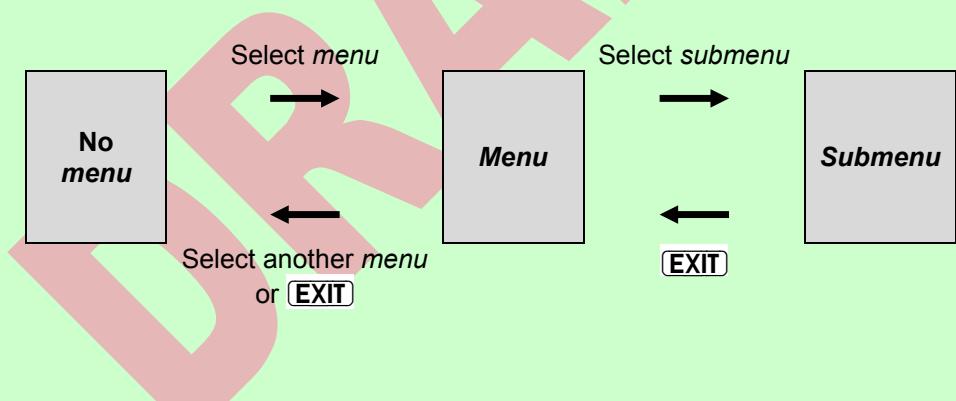
There are many more characters you can enter via the three alpha *menus* or **CATALOG CHARS**. See pp. 112 and 123ff for these menus and pp. 170ff for the entire character sets provided.

SECTION 2: MENUS AND CATALOGS

Due to the large set of operations your *WP 43S* features, most of them are stored in *menus* as they were discussed in the OM, Section 1. Besides operations, numeric constants, or characters (as in the alpha *menus*), there may be also other *items* contained in *menus* (e.g. *submenus*, digits).

Catalogs are a special kind of *menus* with their contents sorted alphabetically. Your *WP 43S* provides two *catalogs*, CATALOG and CNST. Also some *submenus* of CATALOG are treated as *catalogs* (i.e. A...Z, DIGITS, FCNS, MENUS, aINTL, A...Ω, and the *submenus* of VARS, see next chapter). Within *catalogs*, some special operations ease your path accessing the *items* stored there (as shown on pp. 115f).

You may switch *menus* (except *catalogs*) easily by just calling another *menu* accessible in current mode directly from the *menu* you are using – no need to **EXIT** first:



One to Rule Them All – the CATALOG

CATALOG calls a very particular *menu*: CATALOG contains all the *items* defined on your *WP 43S* and visible for the user. Many of them are sorted alphabetically in different branches; these *items* we call *cataloged*. Individual *cataloged items* may be accessed quickly in a way demonstrated on pp. 115f.

Note the contents of the various branches of CATALOG are presented below in reverse order compared to the display of your WP 43S, taking care of your top-down reading habits:

							Remarks
CATALOG:	FCNS	DIGITS	CHARS	PROGS	VARS	MENUS	top branches
FCNS:	$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$	10^x	1COMPL	$1/x$	2^x	functions provided
	2COMPL	$\sqrt[3]{x}$	ABS	$\text{ac} \rightarrow \text{m}^2$	$\text{ac}_{\text{us}} \rightarrow \text{m}^2$	AGM	
	ALL	AND	arccos	arcosh	arcsin	arctan	
	...						
	...	#B					
DIGITS:	0	1	2	3	4	5	digits defined
	6	7	8	9	A	B	
	C	D	E	F	i		
CHARS:	A...Z	A...Ω	αINTL	αMATH	Myα	α•	character branches
A...Z:	A	B	C	D	...		plain Latin letters
A...Ω:	A	B	Γ	Δ	...		Greek letters, see p. 124
αINTL:	Ā	Á	Ă	...			additional (international) Latin letters, see p. 123
αMATH:	<	≤	...				mathematical operators and symbols, see p. 124
α•:	!	...					punctuation marks, see p. 125
PROGS:	RAM					FLASH	global labels currently defined
RAM:	...						empty at startup but will be filled with your creations
FLASH:	...						

							Remarks
VARS:	L.INTS	S.INTS	REALS	CPXS	STRING	MATRS	branches for various types of variables
	DATES	TIMES	ANGLES				
ANGLES:	...						
CPXS:	...						
DATES:	...						
L.INTS:	...						variables currently defined of various <i>data types</i> – all these <i>menus</i> are empty at startup but will be filled and grow with your creations
MATRS:	...						
REALS:	...						
STRING:	...						
S.INTS	...						
TIMES:	...						
MENUS:	ANGLES	A....Z	A:	Binom:	BITS	Cauch:	(sub-) menus currently defined (shown here at startup, but this list will grow due to your creations) – see above and below for their fix contents
	CHARS	CLK	CLR	CNST	CPX	CPXS	
	DATES	DIGITS	DISP	EQN	EXP	Expon:	
	...						
	...	4→					
	...						
A:	...						(sub-) menus provided unless mentioned above already, see pp. 117ff – also this will get new entries for your creations
Binom:	...						
BITS:	...						
...							
...							

Three branches of **CATALOG** are fixed size (**FCNS**, **DIGITS**, and **CHARS**) since all valid functions, digits, and characters are predefined; the other three are expandable (**MENUS** and the *submenus* of **PROGS** and **VARS**) since you may create *items* of these kinds (cf. Section 6 of the OM).

Calling **CATALOG** will display its top level branches (one row of labels, being pointers to the *submenus* containing all the functions,



digits, characters, programs, variables, and *menus* defined at execution time):

Choosing one of these branches will show its first view of *items* (primary, **f**- and **g**-shifted, as applicable). Pressing the leftmost softkey, for instance, will call the *submenu* **FCNS** showing up as pictured below:

	ALL	AND	arccos	arcosh	arcsin	arctan	
2COMPL	$\sqrt[3]{x}$	ABS	$ac \rightarrow m^2$	$acus \rightarrow m^2$	AGM		
${}^{\circ}C \rightarrow {}^{\circ}F$	${}^{\circ}F \rightarrow {}^{\circ}C$	10^x	1COMPL	$1/x$	2^x		

Within CATALOG branches, browsing by **▲** will advance by only six *items* per keystroke (and **▼** will go back by six).²⁶ Select an *item* by pressing the corresponding top row key (headed by a *prefix* if applicable). E.g. call a function by pressing the corresponding softkey. **EXIT** will just leave CATALOG without doing anything.

All the functions available on your *WP 43S* are stored in CATALOG'FCNS. Most of them are also found (and easier to access) in other predefined *menus*. Each and every command featured on your *WP 43S*, the keystrokes calling it, and the necessary particular explanations are also listed for your reference in the *IOI* on pp. 12ff.

Remember all labels printed underlined on the keyboard point to *menus*. All *menus* available are found in CATALOG'MENUS; predefined *menu* names are listed in the *IOI* as well; their particular contents are printed in next chapter. Individual *items* may appear in more than one *menu* and also on the keyboard.

See Section 6 of the OM to learn how to customize your *WP 43S* by creating and filling your own *menus* (assignable to your favorite keyboard locations) and accessing the functions you stored therein. You may as well assign your favorite individual functions to almost any location on the keyboard. Actually, you can design your very own *WP 43S* user interface.

²⁶ Navigating in 'CATALOG, AIM is set as explained in the OM. So you may as well use the alphabetic searching method known from *WP 34S catalogs*, but the matching *item* will be displayed together with its up to 17 successors if applicable. See next chapter.

Accessing Cataloged Items Rapidly

You can browse a *catalog* like any other *menu* just using \blacktriangle and \blacktriangledown as explained in previous chapter. In **CNST** and major parts of **CATALOG** (**FCNS**, **MENUS**, **PROGS** **RAM** or **FLASH**, **CHARS** **INTL**, and the *submenus* of **VARS**), however, you may reach your target significantly faster taking advantage of the alphabetical access method demonstrated here.

Assume we are looking for the function FS?S, for **example**:

1 User input	CATALOG FCNS Your WP 43S displays the first view in this catalog ²⁷																		
Echo	<table border="1"><tr><td>ALL</td><td>AND</td><td>arccos</td><td>arcosh</td><td>arcsin</td><td>arctan</td></tr><tr><td>2COMPL</td><td>$\sqrt[3]{x}$</td><td>ABS</td><td>$ac \rightarrow m^2$</td><td>$ac_{us} \rightarrow m^2$</td><td>AGM</td></tr><tr><td>$^{\circ}C \rightarrow ^{\circ}F$</td><td>$^{\circ}F \rightarrow ^{\circ}C$</td><td>$10^x$</td><td>1COMPL</td><td>$1/x$</td><td>$2^x$</td></tr></table>	ALL	AND	arccos	arcosh	arcsin	arctan	2COMPL	$\sqrt[3]{x}$	ABS	$ac \rightarrow m^2$	$ac_{us} \rightarrow m^2$	AGM	$^{\circ}C \rightarrow ^{\circ}F$	$^{\circ}F \rightarrow ^{\circ}C$	10^x	1COMPL	$1/x$	2^x
ALL	AND	arccos	arcosh	arcsin	arctan														
2COMPL	$\sqrt[3]{x}$	ABS	$ac \rightarrow m^2$	$ac_{us} \rightarrow m^2$	AGM														
$^{\circ}C \rightarrow ^{\circ}F$	$^{\circ}F \rightarrow ^{\circ}C$	10^x	1COMPL	$1/x$	2^x														
2 User input	First character of the <i>item</i> desired (e.g. F)																		
Echo	Your WP 43S displays a view starting with the first <i>item</i> starting with this character ²⁸																		

²⁷ ... unless you visited the same *catalog* before – then it will open showing the last view you looked at. The remaining procedure will stay unchanged though.

²⁸ This search is case independent (i.e. specifying **A** will find **a** as well). Note, however, that **A** and **a** remain different letters nevertheless. Remember you can also enter Greek letters in such a search using prefix **g**, e.g. **g + A** for **α** – though watch the sorting order printed at the beginning of the *IOI*. And **P** will bring you to the print functions immediately. Note the *items* in the *catalog* you search may be displayed at positions in the *menu section* deviating from the ones you see in simple browsing using just \blacktriangledown or \blacktriangle .

You may put in more than one character (see overleaf) – though after 3 *seconds* or after pressing \blacktriangledown or \blacktriangle , whatever comes first, the search string will be reset. Then you may continue browsing using \blacktriangledown or \blacktriangle or start a new search by entering a new first character.

If a character or sequence specified is not found then the first *item* following alphabetically will be shown – see the sorting order in the *IOI*. If there is no such *item*, then the last *item* in this *catalog* will be displayed.

		e.g.								
3 User input	FP	FP?	F _p (x)	FS?	FS?C					
	FF	FIB	FILL	FIX	FLASH?					
	FAST	FB	FC?	FC?C	FC?F					
		Second character of the <i>item</i> desired (e.g. S)								
Echo	Your WP 43S displays a view starting with the first item starting with this sequence									
	e.g.									
	g _d	g _d ⁻¹	Geom	Geom _e	Geom _p					
	F(x)	F ⁻¹ (p)	f'(x)	f''(x)	GAP					
		FS?	FS?C	FS?F	FS?S					
4 User input	Press the corresponding <i>softkey</i>									
	e.g. for FS?S									
	Your WP 43S executes or inserts the command, recalls the constant, or inserts the letter selected.									
Result										
(in this example after specifying the <i>flag</i> number):										
true			1							

At the bottom line, this means that ...

- any function provided can be called by **f CATALOG FCNS** + 4 keystrokes maximum if you know its first two characters (i.e. ≤ 7 keystrokes for any function out of more than 750);
 - any constant provided can be recalled by **g CNST** + 3 keystrokes maximum if you know its first character;
 - any letter provided can be inserted by **f CATALOG CHARS α INTL** (or in AIM by **g (+)**) + 3 keystrokes maximum.

Further Menus and Their Contents

In the table below, all the *menus* provided for you beyond CATALOG are listed in alphabetical sorting order. For each *menu view*, the row of unshifted softkeys is listed first, then the **f**-shifted, then the **g**-shifted, following reading habits. Note, however, that on the screen of your *WP 43S* the order of these three rows is reverted with the unshifted row of each *menu view* displayed at the bottom (see the pictures above).

Different *views* within one *menu* are separated by a dashed line, *submenus* by a double line.

Menu							Remarks
ADV:	SOLVE	SLVQ	$f'(x)$	Π	Σ	$\int f dx$	see Section 4 of the OM
			$f''(x)$				
$\int f dx$:			ACC	\downarrow Lim	\uparrow Lim	\int	
BITS:	AND	OR	XOR	NOT	MASKL	MASKR	encompasses all the Boole's and bit operations of <i>HP-16C</i> and <i>WP 34S</i>
	NAND	NOR	XNOR		MIRROR	ASR	
	SB	BS?	#B	FB	BC?	CB	
	SL	RL	RLC	RRC	RR	SR	
	LJ					RJ	
	1COMPL	2COMPL	UNSIGN	SIGNMT	LZON	WSIZE	
					LZOFF		
CLK:	DATE	\rightarrow DATE	DATE \rightarrow	WDAY	TIME	$x \rightarrow$ DATE	date and time functions and settings.
	J \rightarrow D	D \rightarrow J		DAY	MONTH	YEAR	
	SETTIM	TDISP	SETDAT	D.MY	Y.MD	M.DY	
	CLK12	CLK24				J/G	
CLR:	CL Σ	CLP	CF	CLMENU	CLSTK	CLX	almost as in <i>HP-42S</i>
	CLREGS	CLPall	CLFall	CLCVAR	CLLCD	CLall	
						RESET	

Menu							Remarks
CNST:							catalog of constants, see pp. 125ff
CPX:	dot	cross	UNITV	x	conj	Re Im	special complex functions and settings
			sign	¶	Re	Im	
	CPXi	CPXj			POLAR	RECT	
DISP:	FIX	SCI	ENG	ALL	ROUND1	ROUND	display settings mostly for reals
	SDL	SDR			RDP	RSD	
	SCIOVR	ENGOVR	MULTx	MULT-	RDX.	RDX,	
	GAP					DSTACK	
	CHINA	EUROPE	INDIA	JAPAN	UK	USA	
EQN:	NEW	EDIT	f''	f'	ʃf	Solver	see Section 4 of the OM
	DELETE						
Solver:							show the names of all variables of the current equation and more (s. the OM, Sect. 4).
ʃf:							
f':							
f'':							
EQ.EDI	←	()	^	:	=	→	Equation Editor
EXP:	x ³	xy	log _x y	lb x	2 ^x	x ²	exponential, logarithmic, and hyperbolic functions
	3√x			ln 1+x	e ^x -1		
	sinh	arsinh	cosh	arcosh	tanh	artanh	
FIN:	%	%MRR	%T	%Σ	%+MG	TVM	financial functions including TVM (see the OM, Section 5)
TVM:	n _{PER}	i%/a	per/a	PV	PMT	FV	
	Begin					End	
FLAGS:	SF	FS?	FF	STATUS	FC?	CF	
	FS?S	FS?C	FS?F	FC?F	FC?S	FC?C	
						CLFall	

Menu							Remarks
INFO:	SSIZE?	MEM?	RM?	SMODE?	WSIZE?	KTYP?	system information
	LocR?	FLASH?	ULP?	NEIGHB	SDIGS?	BATT?	
	WHO?	VERS?	DIM?	$\pm\infty$?	α POS?	α LENG?	
INTS:	A	B	C	D	E	F	extra digits for short integers with bases >10, and integer operations
	IDIV	RMDR	MOD	\times MOD	FLOOR	LCM	
	DBL /	DBLR	DBL \times	$^{\wedge}$ MOD	CEIL	GCD	
I/O:	BEEP	LOAD	LOADP	LOADR	LOADSS	LOADΣ	data exchange incl. the PRINT commands of HP-42S
	QUIET	TONE			RECV	SEND	
	ADV	CHAR	DLAY	LCD	MODE	PROG	
	r	REGS	STK	TAB	USER	WIDTH	
	Σ	#					
LOOP:	DSE	DSZ	DSL	ISE	ISZ	ISG	
	DEC					INC	
MATX:	NEW	$[M]^{-1}$	$ M $	$[M]^T$	SIM EQ	EDIT	matrix operations almost as in HP-42S
	dot	cross	UNITY	DIM	INDEX	EDITN	
	ENORM	VANGLE	STOEL	RCLEL	PUTM	GETM	
	I+	I-	STOIJ	RCLIJ	J-	J+	
	RSUM	RNORM	M.LU	DIM?		R \triangleright R	
	EIGVAL					EIGVEC	
M.EDIT:	\leftarrow	\uparrow	OLD	GOTO	\downarrow	\rightarrow	Matrix Editor as in HP-42S
	INSR		DELR		WRAP	GROW	
M.SIMQ:	Mat A	Mat B				Mat X	solver for an SLE (works like SIMQ in the HP-42S)

Menu							Remarks
MODE:	DEG	RAD	GRAD	MUL π	RECT	POLAR	mode settings (top six almost as in HP-42S), second view for short integers
	FAST	SLOW	RM	QUIET	REALRE	CPXRES	
	DENMAX	DENANY	DENFAC	DENFIX	SSIZE4	SSIZE8	
	1COMPL	2COMPL	UNSIGN	SIGNMT	LZON	WSIZE	
	SETSIG				LZOFF		
MyMenu:							will show up outside of AIM ²⁹
Myα:							will show up in AIM ²⁹
PARTS:	IP	FP	MANT	EXPT	sign	x	overlaps with HP-42S CONVERT
	ROUNDI	ROUND				DECOMP	
				Re	Im	\neq	
PROB:	t:	Φ :	C_{yx}	P_{yx}	F:	χ^2 :	15 probability distributions. Selecting one (e.g. Binom) opens a submenu featuring entries for its PDF (or PMF), CDF, error probability, and quantile function
	Norml:	LgNrm:	Cauch:	Expon:	Logis:	Weibl:	
	Binom:	Geom:	Hyper:	NBin:		Poiss:	
	RAN#	SEED				$\Gamma(x)$	
Binom:	Binom _p	Binom			Binom _e	Binom ⁻¹	
Cauch:	Cauch _p	Cauch			Cauch _e	Cauch ⁻¹	
Expon:	Expon _p	Expon			Expon _e	Expon ⁻¹	
F:	$F_p(x)$	$F(x)$			$F_e(x)$	$F^{-1}(p)$	
Geom:	Geom _p	Geom			Geom _e	Geom ⁻¹	
Hyper:	Hyper _p	Hyper			Hyper _e	Hyper ⁻¹	
LgNrm:	LgNrm _p	LgNrm			LgNrm _e	LgNrm ⁻¹	
Logis:	Logis _p	Logis			Logis _e	Logis ⁻¹	
NBin:	NBin _p	NBin			NBin _e	NBin ⁻¹	

²⁹ ... as long as no other menu is called (see Section 6 of the OM).

Menu							Remarks
Norml:	Norml _p	Norml			Norml _e	Norml ⁻¹	
Poiss:	Poiss _p	Poiss			Poiss _e	Poiss ⁻¹	
t:	t _p (x)	t(x)			t _e (x)	t ⁻¹ (p)	
Weibl:	Weibl _p	Weibl			Weibl _e	Weibl ⁻¹	
Φ:	φ(x)	Φ(x)			Φ _e (x)	Φ ⁻¹ (p)	
χ ² :	χ ² _p (x)	χ ² (x)			χ ² _e (x)	(χ ²) ⁻¹	

P.FN:	INPUT	END	ERR	TICKS	PAUSE	P.FN2	additional programming functions (avoided multi-view here).
	PSTO	PRCL	αOFF	αON	CONST	PUTK	
	R-CLR	R-COPY	R-SORT	R-SWAP	LocR	PopLR	
P.FN2:	MENU	KEYG	KEYX	CLMENU	EXITall	RTN+1	
	SDL	SDR	MSG	NOP	VARMNU	MVAR	
	BACK	CASE	SKIP	AGRAPH	PIXEL	POINT	

STAT:	Σ+	Ȑ	s	σ	s _m	x ²	for sample statistics.
	Σ-	Ȑ _w	s _w	σ _w	s _{mw}	SUM	
	CLΣ	Ȑ _G	ε	ε _p	ε _m	PLOT	
	L.R.	r	s _{xy}	ŷ	ŷ	x ²	for curve fitting and 2d sample statistics.
	OrthoF		cov				
	LinF	ExpF	LogF	PowerF		BestF	
	n	Σx	Σx ²	Σxy	Σy ²	Σy	all the sums necessary for the statistics above.
		Σlnx	Σln ² x	Σlnxy	Σln ² y	Σlny	
	Σx ² y	Σxlny				Σylnx	

STK:	x _z	y _z	z _z	t _z	z	DROP	stack related operations.
						DROPy	
	SSIZE4	SSIZE8					

Menu							Remarks
TEST:	x< ?	x≤ ?	x= ?	x≠ ?	x≥ ?	x> ?	binary test commands.
	INT?	EVEN?	ODD?	PRIME?	LEAP?	FP?	
	ENTRY?	KEY?	LBL?	STRI?	CONVG?	TOP?	
	x=+0?	x=-0?	x≈ ?	MATR?	CPX?	REAL?	
	SPEC?	NaN?		M.SQR?		DBL?	
TRI:	sin	arcsin	cos	arccos	tan	arctan	trigonometric functions
U:	E: °C→°F	P: °F→°C	year→s s→year	F&p:	m: V:	x: A:	unit conversions, see pp. 134ff.
	power ratio → dB	dB → power ratio			field ratio → dB	dB → field ratio	
E:	cal→J	J→cal	Btu→J	J→Btu	Wh→J	J→Wh	
P:	hp _E →W	W→hp _E	hp _{UK} →W	W→hp _{UK}	hp _M →W	W→hp _M	
F&p:	lbf→N	N→lbf	bar→Pa	Pa→bar	psi→Pa	Pa→psi	
	in.Hg → Pa	Pa → in.Hg	torr → Pa	Pa → torr	atm→Pa	Pa→atm	
m:	lb.→kg	kg→lb.	cwt→kg	kg→cwt	oz→kg	kg→oz	units of mass
	stone → kg	kg → stone	short cwt→kg	kg → sh.cwt	tr.oz → kg	kg → tr.oz	
	ton→kg	kg→ton	short ton → kg	kg → short ton	carat → kg	kg → carat	
X:	au→m	m→au	ly→m	m→ly	pc→m	m→pc	
	mi.→m	m→mi.	nmi.→m	m→nmi.	ft.→m	m→ft.	
	in.→m	m→in.			yd.→m	m→yd.	
	fathom → m	m → fathom	point → m	m → point	survey foot _{US} → m	m → survey foot _{US}	

Menu							Remarks
A:	acre → m ²	m ² → acre				acre _{US} → m ²	m ² → acre _{US}
V:	gl _{UK} → m ³	m ³ → gl _{UK}	qt. → m ³	m ³ → qt.	gl _{US} → m ³	m ³ → gl _{US}	units of volume
	floz _{UK} → m ³	m ³ → floz _{UK}	barrel → m ³	m ³ → barrel	floz _{US} → m ³	m ³ → floz _{US}	

X.FN:	AGM	B _n	B _n *	erf	erfc	Orthog	advanced mathematical functions like Beta, Bessel, etc.
FIB	g _d	g _d ⁻¹	I _{xyz}	IΓ _p	IΓ _q		
	J _y (x)	lnβ	lnΓ	max	min	NEXTP	
	SHOW	sinc	w _m	w _p	w ⁻¹	β(x,y)	
	γ _{xy}	Γ _{xy}	ζ(x)	(-1) ^x	→DP	→SP	
Orthog:							
	H _n	L _n	L _{nα}	P _n	T _n	U _n	orthogonal polynomials
	H _{np}						

αINTL:	Ā ā	Á á	Ă ď	À à	Ä ä	Ã ã	[α] catalog of international letters, see here . Most letters in this (sub-) menu will change when case is switched in AIM – note you will see the individual letters displayed in either case only at one time. You will reach each and every letter by three keystrokes maximum since letters can be accessed alphabetically.
Â â	Å å	Æ æ	Ą q	Ć č	Č č		
Ç ç	Đ đ	Ď d'	Ē ē	É é	Ě ě		
	È è	Ë ë	Ê ê	È è	Ë ë	Ę e	
	Ĩ ğ	Ī ī	Í í	Ĩ ī	Ī ī	Ĭ ī	
	Î î	Ï ï	Í í	Î î	Ï ï	Ĺ ĺ	
	Ľ ļ	Ń ñ	Ň ñ	Ń ñ	Ó ó	Ó ó	
	Ӯ ö	Ӱ ö	Ӯ ö	Ӯ ö	Ӱ ö	Ӱ ö	
	Œ œ	Ŕ ŕ	Ŕ ŕ	Ŗ ŗ	Ŗ ŗ	Ŗ ŗ	
	Ş ş	Ŏ ŭ	Ŏ ŭ	Ŏ ŭ	Ӯ ū	Ӯ ū	
	Ӯ ū	Ӯ ū	Ӯ ū	Ӯ ū	Ӯ ū	Ӯ ū	
	Ŵ ŷ	Ӯ ŷ	Ӯ ŷ	Ӯ ŷ	Ӯ ŷ	Ӯ ŷ	
	Ž ž						

Menu							Remarks
αMATH:	<	≤	=	≈	≥	>	[α] for comparison symbols, parentheses, & more mathematical and related symbols. You can reach every character by three keystrokes maximum.
	{	[()]	}	
	x / . ³⁰	÷	∫	∞	∞	∞	
	¬	¬	¬	≠		&	
	✗	✗	✗	✗	✓	✗	
	✗	✗	✗	✗	✗	✗	
	:=	≡	≡	E	C	R	
	⊗	⊗	⊕			⊗	
	±	^	T	-1	ℏ		
α.FN:	x→α	αRL	αRR	αSL	αPOS?	α→x	dedicated functions for <i>alpha strings</i> , plus font browser
					αLENG?		
	FBR						
A...Ω:	A α	á	B β	Γ γ	Δ δ	Ε ε	[α] Greek letters. The keyboard grants direct access to 24 of them. ³¹
	é	z ζ	H η	ή	θ θ	I ι	
	í	í	ī ī	K κ	λ λ	M μ	
	N ν	Ξ ξ	O o	ó	Π π	P ρ	
	Σ σ	ς	T τ	Υ υ	ύ	Ŷ ü	Note the two kinds of lower case Σ. See αINTL for more.
	Ü	Φ φ	X χ	Ψ ψ	Ω ω	ώ	

³⁰ With startup default settings, the multiplication dot is found herein and the multiplication cross is called via in A/M. If MULT· is set, however, this dot is called via and the multiplication cross via **αMATH**.

³¹ Reminder: The Greek alphabet (sic!) goes alpha, beta, gamma, delta, epsilon, zeta, eta, theta, iota, kappa, lambda, my, ny, xi, omicron, pi, rho, sigma, tau, ypsilon, phi chi, psi, omega. Note ancient Greek H and Y are pronounced like Finnish Ä and Y, modern Greek H and Y both like Finnish I.

Menu							Remarks
α•:	!	:	;	'	"	@	[α] for punctuation marks and further special characters.
	¡	¿	#	_	~	\	
	\$	€	%	&	£	¥	
	←	↑	↓	↓	→	↑	
	«	»	¤	⌚	*	*	
	⌚	⌚	⌚	⌚	⌚	*	
4⇒:	4⇒4°	4⇒4ʳ	4⇒4ᵍ		4⇒4"	4⇒4π	angular conversions, cf. pp. 140f.
	4°⇒4	4ʳ⇒4	4ᵍ⇒4		4"⇒4	4π⇒4	
	4°⇒4ʳ	4ʳ⇒4°		4°⇒4"	4"⇒4°		

Constants

Your WP 43S contains a *catalog* of 80 physical, astronomical, and mathematical constants sorted alphabetically:

G	G_0	G_c	g_e	GM_{\oplus}	g_{\oplus}	
c_1	c_2	e	e_E	F_{α}	F_{δ}	
1/2	a	a_0	a_M	a_{\oplus}	c	

Names of astronomical and mathematical constants are printed on colored background in the table starting overleaf. Values of physical constants (including their relative standard deviations in *red print* below) are printed on light background if they are exactly defined or almost exactly known – the darker the background, the less precisely the particular value is known.³²

³² For the physical constants, their precise numeric values (incl. their units) and their relative standard deviations (SD) are from CODATA 2014xxx, copied in September 2015. These are the best values known in the scientific community, agreed on by the national standards institutes worldwide. Note that all of them feature less than 16 significant digits. – *Relative uncertainties* are included in the printed table here though

Now here are the contents of CNST (printing commas as radix marks for better visibility and multiplication dots for space reasons). Formulas are printed if applicable.

Name	Numeric value and rel. SD	Remarks
$\frac{1}{2}$	0,5	Trivial but helpful constant for some iterations.
α	365,242 5 d (<i>per definition</i>)	<i>Gregorian year</i>
a_0	$5,291\,772\,106\,7 \cdot 10^{-11} \text{ m}$ $(2,3 \cdot 10^{-10})$	<i>Bohr radius</i> $a_0 = \alpha / 4\pi R_\infty$
a_M	$3,844 \cdot 10^8 \text{ m}$ $(1 \cdot 10^{-3})$	Semi-major axis of the Moon's orbit around the earth $\approx 1,3$ <i>light seconds</i> .
a_\oplus	$1,495\,979 \cdot 10^{11} \text{ m}$ $(1 \cdot 10^{-6})$	Semi-major axis of the Earth's orbit around the sun. Within the uncertainty stated here, it equals 1 <i>astronomic unit</i> ≈ 499 <i>light seconds</i> .

not contained in CNST. These uncertainties are important for determining the precision of results you obtain using the constants given, through the process of 'error propagation' going back to C. F. Gauß (1777 – 1855). This procedure is essential if your results are to be trustworthy – not only in science (remember each and every scientific result shall include the indication of its uncertainty). Please consult suitable reference (e.g. <http://physics.nist.gov/cgi-bin/cuu/Info/Constants/definitions.html> gives a nice introduction). There is simply no way yardstick measurements can yield results accurate to four decimals.

By the way, the terms *resolution*, *precision*, and *accuracy* are confused frequently in measuring. In a nutshell, *resolution* is the least significant digit a measuring instrument indicates. Using this instrument for measuring the same object under identical conditions multiple times, you get an idea about its *repeatability* (or *precision*); this can be no better than its *resolution* but may be significantly worse – a factor of ten or more may be observed easily in real life. *Accuracy* of a measuring instrument, however, can never be better than its *repeatability*. Since you cannot know anything about a real life object or process any better than you can measure it, these considerations are of fundamental importance. We recommend watching them – in your very own interest.

Name	Numeric value and rel. SD	Remarks
c	$2,997\,924\,58 \cdot 10^8 \text{ m/s}$ <i>(per definition)</i>	Speed of light in vacuum $\approx 300\,000 \frac{\text{km}}{\text{s}} = 300 \frac{\text{km}}{\text{ms}} = 300 \frac{\text{m}}{\mu\text{s}} =$ $30 \frac{\text{cm}}{\text{ns}} = 0,3 \frac{\text{mm}}{\text{ps}}$ etc.
c_1	$3,741\,771\,79 \cdot 10^{-16} \text{ m}^2\text{W}$ <i>($1,2 \cdot 10^{-8}$)</i>	First radiation constant $c_1 = 2\pi h c^2$
c_2	$0,014\,387\,773\,6 \text{ m} \cdot K$ <i>($5,7 \cdot 10^{-7}$)</i>	Second radiation constant $c_2 = hc/k$
e	$1,602\,176\,620\,8 \cdot 10^{-19} \text{ A s}$ <i>($6,1 \cdot 10^{-9}$)</i>	Elementary charge $e = \frac{2}{K_J R_K} = \Phi_0 G_0$
e_E	$2,718\,281\,828\,459\,045\,2\dots$	Euler's e .
e/m_e	$1,758\,820\,024 \cdot 10^{11} \text{ A s/kg}$ <i>($6,2 \cdot 10^{-9}$)</i>	Electron charge to mass ratio
F	$96\,485,332\,89 \text{ A s/mol}$ <i>($6,2 \cdot 10^{-9}$)</i>	Faraday constant $F = e N_A$
F_α	$2,502\,907\,875\,095\,892\,8\dots$	Feigenbaum's α and δ
F_δ	$4,669\,201\,609\,102\,990\,6\dots$	
G	$6,674\,08 \cdot 10^{-11} \text{ m}^3/\text{kg s}^2$ <i>($4,7 \cdot 10^{-5}$)</i>	Newtonian constant of gravitation; also known as γ from other authors. See GM_{\oplus} below for a more precise value.
G_0	$7,748\,091\,731 \cdot 10^{-5} / \Omega$ <i>($2,3 \cdot 10^{-10}$)</i>	Conductance quantum $G_0 = 2e^2/h = 2/R_K = eK_j$
G_C	$0,915\,965\,594\,177\,219\,0\dots$	Catalan's constant
g_e	$-2,002\,319\,304\,361\,82$ <i>($2,6 \cdot 10^{-13}$)</i>	Landé's electron g-factor

Name	Numeric value and rel. SD	Remarks
GM_{\oplus}	$3,986\,004\,418 \cdot 10^{14} \text{ m}^3/\text{s}^2$ $(2,0 \cdot 10^{-9})$	Newtonian constant of gravitation times the Earth's mass with its atmosphere included (according to WGS84 ³³)
g_{\oplus}	$9,806\,65 \text{ m/s}^2$ (per def.)	Standard earth acceleration
\hbar	$6,626\,070\,04 \cdot 10^{-34} \text{ J s}$ $(1,2 \cdot 10^{-8})$	Planck constant
\hbar	$1,054\,571\,8 \cdot 10^{-34} \text{ J s}$ $(1,2 \cdot 10^{-8})$	So-called 'Dirac constant' $\hbar = h/(2\pi)$
k	$1,380\,648\,52 \cdot 10^{-23} \text{ J/K}$ $(5,7 \cdot 10^{-7})$	Boltzmann constant $k = R/N_A$
K_J	$4,835\,978\,525 \cdot 10^{14} \text{ Hz/V}$ $(6,1 \cdot 10^{-9})$	Josephson constant $K_J = 2e/h$
l_{PL}	$1,616\,229 \cdot 10^{-35} \text{ m}$ $(2,3 \cdot 10^{-5})$	Planck length $l_P = \sqrt{\eta G/c^3} = t_p c$
m_e	$9,109\,383\,56 \cdot 10^{-31} \text{ kg}$ $(1,2 \cdot 10^{-8})$	Electron mass $\triangleq 511,00 \text{ keV}$
M_M	$7,349 \cdot 10^{22} \text{ kg}$ $(5 \cdot 10^{-4})$	Mass of the Moon
m_n	$1,674\,927\,471 \cdot 10^{-27} \text{ kg}$ $(1,2 \cdot 10^{-8})$	Neutron mass $\triangleq 939,57 \text{ MeV}$
m_n/m_p	$1,001\,378\,418\,98$ $(5,1 \cdot 10^{-10})$	Neutron to proton mass ratio
m_p	$1,672\,621\,898 \cdot 10^{-27} \text{ kg}$ $(1,2 \cdot 10^{-8})$	Proton mass $\triangleq 938,27 \text{ MeV}$

³³ See http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html

Name	Numeric value and rel. SD	Remarks
M_{PL}	$2,176\,47 \cdot 10^{-8} \text{ kg}$ $(2,3 \cdot 10^{-5})$	Planck mass $M_P = \sqrt{\hbar c/G} \approx 22 \mu\text{g}$
m_p/m_e	$1\,836,152\,673\,89$ $(9,5 \cdot 10^{-11})$	Proton to electron mass ratio
m_u	$1,660\,539\,04 \cdot 10^{-27} \text{ kg}$ $(1,2 \cdot 10^{-8})$	Atomic mass constant $= 10^{-3} \text{ kg}/N_A$
$m_u c^2$	$1,492\,418\,062 \cdot 10^{-10} \text{ J}$ $(1,2 \cdot 10^{-8})$	Energy equivalent of the atomic mass constant $\approx 931,49 \text{ MeV}$
m_μ	$1,883\,531\,594 \cdot 10^{-28} \text{ kg}$ $(2,5 \cdot 10^{-8})$	Muon mass $\triangleq 105,66 \text{ MeV}$
M_\odot	$1,989\,1 \cdot 10^{30} \text{ kg}$ $(5 \cdot 10^{-5})$	Mass of the Sun
M_\oplus	$5,973\,6 \cdot 10^{24} \text{ kg}$ $(5 \cdot 10^{-5})$	Mass of the Earth. See GM_\oplus above for a more precise value.
N_A	$6,022\,140\,857 \cdot 10^{23} / \text{mol}$ $(1,2 \cdot 10^{-8})$	Avogadro's number
NaN	Not a number	See the corresponding entry in Section 5 of the OM.
p_0	101 325 Pa (per def.)	Standard atmospheric pressure
R	$8,314\,459\,8 \frac{\text{J}}{\text{mol K}}$ $(5,7 \cdot 10^{-7})$	Molar gas constant
r_e	$2,817\,940\,322\,7 \cdot 10^{-15} \text{ m}$ $(6,8 \cdot 10^{-10})$	Classical electron radius $r_e = \alpha^2 a_0$
R_K	$25\,812,807\,455\,5 \Omega$ $(2,3 \cdot 10^{-10})$	Von Klitzing constant $R_K = h/e^2$

Name	Numeric value and rel. SD	Remarks
R_M	$1,737\,530 \cdot 10^6$ m $(5 \cdot 10^{-7})$	Mean radius of the Moon
R_∞	$10\,973\,731,568\,508$ / m $(5,9 \cdot 10^{-12})$	Rydberg constant $R_\infty = \frac{\alpha^2 m_e c}{2 h}$
R_\odot	$6,96 \cdot 10^8$ m $(5 \cdot 10^{-3})$	Mean radius of the sun
R_\oplus	$6,371\,010 \cdot 10^6$ m $(5 \cdot 10^{-7})$	Mean radius of the Earth
S_a	$6,378\,137\,0 \cdot 10^6$ m <i>(per definition)</i>	Semi-major axis
S_b	$6,356\,752\,314\,2 \cdot 10^6$ m $(1,6 \cdot 10^{-11})$	Semi-minor axis
S_e^2	$6,694\,379\,990\,14 \cdot 10^{-3}$ $(1,5 \cdot 10^{-12})$	First eccentricity squared
$S_e'^2$	$6,739\,496\,742\,28 \cdot 10^{-3}$ $(1,5 \cdot 10^{-12})$	Second eccentricity squared
S_f^{-1}	298,257 223 563 <i>(per def.)</i>	Flattening parameter
T_0	273,15 K <i>(per def.)</i>	= 0°C, standard temperature
T_P	$1,416\,808 \cdot 10^{32}$ K $(2,3 \cdot 10^{-5})$	Planck temperature $T_P = \frac{c^2}{k} \sqrt{\frac{\hbar c}{G}} = \frac{M_P c^2}{k} = \frac{E_P}{k}$
t_{PL}	$5,391\,16 \cdot 10^{-44}$ s $(2,3 \cdot 10^{-5})$	Planck time $t_P = l_P/c$

Name	Numeric value and rel. SD	Remarks
V_m	$0,022\,413\,962 \text{ } m^3/mol$ $(5,7 \cdot 10^{-7})$	Molar volume of an ideal gas at standard conditions $V_m = \frac{RT_0}{p_0} \approx 22,4 \text{ l/mol}$
Z_0	$376,730\,313\,461\,770\,6\dots \Omega$	Characteristic impedance of vacuum $Z_0 = \mu_0 c$
α	$7,297\,352\,566\,4 \cdot 10^{-3}$ $(2,3 \cdot 10^{-10})$	Fine-structure constant $\alpha = \frac{e^2}{2\varepsilon_0 h c} \approx \frac{1}{137}$
γ	$6,674\,08 \cdot 10^{-11} \text{ } m^3/kg\,s^2$ $(4,7 \cdot 10^{-5})$	Newtonian constant of gravitation; also known as G from other authors. See GM_\oplus below for a more precise value.
γ_{EM}	$0,577\,215\,664\,901\,532\,9\dots$	Euler-Mascheroni constant
γ_p	$2,675\,221\,9 \cdot 10^8 \frac{1}{s \cdot T}$ $(6,9 \cdot 10^{-9})$	Proton gyromagnetic ratio $\gamma_p = 4\pi \frac{\mu_p}{h}$
ε_0	$8,854\,187\,81\dots \cdot 10^{-12} \frac{As}{Vm}$	Electric constant or vacuum permittivity $\varepsilon_0 = 1/\mu_0 c^2$ (note the so-called Coulomb's constant is just $1/4\pi\varepsilon_0 = \mu_0 c^2/4\pi = 10^{-7} c^2 \approx 8,987\,552 \cdot 10^9 V\,m/As$, cf. μ_0).
λ_c	$2,426\,310\,236\,7 \cdot 10^{-12} \text{ m}$ $(4,5 \cdot 10^{-10})$	Compton wavelengths of the electron
λ_{cn}	$1,319\,590\,904\,81 \cdot 10^{-15} \text{ m}$ $(6,7 \cdot 10^{-10})$	$\lambda_c = h/m_e c$, neutron $\lambda_{cn} = h/m_n c$, and proton $\lambda_{cp} = h/m_p c$, respectively
λ_{cp}	$1,321\,409\,853\,96 \cdot 10^{-15} \text{ m}$ $(4,6 \cdot 10^{-10})$	

Name	Numeric value and rel. SD	Remarks
μ_0	$1,256\,637\,061\dots \cdot 10^{-6} \frac{V\,s}{A\,m}$ <i>(per definition)</i>	Magnetic constant or vacuum permeability $\mu_0 := 4\pi \times 10^{-7} V\,s/A\,m$
μ_B	$9,274\,009\,994 \cdot 10^{-24} J/T$ <i>(6,2·10⁻⁹)</i>	Bohr magneton $\mu_B = e\hbar/2m_e$
μ_e	$-9,284\,764\,62 \cdot 10^{-24} J/T$ <i>(6,2·10⁻⁹)</i>	Electron magnetic moment
μ_e/μ_B	$-1,001\,159\,652\,180\,91$ <i>(2,6·10⁻¹³)</i>	Ratio of electron magnetic moment to Bohr's magneton
μ_n	$-9,662\,365 \cdot 10^{-27} J/T$ <i>(2,4·10⁻⁷)</i>	Neutron magnetic moment
μ_p	$1,410\,606\,787\,3 \cdot 10^{-26} J/T$ <i>(6,9·10⁻⁹)</i>	Proton magnetic moment
μ_u	$5,050\,783\,699 \cdot 10^{-27} J/T$ <i>(6,2·10⁻⁹)</i>	Nuclear magneton $\mu_u = e\hbar/2m_p$
μ_μ	$-4,490\,448\,26 \cdot 10^{-26} J/T$ <i>(2,3·10⁻⁸)</i>	Muon magnetic moment
σ_B	$5,670\,367 \cdot 10^{-8} \frac{W}{m^2 K^4}$ <i>(2,3·10⁻⁶)</i>	Stefan-Boltzmann constant $\sigma_B = \frac{2\pi^5 k^4}{15h^3 c^2}$
Φ	$1,618\,033\,988\,749\,894\,8\dots$	Golden ratio $\Phi = \frac{1}{2}(1 + \sqrt{5})$
Φ_0	$2,067\,833\,831 \cdot 10^{-15} V\,s$ <i>(6,1·10⁻⁹)</i>	Magnetic flux quantum $\Phi_0 = \frac{\hbar}{2e}$

Name	Numeric value and rel. SD	Remarks
ω	$7,292\,115 \cdot 10^{-5} \text{ rad/s}$ $(2 \cdot 10^{-8})$	Angular velocity of the Earth according to WGS84 (see footnote 33 on p. 128)
$-\infty$	$-\infty$	Note both are counted as numeric values in your WP 43S.
∞	∞	

Some orders of magnitude found in nature are not stored in CNST. They are known with one or two digits precision only but may be helpful nevertheless:

Radius of an atomic nucleus	$\sim 10^{-15} \text{ m}$
Radius of an atom ³⁴	$\sim 10^{-10} \text{ m}$
Radius of the observable universe	$\approx 45 \times 10^9 \text{ l.y.} \approx 4.3 \times 10^{26} \text{ m}$
Amount of stars in observable universe	$\sim 10^{23}$
Amount of atoms in the Sun	$\sim 10^{57}$
Amount of atoms in observable universe	$\sim 10^{80}$
Baryonic mass in observable universe	$\sim 10^{53} \text{ kg}$

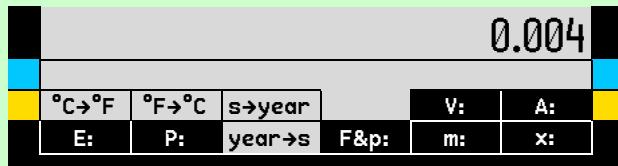
Note these quantities are all well within the range of *SP* numbers on your *WP 43S* (cf. App. B). Precision of physical constants is seldom better known than twelve digits. Please take these facts into account when assessing small differences as well as talking about very small or very large numbers.

³⁴ So the nucleus takes far less than a billionth of the volume of an atom. Electrons are even smaller. Thus, an atom is almost completely empty space. Our world as we know and see it every day is built of atoms. Think about it!

By the way, these facts also give some hand-waving arguments why cancer therapy using heavy ion beams works (and significantly better than using X-rays).

Unit Conversions

Your WP 43S features 14 angular conversions provided in $\underline{\text{A}\rightarrow}$ (see p. 125) and 88 unit conversions in $\underline{\text{U}\rightarrow}$. The structure of $\underline{\text{U}\rightarrow}$ follows various branches as introduced in the OM, Section 5. Its top view looks like this:



with **E:** standing for the *submenu* of energy unit conversions, **P:** for power, **F&p:** for force and pressure, **m:** for mass, **x:** for length, **A:** for area, and **V:** for volume. See p. 122 for further details of the structure.

Conversions contained in $\underline{\text{U}\rightarrow}$ either begin or end in basic SI units. Beyond them and products or powers of these, knowledge of the following *SI derived units* carrying special names may be helpful in your further calculations:

Quantity	Unit	Symbol and formula
Temperature	degree Celsius	$\vartheta[\text{ }^{\circ}\text{C}] = \text{T}[K] - 273.15$
Force	newton	$1 \text{ N} = 1 \text{ kg m/s}^2$
Pressure	pascal	$1 \text{ Pa} = 1 \text{ N/m}^2 = 1 \text{ kg/m s}^2$
Energy	joule	$1 \text{ J} = 1 \text{ N m} = 1 \text{ kg m}^2/\text{s}^2$
Power	watt	$1 \text{ W} = 1 \text{ V A s} = 1 \text{ J/s}$
Electric potential	volt	$1 \text{ V} = 1 \text{ W/A}$
Charge	coulomb	$1 \text{ C} = 1 \text{ A s}$

Quantity	Unit	Symbol and formula
Capacitance	<i>farad</i>	$1 F = 1 \text{ C/V} = 1 \text{ As/V}$
Conductance	<i>siemens</i>	$1 S = 1 \text{ A/V}$
Resistance	<i>ohm</i>	$1 \Omega = 1 \text{ V/A}$
Magnetic flux	<i>weber</i>	$1 \text{ Wb} = 1 \text{ Vs}$
Magnetic flux density	<i>tesla</i>	$1 T = 1 \text{ Wb/m}^2 = 1 \text{ Vs/m}^2$
Inductance	<i>henry</i>	$1 H = 1 \text{ Wb/A} = 1 \text{ Vs/A}$
Frequency	<i>hertz</i>	$1 \text{ Hz} = 1/\text{s}$
Absorbed dose	<i>gray</i>	$1 \text{ Gy} = 1 \text{ J/kg}$

For talking about inputs and results, knowing the symbols and names of the SI prefixes as listed here is beneficial:

Prefix symbol	Name	Factor
h	<i>hecto-</i>	10^2
k	<i>kilo-</i>	10^3
M	<i>mega-</i>	10^6
G	<i>giga-</i>	10^9
T	<i>tera-</i>	10^{12}
P	<i>peta-</i>	10^{15}
E	<i>exa-</i>	10^{18}

Prefix symbol	Name	Factor
d	<i>deci-</i>	10^{-1}
c	<i>centi-</i>	10^{-2}
m	<i>milli-</i>	10^{-3}
μ	<i>micro-</i>	10^{-6}
n	<i>nano-</i>	10^{-9}
p	<i>pico-</i>	10^{-12}
f	<i>femto-</i>	10^{-15}
a	<i>atto-</i>	10^{-18}

All conversions featured in **U→** and **A→** are explained in alphabetical order in the following two tables. Therein, numbers are either exact (printed on white background) or generally rounded to six significant digits; they are for your orientation only (your WP 43S uses more precise values). And commas are used as radix marks for better visibility.

Softkey	Calculation	Remarks	Branch
°C→°F	$\times 1,8 + 32$		U→
°F→°C	$- 32) / 1,8$		
acre → m²	$\times 4\,046,86$	These <i>acres</i> are based on the ' <i>internationalfeet</i> ', see below	
acre_{us} → m²	$\times 4\,046,87$	These <i>acres</i> are based on the ' <i>U.S. surveyfeet</i> ', see below	U→ A:
atm→Pa	$\times 1,013\,25 \times 10^5$	<i>Atmospheres</i>	U→ F&p:
au→m	$\times 1,495\,98 \times 10^{11}$	<i>Astronomic units</i>	U→ x:
barrel → m³	$\times 0,158\,987$	<i>Barrel (U.S.) of oil, abbr. bbl</i>	U→ f V:
bar→Pa	$\times 10^5$	1 mbar = 1 hPa	U→ F&p:
Btu→J	$\times 1\,055,06$	<i>British thermal units</i>	U→ E:
cal→J	$\times 4,186\,8$	<i>Calories</i>	
carat → kg	$\times 0,000\,2$		
cwt→kg	$\times 50,802\,4$	(<i>Long</i>) <i>hundredweight</i> = 112 lbs	U→ m:
dB → field ratio	$10^{R_{dB}/20}$	<i>Decibels</i>	
dB → power ratio	$10^{R_{dB}/10}$		U→ ▼
fathom → m	$\times 1,828\,8$	1 <i>fathom</i> := 6 <i>feet</i>	
ft.→m	$\times 0,304\,8$	The ' <i>international feet</i> ' of 1959 – 1 <i>foot</i> := $1/3$ <i>yard</i>	U→ x:
field ratio→dB	$20 \lg(a_1/a_2)$	Also known as amplitude ratio	U→ ▼

$\text{floz}_{\text{UK}} \rightarrow \text{m}^3$	$\times 2,841\ 31 \times 10^{-5}$	<i>Fluid ounces</i>	
$\text{floz}_{\text{US}} \rightarrow \text{m}^3$	$\times 2,957\ 35 \times 10^{-5}$		U → f V:
$\text{gl}_{\text{UK}} \rightarrow \text{m}^3$	$\times 4,546\ 09 \times 10^{-3}$	<i>Gallons</i>	
$\text{gl}_{\text{US}} \rightarrow \text{m}^3$	$\times 3,785\ 42 \times 10^{-3}$		
$\text{hp}_{\text{E}} \rightarrow \text{W}$	$\times 746$	<i>Electric horsepower</i>	
$\text{hp}_{\text{M}} \rightarrow \text{W}$	$\times 735,498\ 8$	'Metric' horsepower (equivalent to PS in German)	U → P:
$\text{hp}_{\text{UK}} \rightarrow \text{W}$	$\times 745,699\ 9$	<i>British Imperial horsepower</i>	
$\text{in.} \rightarrow \text{m}$	$\times 0,025\ 4$	$1 \text{ inch} := 1/12 \text{ foot}$ and $1 \text{ inch} := 1\ 000 \text{ mil}$	U → x:
$\text{in.Hg} \rightarrow \text{Pa}$	$\times 3\ 386,39$	<i>Inches of mercury</i>	U → F&p:
$\text{J} \rightarrow \text{Btu}$	$/ 1\ 055,06$	<i>Joule</i>	
$\text{J} \rightarrow \text{cal}$	$/ 4,186\ 8$		U → E:
$\text{J} \rightarrow \text{Wh}$	$/ 3\ 600$		
$\text{kg} \rightarrow \text{carat}$	$/ 0,000\ 2$		
$\text{kg} \rightarrow \text{cwt}$	$/ 50,802\ 4$		
$\text{kg} \rightarrow \text{oz}$	$/ 0,028\ 349\ 5$		
$\text{kg} \rightarrow \text{lb.}$	$/ 0,453\ 592$		
$\text{kg} \rightarrow \text{sh.cwt}$	$/ 45,359\ 2$	$1 \text{ t } [(metric) \text{ ton}] = 1\ 000 \text{ kg}$	U → m:
$\text{kg} \rightarrow \text{short ton}$	$/ 907,185$		
$\text{kg} \rightarrow \text{stone}$	$/ 6,350\ 29$		
$\text{kg} \rightarrow \text{ton}$	$/ 1\ 016,05$		
$\text{kg} \rightarrow \text{tr.oz}$	$/ 0,031\ 103\ 5$		
$\text{lbf} \rightarrow \text{N}$	$\times 4,448\ 22$	<i>Pounds force</i>	U → F&p:
$\text{lb.} \rightarrow \text{kg}$	$\times 0,453\ 592$	<i>Pounds;</i> $1 \text{ lb} := 16 \text{ ounces}$	U → m:

$ly \rightarrow m$	$\times 9,460\,73 \times 10^{15}$	<i>Light years</i>	
$m^2 \rightarrow acre$	/ 4 046,86	<i>Square meter;</i> 1 a [are] = 100 m ² , 1 ha [hectare] = 10 000 m ² , 1 km ² = 100 ha = 1 000 000 m ²	
$m^2 \rightarrow acre_{us}$	/ 4 046,87		
$m^3 \rightarrow barrel$	/ 0,158 987		
$m^3 \rightarrow floz_{uk}$	/ 2,841 31 $\times 10^{-5}$	<i>Cubic meter;</i> 1 l [liter] = 10^{-3} m ³ , 1 ml [milliliter] = 1 cm ³	
$m^3 \rightarrow floz_{us}$	/ 2,957 35 $\times 10^{-5}$		
$m^3 \rightarrow gl_{uk}$	/ 4,546 09 $\times 10^{-3}$		
$m^3 \rightarrow gl_{us}$	/ 3,785 42 $\times 10^{-3}$		
$m^3 \rightarrow qt.$	/ 1,1365 $\times 10^{-3}$		
$mi. \rightarrow m$	$\times 1\,609,344$	<i>1 mile = 1 760 yards</i>	
$m \rightarrow au$	/ 1,495 98 $\times 10^{11}$		
$m \rightarrow fathom$	/ 1,828 8		
$m \rightarrow ft.$	/ 0,304 8		
$m \rightarrow in.$	/ 0,025 4		
$m \rightarrow ly$	/ 9,460 73 $\times 10^{15}$		
$m \rightarrow mi.$	/ 1 609,344		
$m \rightarrow nmi.$	/ 1 852		
$m \rightarrow pc$	/ 3,085 68 $\times 10^{16}$		
$m \rightarrow point$	/ 352,778 $\times 10^{-6}$		
$m \rightarrow survey\ foot_{us}$	/ 0,304 801		
$m \rightarrow yd.$	/ 0,914 4		
$nmi. \rightarrow m$	$\times 1\,852$	<i>Nautical miles</i>	
$N \rightarrow lbf$	/ 4,448 22	<i>Newton</i>	

oz→kg	$\times 0,028\ 349\ 5$	1 ounce := $1/16$ pound	[U→ m:]
Pa→atm	$/1,013\ 25 \times 10^5$		
Pa→bar	$/10^5$		
Pa → in.Hg	$/3\ 386,39$	Pascal; 1 mbar = 1 hPa	[U→ F&p:]
Pa→psi	$/6\ 894,76$		
Pa → torr	$/133,322$		
pc→m	$\times 3,085\ 68 \times 10^{16}$	Parsec	[U→ x:]
point → m	$\times 352,778 \times 10^{-6}$	(Typographical) point	
power ratio →dB	$10 \lg(p_1/p_2)$		[U→ ▼]
psi→Pa	$\times 6\ 894,76$	Pounds per square inch	[U→ F&p:]
qt.→m³	$\times 1,1365 \times 10^{-3}$	(Imperial) quart	[U→ V:]
short cwt → kg	$\times 45,359\ 2$	1 short hundredweight = 100 lbs	
short ton → kg	$\times 907,185$	1 short ton = 2 000 lbs	[U→ m:]
stone → kg	$\times 6,350\ 29$		
survey foot_{US} → m	$\times 0,304\ 801$	1 U.S. survey foot := $\frac{1200}{3937}$ m	[U→ x:]
s→year	$/31\ 556\ 952$		[U→]
ton→kg	$\times 1\ 016,05$	1 Imperial ton = 200 (long) cwt = 2 240 lbs	[U→ m:]
torr → Pa	$\times 133,322$	1 torr = 1 mm Hg	[U→ F&p:]
tr.oz → kg	$\times 0,031\ 103\ 5$	Troy ounces	[U→ m:]
Wh→J	$\times 3\ 600$	Watt-hours	[U→ E:]

$W \rightarrow hp_E$	/ 746	Watt	P:
$W \rightarrow hp_M$	/ 735,498 8		
$W \rightarrow hp_{UK}$	/ 745,699 9		
$yd. \rightarrow m$	$\times 0,914\ 4$	1 yard := 3 feet and 2 yards := 1 fathom	x:
$year \rightarrow s$	$\times 31\ 556\ 952$	$= 365,242\ 5 \times 24 \times 60^2$	

...	Remarks
$4^\circ \rightarrow 4$ (DEG→)	Takes an integer or real ³⁵ x as an angular input in <i>decimal degrees</i> and converts it to an <i>angle</i> in ...
$4^\circ \rightarrow 4^r$ (D→R)	... the current ADM.
$4^\circ \rightarrow 4''$ (D→D.MS)	... radians (equaling the old command D→R). ³⁶
$4^g \rightarrow 4$ (GRAD→)	... sexagesimal degrees (corresponding to the old command H→H.MS).
$4^r \rightarrow 4$ (RAD→)	Takes an integer or real ³⁵ x as an angular input in <i>grad/gon</i> ...
	... and converts it to an <i>angle</i> in the current ADM.

³⁵ If x is neither integer nor real (i.e. neither data type 1, 2, nor 11), error 24 will be thrown. Conversions of integer values to angles in sexagesimal degrees do not make real sense but are allowed nevertheless.

³⁶ Note that angles given in radians cannot represent full circles (or simple fractions of π like $\pi/2$, $\pi/4$, $\pi/5$, $\pi/8$, $\pi/10$, etc.) exactly but with an accuracy of sixteen digits "only". If you want to avoid rounding errors, multiples of π may be a better choice here. This applies especially to large numeric inputs in trigonometric functions since those will be reduced to values between $-\pi$ and $+\pi$ before calculating as mentioned in Section 2 of the OM.

 ...	Remarks
$\text{4}^{\text{r}} \rightarrow \text{4}^{\circ}$ ($\text{R} \rightarrow \text{D}$)	Takes an integer or real ³⁵ x as an angular input in <i>radians</i> and converts it to an <i>angle</i> in <i>decimal degrees</i> (equaling the old command $\text{R} \rightarrow \text{D}$). ³⁶
$\text{4}\pi \rightarrow \text{4}$ ($\text{MUL}\pi \rightarrow$)	Takes an integer or real ³⁵ x as an angular input in <i>multiples of π</i> and converts it to an <i>angle</i> in the current <i>ADM</i> .
$\text{4}'' \rightarrow \text{4}$ ($\text{D.MS} \rightarrow$)	Takes an integer or real ³⁵ x as an angular input in <i>sexagesimal degrees</i> ...
$\text{4}'' \rightarrow \text{4}^{\circ}$ ($\text{D.MS} \rightarrow \text{D}$)	Takes an integer or real ³⁵ x as an angular input in <i>sexagesimal degrees</i> (formatted dddd.dmmsshh) and converts it to an <i>angle</i> in <i>decimal degrees</i> (corresponding to the old command $\text{H.MS} \rightarrow \text{H}$).
$\text{4} \rightarrow \text{4}^{\circ}$ ($\rightarrow \text{DEG}$)	If x is an integer or real (<i>data type</i> 1, 2, or 11), takes it as an angular input in the current <i>ADM</i> and converts it to an <i>angle</i> in <i>decimal degrees</i> , <i>grad/gon</i> , <i>radians</i> , <i>multiples of π</i> , or <i>sexagesimal degrees</i> , respectively. ³⁶
$\text{4} \rightarrow \text{4}^{\text{g}}$ ($\rightarrow \text{GRAD}$)	
$\text{4} \rightarrow \text{4}^{\text{r}}$ ($\rightarrow \text{RAD}$)	If x is a tagged <i>angle</i> (<i>data type</i> 4), on the other hand, this information is used in conversion (e.g. if $x = 1.5\pi$ then $\rightarrow \text{GRAD}$ will return 300^{g} regardless of current <i>ADM</i>).
$\text{4} \rightarrow \text{4}\pi$ ($\rightarrow \text{MUL}\pi$)	
$\text{4} \rightarrow \text{4}''$ ($\rightarrow \text{D.MS}$)	If x is neither of <i>data type</i> 1, 2, 4, nor 11, error 24 will be thrown.

Angular output is tagged always.

SECTION 3: CALLING AND EXECUTING OPERATIONS

As mentioned at the beginning of *Section 2* and in the *OM*, the number of *items* featured on your *WP 43S* is far too large to fit them on the keyboard. Hence, there are several ways to call such an *item*. You have already learned how to call *items* appearing on the keyboard and in *menus* (including *catalogs*). In *Section 6* of the *OM*, you have seen that you can store *items* in user *menus* and/or assign them to specific locations on your *WP 43S*. In the following you will learn about one more way you can use for calling and executing operations:

- Using **[XEQ]** followed by the name of the operation typed in *AIM*.

Furthermore, we will summarize the functions requiring parameters.

Using XEQ for Executing Operations

Instead of picking an operation from a *menu* or *catalog*, you can also call it using **XEQ** as follows:

1. Press **[XEQ]**.
2. Press **@**. You are in *AIM* thereafter; see *Section 2* of the *OM* for the *virtual keyboard* applying in this mode.
3. Key in the name of the function wanted. Either case will do.
4. Press **[ENTER↑]**. Your input will be checked – if the operation specified exists
 - a. it will be checked for required parameters (cf. overleaf);
 - i. if true, you will be prompted for these parameters; then the function will be executed. End.
 - ii. else the function will be executed. End.
 - b. else error 7 (**No such function**) will be thrown (see App. C). End.

Operations Requiring Trailing Parameters

Many functions require at least one trailing parameter specifying what they shall do precisely. See the OM, Section 1. The following three lists summarize these operations:

Operations requiring one parameter

Operation	Numeric param.	Alpha param.
AGRAPH CONVG? DEC DSE DSL DSZ INC INPUT ISE ISG ISZ KEY? KTyp? PUTK RCL RCLCFG RCLS RCL+ RCL- RCLx RCL/ RCL↑ RCL↓ STO STOCFG STOS STO+ STO- STOx STO/ STO↑ STO↓ t \approx VIEW x \approx x=? x≠? x≈? x<? x≤? x≥? x>? y \approx z \approx αENG? αPOS? αRL αRR αSL α \rightarrow x \blacksquare r	Register number	Variable name
ALL ENG FIX GAP RDP RSD SCI SDL SDR	Number of decimals	
ASR MASKL MASKR RL RLC RR RRC SL SR WSIZE	Number of bits	
BACK CASE SKIP	Number of program steps	
BC? BS? CB FB SB	Bit number	
CF FC? FC?C FC?F FC?S FF FS? FS?C FS?F FS?S SF	Flag number	
CONST	Constant number	
DSTACK	Number of levels	
ERR MSG	Error number	
f'(x) f''(x) GTO LBL LBL? PGMINT PGMSLV XEQ Π Σ		Program label

Operations requiring one parameter

Operation	Numeric param.	Alpha param.
GTO.	Number of program steps	Program label
INDEX MVAR M.DIM M.EDIN SOLVE VARMNU ∫		Variable name
LocR	Number of local <i>registers</i>	
PAUSE ⌂DLAY	Number of ticks	
RM ⌂MODE	Mode number	
SIM_EQ	Number of unknowns	
TDISP	Time format number	
TONE	Tone number	
→INT	Base	
⌂CHAR	Character code	
⌂TAB	Column number	
⌂#	Byte	

Note that for any command XYZ requiring one trailing parameter, you can enter XYZ → ST.X and it will take its parameter from X instead.

Operations requiring two parameters

Operation	First parameter	Second parameter
ASSIGN	<i>Item</i>	Sequence of keystrokes
KEYG KEYX	Key number (1 ... 18)	Program label

Operation requiring four parameters

Operation	First to fourth parameter
⌘	<i>Stack registers</i>

Operations Changing Data Types

Most functions will return data of the same type they operate on. Some functions, however, will change the *data type (DT)* of the lowest *stack register(s)* regardless of the specific input values, as mentioned at various locations in the OM. These operations are collected in the list here:

Input DT	Operation(s)	Output DT	Output registers involved
1	$1/x$ $\sqrt[3]{x}$ AGM cos erf erfc e^x f' f'' gd gd^{-1} IMPFRC $J_y(x)$ LN LNB $LN\Gamma$ LOG_{10} LOG_2 LOG_{xy} MANT POISS... PROFRC sin tan W_m W_p W^{-1} \sqrt{y} $\beta(x,y)$ Γ_{xy} γ_{xy} $\Gamma(x)$ $\Delta\%$ →REAL % %MRR %T %Σ %+MG \sqrt{x} as well as all unit conversions and all orthogonal polynomials	2^{37}	X
	→POL →REC	2	X, Y
	SLVQ	2 or 3	X, Y, Z
	$f'(x)$ $f''(x)$ SOLVE	2	X, Y, Z, T
	ALL ENG FIX SCI	2	X, Y, Z, T (...D)
	all angular conversions	4	X
	→H.MS	5	X
	J→D →DATE	6	X
	x→α	7	X
	M.NEW	8	X
	M.GET M.NEW	9	X
	→INT	10	X

³⁷ The functions printed on yellow background will return *long integers (data type 1)* wherever possible.

Input DT	Operation(s)	Output DT	Output registers involved
2	AND CEIL DATE→ DAY D→J EXPT FLOOR IDIV IDIVR MONTH NAND NEXTP NOR NOT OR ROUNDING SIGN WDAY XNOR XOR YEAR ±∞?	1	X
	DECOMP	1	X, Y
	SLVQ	2 or 3	X, Y, Z
	RE→CX	3	X
	arccos arcsin arctan and all angular conversions	4	X
	→H.MS	5	X
	x→DATE →DATE	6	X
	x→α	7	X
	M.GET M.NEW	8 or 9	X
	→INT	10	X
	→DP	11	X
3	ABS CROSS IM RE x 4	2	X
	CX→RE	2	X, Y
	SLVQ	2 or 3	X, Y, Z
	→DP	12	X
4	cos sin tan	2	X
5	→HR	2	X
6	DAY D→J MONTH WDAY YEAR	1	X
	DATE→	1	X, Y, Z
	→REAL	2	X

Input <i>DT</i>	Operation(s)	Output <i>DT</i>	Output registers involved
8	M.DIM?	1	X, Y
	DET DOT ENORM M	2	X
	$\Sigma +$	2	X, Y, statistics
	VANGLE	4	X
9	M.DIM?	1	X, Y
	ENORM	2	X
	DET DOT M	3	X
	ABS IM RE ROUNDI x	8	X
10	SIGN	1	X
	$e^x \text{ LN } \text{LOG}_{xy} \rightarrow \text{REAL}$	2	X
	$x \rightarrow \alpha$	7	X
11	AND CEIL DATE \rightarrow DAY D \rightarrow J EXPT FLOOR IDIV IDIVR MONTH NEXTP NAND NOR NOT OR ROUNDI SIGN WDAY XNOR XOR YEAR $\pm\infty$?	1	X
	DECOMP	1	X, Y
	$\rightarrow SP$	2	X
	arccos arcsin arctan	4	X
	$\rightarrow H.MS$	5	X
	$x \rightarrow \alpha$	7	X
	$\rightarrow INT$	10	X
	RE \rightarrow CX	12	X
	ROUNDI	1	X
	arccos arcsin arctan $\rightarrow SP$	3	X
12	CROSS x 4	11	X

APPENDIX A: HARDWARE

Overall dimensions: wedge-shaped: 77 mm × 144 mm × 13 mm or 8 mm (see p. 149)

Mass with battery: ~100 g

LCD dimensions: about 58.8 mm × 35.3 mm visible area,
400 × 240 quadratic pixels monochrome (see App. D on pp. 168ff)

Processor: STMicroelectronics STM32L476 incl. RTC (see this [link to st.com](#) for the development board used by SwissMicros) running at 25 MHz on battery power or 80 MHz connected to USB (see below).

Memory: 1 MB *FM*, 128 kB *RAM* (see App. B on pp. 152ff),
8 MB additional *FM* on a QSPI chip;
user *RAM* is some 75 kB, user *FM* is 6 MB.

Power supply: 3 V by one CR2032 coin cell; alternative power supply through USB port; typical average currents drawn for power on and busy: ~4.2 mA; idle: ~0.1 mA; power off: ~3 µA.

Buzzer frequency: ≥ 1 Hz up to > 20 kHz in steps of 1 Hz.

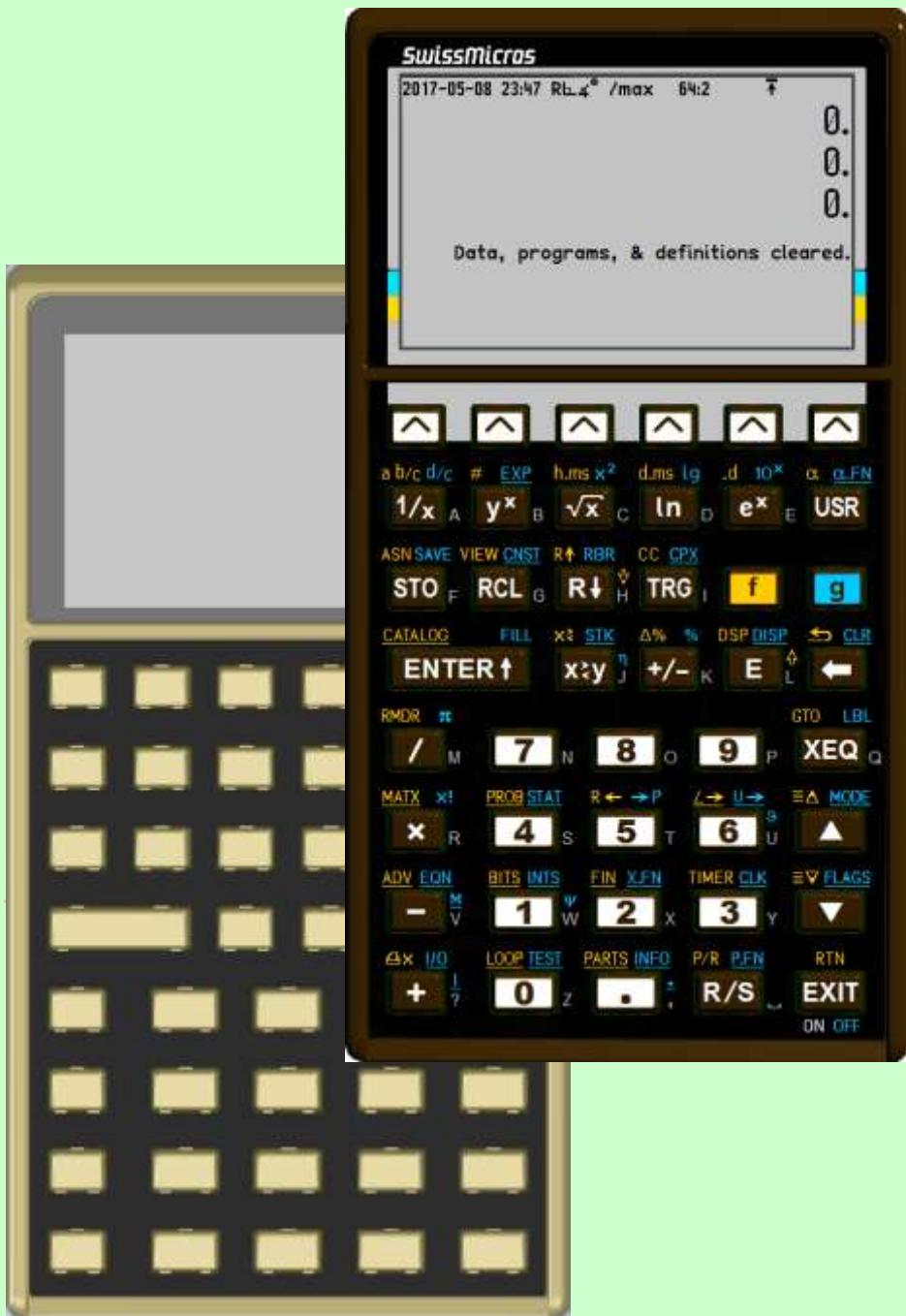
I/O: infrared printer port, standard micro-USB port.

Self-test: initiated by **xxx**

Keyboard overlays: Three short slots are provided in the calculator case on either side for easy fixing overlay sheets with your personal user layouts printed on them. See App. G for more.

Six pictures of the hardware are displayed here and on the two pages following. The default keyboard layout as delivered by Swiss-Micros (bottom right) and the front views on next page are printed approximately to scale.

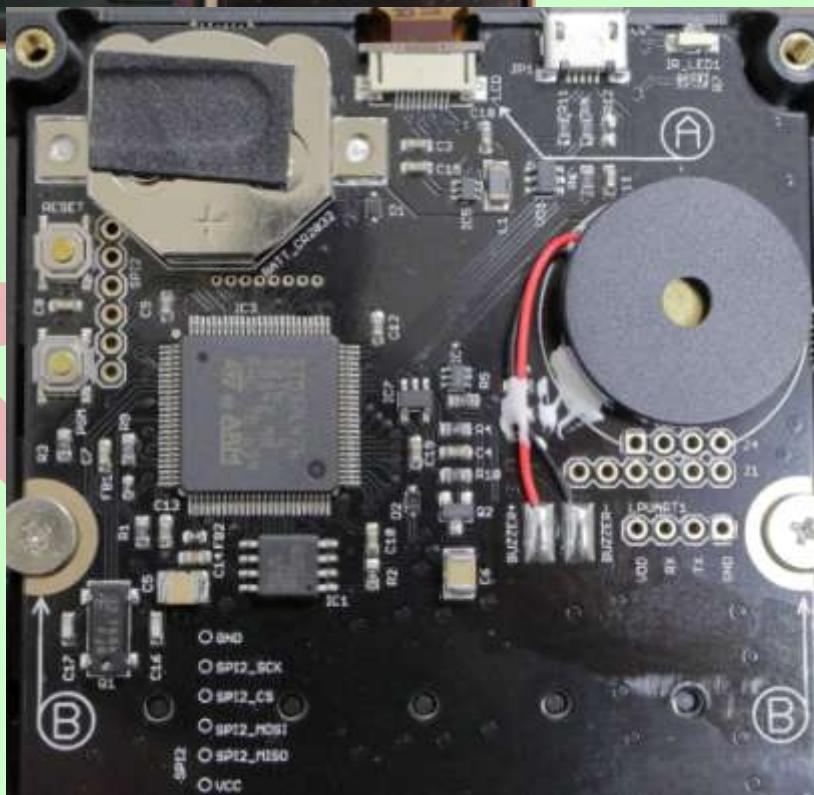






See here the internals, details below. Unfasten two bolts at the top of the calculator backside to get there.

To access the keyboard side of the *PCB* carrying the switching domes, carefully release the *LCD* connection Ⓐ, then unfasten the two Philips bolts Ⓑ pictured left and below. These operations are at your own risk.



APPENDIX B: MEMORY MANAGEMENT

Data Types

There are eleven *data types* you know from *Section 2* of the OM. Some more had to be defined for internal use, e.g.:

- seven-characters strings for all kinds of *labels*, also including names of commands and all other *menu items*; this is the reason why such names are confined to 7 characters,
- system integers in the range of $\pm 2\,147\,483\,648$,
- flag words for storing 128 (i.e. 112 global plus 16 local) user flags and the same amount of system flags,³⁸
- a *data type* of variable length for storing configurations (modes and user assignments, see STOCFG),
- two more for program steps and routines (see *Section 3* of the OM).

Due to data typing, a 2- or 4-byte *header* is specified for each object indicating the *data type* in its first half and the length of data trailing.

Data type number and meaning		Size [bytes]
1	\mathbb{Z} Long integer ³⁹	$\geq 4 + 8 =$ 12
2	\mathbb{R} Real number ⁴⁰	$4 + 8 =$ 12
3	\mathbb{C} Complex number	$4 + 2 \times 8 =$ 20
4	Angle ⁴¹	$4 + 8 =$ 12

³⁸ Up to 256 flags would be possible.

³⁹ This *data type* is for number theory kind of problems. 64 *bits* allow for signed integers up to $2^{63} \approx 9 \times 10^{18}$. Size will be increased in steps of 2 *bytes* if required. Look up App. K for the display limits.

⁴⁰ As in the WP 34S, standard SP reals feature 64 *bits* and 16 digits precision.

⁴¹ An *angle* is stored as a real number, just with a specific header.

Data type number and meaning		Size [bytes]
5	<i>Time</i> ⁴²	$4 + 8 =$ 12
6	<i>Date</i> ⁴³	$4 + 4 =$ 8
7	<i>Alpha string</i> (each character requires 16 bits)	$4 + n \times 2$
8	Real matrix (featuring n rows and m columns)	$4 + n \times m \times 8$
9	Complex matrix (featuring n rows and m columns)	$4 + n \times m \times 16$
10	<i>Short integer</i> ⁴⁴	$\leq 4 + 8 =$ 12
11	Double precision real number	$4 + 2 \times 8 =$ 20
12	Double precision complex number	$4 + 2 \times 16 =$ 36
13	<i>Label</i> (each character requires 16 bits)	$4 + 7 \times 2 =$ 18
14	System integer	$4 + 4 =$ 8
15	System and user flags (128 flags each)	$4 + 2 =$ 6
16	User-created menu	$4 + 18 \times 7 \times 2 =$ 256
17	Predefined menu (featuring n views)	$4 + n \times 18 \times 7 \times 2$
18	<i>Configuration</i> ⁴⁵	M
19	Program step, ⁴⁶ may be stored as <i>alpha string</i> (7)	M
20	Program	M

⁴² A *time* is stored as a real number of *seconds* internally, just with a specific header. A day corresponds to 86 400 s, a year to 31 556 952 s.

⁴³ A *date* is stored as its Julian day number internally. Four *bytes* do for $>10^7$ years.

⁴⁴ This is for HP-16C kind of problems. Most probably, such a storage space will be either $4 + 2$, $4 + 4$, $4 + 6$, or $4 + 8$ bytes long (cf. the footnote for *data type 1* on previous page).

⁴⁵ This size will vary according to the number of user assignments being part of it (see *Section 6* of the OM).

⁴⁶ This size will vary depending on parameters. Exact limits and methods are not decided yet.

Data type number and meaning		Size [bytes]
21	Expression (for members of EQN), may be stored as <i>alpha string</i> (7)	M
22	Directory (proposed by P. 2012-12)	M

Data types 1, 7 - 9, and 17ff are of ‘infinite’ size limited by available memory (M) only. Individual size of each object is fixed though.

As mentioned above, any object of any *data type* will take one storage space only: one *register* or one variable. In consequence, *register* lengths in your *WP 43S* may vary considerably. You do not have to bother – the operating system of your *WP 43S* will take care of all the necessary administration. Thus, the amount of RAM required for data storage is not fixed. Data and programs allocate their memory from the same large pool.

Statistical Summation Registers

Your *WP 43S* features a block of 14 special registers for storing statistical sums (like the *WP 34S* and *WP 31S* before).⁴⁷ The contents of these registers can be recalled using their names; please see pp. 59 and 93.

And like on our calculators before, this block of registers is allocated from the pool of free memory available as soon as the first statistical data are entered via $\Sigma+$ or $\Sigma-$; it is deallocated and the memory is returned to the pool by $\text{CL}\Sigma$.

⁴⁷ These statistical registers do neither overlap nor interfere with any general purpose registers unlike they did on *HP*'s pocket calculators.

Range of Standard (SP) Real Numbers

Your WP 43S can calculate with reals of more than 750 orders of magnitude. Floating point numbers within $10^{-383} \leq |x| < 10^{+385}$ may be entered directly easily.

Within this range, your WP 43S calculates with 16 digits precision (thus, it can display up to 16 digits in *startup default* display format). Results should be accurate within $\pm 1 \times 10^{-15}$ (e.g. **n** **1/x** **2n** **x**) returns a plain 2 for all primes < 500 – just **7** **1/x** **1** **4** **x** returns $2 + 1 \times 10^{-15}$; the same results for 79 and 89, and for 97 a $2 - 1 \times 10^{-15}$ is returned). Results $|x| < 10^{-398}$ are set to zero (see the *ReM, App. B*). For results $|x| \geq 10^{+385}$, error 4 or 5 will appear unless *flag D* is set (see *App. C*).

All these effects are caused by the **internal representation of reals**: *Standard floating point numbers* are stored in eight bytes using an internal format as follows:

- Real zero is stored as integer zero, i.e. all bits cleared.
- The mantissa of a real number (also known as *significand* in this context) is encoded in five groups of three digits. Each such group is packed into 10 bits straight forward, meaning e.g. $555_{10} = 10\ 0010\ 1100_2$ or $999_{10} = 11\ 1110\ 0111_2 = 3E7_{16}$. So the 15 rightmost decimal digits of the *significand* take the least significant 50 bits. Trailing zeroes are omitted, so the *significand* will be right adjusted.
- The most significant (64^{th}) bit takes the sign of the mantissa.
- The remaining 13 bits are used for the exponent and the leftmost digit of the mantissa. Of those 13, the lowest 8 are reserved for the exponent. For the top 5 bits it becomes complicated.⁴⁸ If they read
 - 00ttt, 01ttt, or 10ttt then ttt takes the leftmost digit of the *significand* ($0 - 7_{10}$), and the top two bits will be the most significant bits of the exponent;
 - 11uut then t will be added to 1000_2 and the result (8_{10} or 9_{10}) will become the leftmost digit of the *significand*. If uu reads 00,

⁴⁸ Don't blame us – this part follows the standard IEEE 754.

01, or 10 then these two will be the most significant bits of the exponent. If uu reads 11 instead, there are codes left for encoding special numbers (e.g. infinities).

In total, we get 16 digits for the mantissa and a bit less than 10 bits for the exponent: its maximum is $10\ 1111\ 1111_2$ (i.e. 767_{10}). For reasons becoming obvious below, 398 must be subtracted from the value in this field to get the true exponent of the number represented. The 16 digits of the *significand* allow for a range from 1 to almost 10^{16} .

Rewarding your patience so far, we will show you some illustrative **examples** of the encoding in your WP 43S instead of telling you more theory:

Floating point number	Hexadecimal value stored	Bottom bits in groups of 10	Top 14 bits in binary representation	Stored exponent
1.	22 38 00 00 00 00 00 01		0010 0010 0011 10	398
-1.	A2 38 00 00 00 00 00 01		1010 0010 0011 10	398
111.	22 38 00 00 00 00 00 6F	06F	0010 0010 0011 10	398
111.111	22 2C 00 00 00 01 bC 6F	06F 06F	0010 0010 0010 11	395
-123.000123	A2 20 00 00 07 b0 00 7b	07b 000 07b	1010 0010 0010 00	392
9.99×10^{99}	23 bC 00 00 00 00 03 E7	3E7	0010 0011 1011 11	495
1×10^{-99}	20 AC 00 00 00 00 00 01		0010 0000 1010 11	299
1×10^{-383}	00 3C 00 00 00 00 00 01		0000 0000 0011 11	15
$0.000\ 000\ 000\ 01 \times 10^{-383}$	00 04 00 00 00 00 00 01		0000 0000 0001 00	4

The last number is the smallest that can be entered directly from the keyboard. Dividing it by 10^4 results in 1×10^{-398} , being stored as hexadecimal 1. Divide this by 1.999 999 999 99 and the result will remain 1×10^{-398} in default rounding mode (and in RM 1, 2, 3, and 5, see the command RM). Divide it by 2 instead and the result will become zero.

Let us look at the high end of our numeric range now:

Floating point number	Hexadecimal value stored	Bottom bits in groups of 10	Top 14 bits	Stored exp.
$9.999\ 999\ 999\ 999 \times 10^{384}$	77 FF E7 F9 FE 7F 78 00	9 3E7 3E7 3E7 3dE 000	0111 0111 1111 11	767

This number (featuring 12 times the digit 9) is the maximum which can be keyed in directly. Adding 9.999×10^{372} to it will display 1×10^{385} ...

Floating point number	Hexadecimal value stored	Bottom bits in groups of 10	Top 14 bits	Stored exp.
1×10^{385}	77 FF E7 F9 FE 7F 9F E7	9 3E7 3E7 3E7 3E7 3E7	0111 0111 1111 11	767

... being stored as $9.999\ 999\ 999\ 999\ 999\ 999 \times 10^{384}$. This is the greatest number representable in this format. Thus, the greatest significand possible is $9\ 999\ 999\ 999\ 999\ 999 = 10^{16} - 1$;

All this follows *decimal64* floating point format, though not exactly. Additionally, your WP 43S features three 'special reals':

Floating point 'number'	Hexadecimal value stored	Top byte in binary representation
$+\infty$	78 00 00 00 00 00 00 00	0111 1000
$-\infty$	F8 00 00 00 00 00 00 00	1111 1000
NaN	7C 00 00 00 00 00 00 00	0111 1100

An exponent is not applicable here. These ‘three special reals’ may be legal results of your WP 43S if flag D is set – no error will be thrown then. ‘Not a number’ (NaN) covers outputs exceeding real domain or being undefined – see examples in next chapter. Note that $+\infty$ and $-\infty$ may be also legal inputs while NaN is not.

Special Results

The following monadic functions work with $+\infty$ and $-\infty$, returning numeric results within real domain (with R lit, i.e. flag I cleared):

Input	Operation(s)	Output
$-\infty$	arctan	$-90.^\circ$ or equivalents in other ADM
$-\infty$	e^x , 10^x , 2^x	0.
$-\infty$ or $+\infty$	$1/x$, sinc	0.
$-\infty$ or $+\infty$	x^2	$+\infty$
$+\infty$	\tanh	1.
$+\infty$	arctan	$90.^\circ$ or equivalents
$+\infty$	\ln , e^x , \sqrt{x} , \lg , 10^x , $\text{lb } x$, sinh, cosh	$+\infty$
0. or $-\infty$	$1/x$	$+\infty$
0. or $-\infty$	\ln , \lg , $\text{lb } x$	$-\infty$
1. or $-\infty$	artanh	$+\infty$
$-90^\circ \pm n \times 360^\circ$ or equivalents	\tan	$-\infty$
$90^\circ \pm n \times 360^\circ$ or equivalents	\tan	$+\infty$
...		

The following dyadic functions work alike under these conditions:

y	Input x	Operation	Output
∞	arbitrary $x \neq -\infty$	$+$	∞ ⁴⁹
$-\infty$	arbitrary $x \neq \infty$	$+$	$-\infty$ ⁴⁹
∞	arbitrary $x > 0$	\times	∞ ⁴⁹
$-\infty$	arbitrary $x > 0$	\times	$-\infty$ ⁴⁹
∞	arbitrary $x < 0$	\times	$-\infty$ ⁴⁹
$-\infty$	arbitrary $x < 0$	\times	∞ ⁴⁹
∞	arbitrary $x \neq \infty$	$-$	∞ ⁵⁰
$-\infty$	arbitrary $x \neq -\infty$	$-$	$-\infty$ ⁵⁰
arbitrary $y > 0$	0. or 0	$/$	∞
arbitrary $y < 0$	0. or 0	$/$	$-\infty$
$-\infty$	odd $x > 0$	y^x	$-\infty$
$-\infty$	even $x > 0$	y^x	∞
∞	arbitrary $x > 0$	y^x	∞
0. or 0	$-\infty$	y^x	∞
0. or 0	∞	y^x	0.
arbitrary $y \neq 0$	$-\infty$	y^x	0.
arbitrary $y \neq 0$	∞	y^x	∞

⁴⁹ Swapping x and y will return the same result here.

⁵⁰ Swapping x and y will return the result times -1.

The following functions return **NaN** under these conditions:

Input		Operation(s)	Output
y	x		
$-\infty$ or ∞	$0.$ or 0	y^x	NaN
$-\infty$ or ∞	$-\infty$ or ∞	$/$	NaN
$-\infty$ or ∞	$0.$ or 0	\times	NaN ⁴⁹
$-\infty$	∞	$+$	NaN ⁴⁹
$-\infty$	$-\infty$	$-$	NaN
∞	∞		NaN
$0.$ or 0	$0.$ or 0	y^x , $/$	NaN
$y < 0$	non-integer x	y^x , $\sqrt[3]{y}$	NaN
n/a	$-\infty$ or $+\infty$	\cos , \sin , \tan , arcosh , arsinh , artanh	NaN
n/a	$ \text{Re}(x) > 1$	arccos , arcsin	NaN
n/a	$\text{Re}(x) < 1$	arcosh	NaN
n/a	$\text{Re}(x) \geq 1$	artanh	NaN
n/a	$x < 0$	\sqrt{x} , \ln , lg , $\text{lb } x$	NaN
$y < 0$	$x > 0$ and even	$\sqrt[3]{y}$	NaN

The functions printed on light red background will return **NaN** with **C** lit (i.e. *flag 1* set) as well. The others will change their output when complex results become allowed:

Input	Op.	Output
$-\infty$	\ln	$\infty + i\pi$
$\infty + i\pi$	e^x	$-\infty + i\infty$

Input	Op.	Output
$-\infty$	lg	$\infty + i \times 1.364\ 376\ 353\ 841\ 841 = \infty + i \pi / \ln(10)$
$-\infty$	$\text{lb } x$	$\infty + i \times 1.570\ 796\ 326\ 794\ 896 = \infty + i \pi / 2$
$-\infty$	\sqrt{x}	$0. + i \times \infty$
$0. + i \times \infty$	x^2	$-\infty + i \times 0. - \text{present output: } -\infty + i \times \text{NaN}$

Calculations with Double Precision (*DP*) Real Numbers

Your *WP 43S* uses *single precision* (*data type 2*) in real number calculations per default, wherein 16 digit precision is reached in all calculations. Additionally, you may use *double precision* reals (*data type 11*), allowing for 34 digits instead of 16 (see below).

- ☞ Matrix commands will not work with *DP* reals.
- ☞ *DP* allows for more precise calculations. While some computations will reach high accuracy, we do not warrant 34 digit precision in all calculations with *DP* reals.⁵¹

⁵¹ Not all functions are expanded to *DP*, some stay in *single precision* or merely a little bit more.

The *WP 43S* software is based on the *decNumber* library supporting arbitrary precision *BCD* numbers. As mentioned at some places in the *IOP*, internal computations are carried out with 39 digits. Actually this is the minimum; some modulo calculations are performed with a few hundreds of digits to avoid cancellation (e.g. 2π features 451 digits for proper reduction to the standard range for trigonometric functions).

More elaborate algorithms are coded as *DP* keystroke programs to save flash space (for the cost of execution speed and the loss of a few digits of accuracy in *DP* mode). The internal formats used for storing numbers in your *WP 43S* (as shown just above for *single precision* and below for *DP*) need to be converted back and forth from and to the *decNumber* format. This is a lot of overhead and doesn't come for free in terms of execution speed.

There is a quasi standard to find out about processors and test accuracy of calculators to some extent – compute $\arcsin\{\arccos[\arctan(\tan\{\cos[\sin(9^\circ)]\})]\}$. An

DP reals are stored coarsely following *decimal128* packed coding, though with some exceptions. The lowest 110 bits take the rightmost 33 digits of the *significand*. Going left, a 12 bit exponent field follows, then 5 bits used and coded exactly as in *SP*, and finally the sign bit. The maximum absolute value of the stored exponent is $10\ 1111\ 1111\ 1111_2 = 12287_{10}$. For reasons analogous to those explained on pp. 155ff, 6176 must be subtracted from this value to get the true exponent of the floating point number represented. Thus, *data type 11* could support 34-digit numbers within $10^{-6143} \leq |x| < 10^{+6145}$ (whether such a number range makes sense for real-world problems is questionable, cf. p. 133). Coding works in full analogy to the way described for *SP* in previous chapter.

Even smaller numbers may be entered using a decimal mantissa, but you will lose one digit per factor of 10. The same happens if you divide 10^{-6143} by 10 several times. At 10^{-6176} , only one digit will be left, stored as hexadecimal 1. Divide it by 1.999 999 999 99 and the result will remain 10^{-6176} . Divide it by 2 instead and the result will become zero.

Full precision of a *DP* number x may be displayed by SHOW as a temporary message in small font. SHOW lives in X.FN. Remember not every number may be true to 34 digits. And errors accumulate as explained in footnote 32 on p. 124.

Returning to *SP* via →SP with input exceeding the *SP* number range explained on pp. 155ff will cause 0 or $\pm\infty$ (or an overflow) being displayed instead.

ideal calculator would return exactly 9 without cheating. Real calculators (all computing with a finite number of digits) differ. Your WP 43s returns

- $9,000\ 000\ 000\ 029\ 361 = 9 + 3 \cdot 10^{-11}$ for a *single precision* and
- $8,999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 937\ 535 = 9 - 6 \cdot 10^{-29}$ for a *DP* argument.

If you are interested in the results of other calculators in that test, too, look at <http://www.rskey.org/~mwsebastian/miscpri/results.htm>.

Another simple test discussed recently in the internet: Enter 1,000 000 1 and then execute x^2 just 27 times. Your WP 43s returns

- 674 530,470 539 687 4 for a *single precision* and
- 674 530,470 741 084 559 382 689 184 727 772 2 for a *DP* argument.

The latter is the most precise result known of a pocket calculator so far. Nevertheless, its last 9 digits deviate from the truth here. Take these results into account when assessing small systematic deviations or many decimals.

As mentioned above, some calculations are executed in “*internal high precision*” even for *SP* arguments. “*Internal high precision*” means even more digits than *DP* – it may go up to some hundred digits in special cases.

- ☞ Rounding mode settings (see RM) may affect the results of such calculations!

DRAFT

APPENDIX C: MESSAGES AND ERROR CODES

There are some commands generating *temporary information*, e.g. CORR, DAY, ERR, L.R., MSG, RBR, s, STATUS, VERS, WDAY, \bar{x} , \hat{y} , $\Sigma+$, $\Sigma-$, σ , \rightarrow POL, \rightarrow REC, and the binary test commands. Also three constants will return such information when called (see pp. 117ff).

Furthermore, there are a number of error messages issued by the system. Depending on conditions, the following messages will be displayed (EC means *error code*):

	EC	Explanations, countermeasures and examples
An argument exceeds the function domain	1	{1, 2, 3} An argument exceeds the domain of the mathematical function called. May be caused by roots of negative numbers or logs of $x \leq 0$ (unless CPXRES is set), by 0^0 , $0/0$, $\Gamma(0)$, $\tan(\pm 90^\circ)$ and equivalents, by $\text{artanh}(x)$ for $ \text{Re}(x) \geq 1$, by $\text{arcosh}(x)$ for $\text{Re}(x) < 1$, etc. ⁵²
Bad time or date input	2	{2, 6} Invalid date format or incorrect date or time in input, e.g. month >12, day >31. Will be thrown as soon as the input is closed.
Cannot delete a predefined item	27	Self-explanatory.
Distribution parameter out of valid range	16	{1, 2} A parameter specified in I , J , or K is out of valid range for the distribution function called (e.g. if LGNRM is called with $j < 0$).
Flash memory is full	23	Delete a program from FM to regain space.
Flash memory is write protected	19	Attempt to edit or delete program steps in FM. See PRCL and PSTO.

⁵² Note that e.g. $\tan(90^\circ)$, logs of 0, and $1/0$ are legal operations on {1, 2, 3} if flag **D** is set. See the end of this appendix.

	EC	Explanations, countermeasures and examples
Illegal digit in integer input for this base	9	{10} E.g. 2 in binary or 9 in octal input. Will be thrown as soon as the respective base is entered (i.e. as soon as input is closed).
Illegal input data type for this operation	24	... called. Convert what is necessary. Cf. “operation is undefined in this mode”.
Input is too long	10	Keyboard input is too long for the buffer. (This error is not used currently. Only alpha input is limited presently.)
Invalid or corrupted data	18	Set when there is a checksum error either in FM or as part of a serial download. Also set if a FM segment is otherwise not usable.
I/O error	17	See Section 3 of the OM.
Matrix mismatch	21	{8, 9} <ul style="list-style-type: none"> A matrix isn't square although it should be. Matrix sizes aren't miscible.
No root found	20	{2} The Solver did not converge.
No such function	7	... calling a nonexistent function via [XEQ] [α] ... [ENTER↑] (check for typos!) or running a routine containing a nonprogrammable command.
No such label found	6	Attempt to address an undefined label.
Operation is undefined in this mode	13	Caused e.g. by calling a real number operation in AIM. Cf. “illegal input data type for this operation”.

	EC	Explanations, countermeasures and examples
Out of range	8	<p>{1, 2, 3}</p> <ul style="list-style-type: none"> A number exceeds the valid range. This can be caused by specifying decimals > 16, word size > 64, negative flag numbers, integers $\geq 2^{64}$, hours or degrees > 9 000, invalid dates or times, denominators $\geq 9\,999$, etc. A register or flag address exceeds the valid range of currently allocated registers or flags. May also happen in indirect addressing or when calling nonexistent local addresses. An R-operation (e.g. R-COPY) attempts accessing invalid register addresses.
Overflow at $+\infty$	4	<p>{1, 2, 3, 8, 9, 11, 12} unless flag D is set</p> <ul style="list-style-type: none"> Division of a number >0 by 0. Divergent sum or product or integral. Positive overflow (see p. 155).
Overflow at $-\infty$	5	<p>{1, 2, 3, 8, 9, 11, 12} unless flag D is set</p> <ul style="list-style-type: none"> Division of a number <0 by 0. Divergent sum or product or integral. Negative overflow (see p. 155). Logarithm of (+) 0 (note a logarithm of -0 returns NaN).
Please enter a NEW name	26	Trying to define a new variable or user menu with a name already in use.
RAM is full	11	May be caused by attempts to write too large routines, allocate too many variables, and the like (see pp. 152ff for the space required by different data types). May happen also in program execution due to dynamic allocations (see Section 3 of the OM).

	EC	Explanations, countermeasures and examples
Singular matrix	22	<p>{8, 9}</p> <ul style="list-style-type: none"> Attempt to use a LU decomposed matrix for solving a system of equations. Attempt to invert a matrix which isn't of full rank.
Stack clash	12	STOS or RCLS attempts using <i>registers</i> that would overlap the <i>stack</i> (see Section 1 of the OM). Will happen with e.g. SSIZE8 and STOS 94 .
Too few data points for this statistic	15	{2} A statistical calculation was attempted with too few data, e.g. <i>regression</i> or <i>standard deviation</i> for less than 2 points.
Undefined op-code	3	An instruction with an undefined operation code occurred. Should never happen – but who knows?
Word size is too small	14	{10} Input or <i>register</i> content is too great to be handled by the <i>word size</i> currently set.
	25	Left unused for WP 34S compatibility

Each error message will be displayed in second numeric row and is *temporary information* (see Section 2 of the OM). So or **EXIT** will erase it and allow continuation. Any other key pressed will erase the message as well, but will also execute with the *stack* contents present. Thus, another easy and safe return to the display shown before the error occurred is pressing an arbitrary *prefix* twice.

A final note about **flag D**: If it is set, errors 4 and 5 will not occur at all, and error 1 will happen less frequently, since $\pm\infty$ and NaN are legal results then (see the corresponding entries in CNST on pp. 125ff and the tables on pp. 158ff). E.g., **0 In** will return $-\infty$.

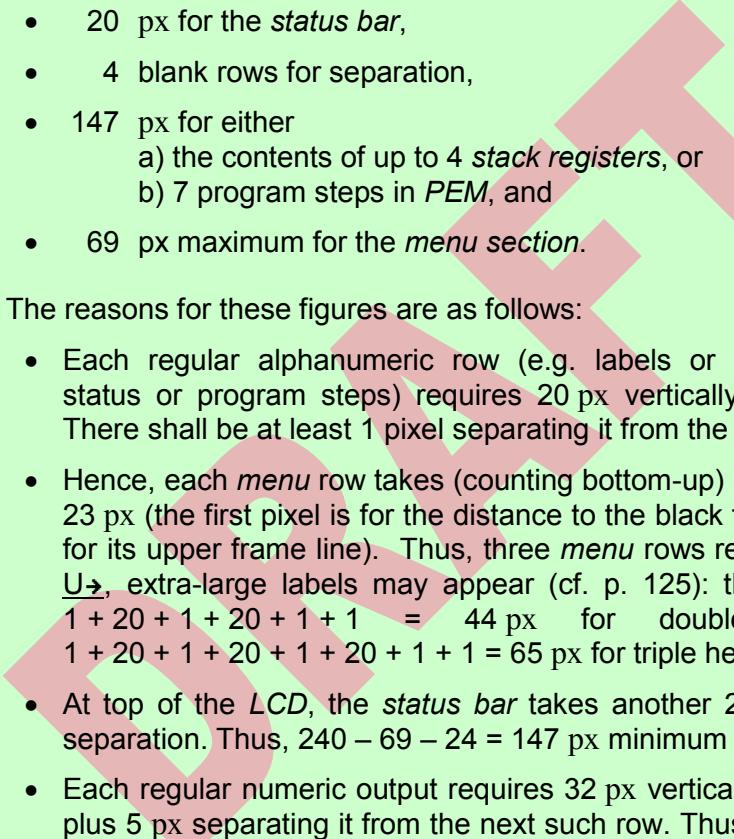
APPENDIX D: DISPLAY INTERNALS

Segmentation

The *LCD* of your *WP 43S* is full dot matrix: 400 px wide and 240 high. Each pixel is 0.147 mm square. Going top down, you will find

- 20 px for the *status bar*,
- 4 blank rows for separation,
- 147 px for either
 - a) the contents of up to 4 *stack registers*, or
 - b) 7 program steps in *PEM*, and
- 69 px maximum for the *menu section*.

The reasons for these figures are as follows:



2017-05-08 23:49	
-12 345 67	
-9.234 56	
-5.678 901	
010 1100 0	
ABCDEF	B
B	
C	

- Each regular alphanumeric row (e.g. labels or status or program steps) requires 20 px vertically (cf. pp. 170f). There shall be at least 1 pixel separating it from the next row.
- Hence, each *menu* row takes (counting bottom-up) $1 + 20 + 1 + 1 = 23$ px (the first pixel is for the distance to the black frame, the latter for its upper frame line). Thus, three *menu* rows require 69 px. In U_→, extra-large labels may appear (cf. p. 125): they will require $1 + 20 + 1 + 20 + 1 + 1 = 44$ px for double height or $1 + 20 + 1 + 20 + 1 + 20 + 1 + 1 = 65$ px for triple height then.
- At top of the *LCD*, the *status bar* takes another 20 px plus 4 for separation. Thus, $240 - 69 - 24 = 147$ px minimum remain.
- Each regular numeric output requires 32 px vertically (cf. pp. 170f) plus 5 px separating it from the next such row. Thus, for 4 rows we need $4 \times 32 + 3 \times 5 = 143$ px. We put the remaining 4 px below this output block.
- If there is a short alpha string in any stack *register*, its ground line is positioned where the ground line of the respective numeric output would have been.

- With an alpha string needing two rows, $20 + 1 + 20 + 5 = 46$ px are required vertically where 37 px are available for one numeric row. So a second numeric row has to go here – up to three stack registers can be displayed only now.

Two subsequent alpha strings of this kind, e.g. after **ENTER↑**, require 92 px. Three regular numeric rows cover 111 px, so here is no further loss. The ground line of the lowest alpha string is positioned where the ground line of the respective numeric output would have been. The other string is positioned in the center of the free space remaining.

- An alpha string needing three rows requires 67 px which are still covered by two regular numeric rows. Two such strings, however, push any other stack register out of the screen. Then string **y** shall have its ground line where the second numeric row would have it.
- In *PEM*, on the other hand, $147 = 7 \times 21$ corresponds to a block of 7 alphanumeric program rows.
- If there will be more space left, we will put it below the numeric block. This is for clear separation from the *menu section* and avoiding vertical output jitter. (gibt's diesen Fall überhaupt noch?)

In the *menu section*, we also have a horizontal structure for the six *softkeys*. We start one pixel off the black frame at left display edge. On the right edge, the characters themselves contain at least one blank column. A minimum of two pixels separate *softkey* labels from each other (one black and one blank). This way we lose a total of $1 + 5 \times 2$ px. The remaining 389 px mean a width of 65 available for 6 *softkey* labels, corresponding to six standard width letters (though letters may extend from 4 to 14 px in small font) which should be centered as good as possible. Note that labels in *menu views* may be not fully displayed if they are wider than 64 px, so labels deviating only in their very last characters may become visually indistinguishable.

Character Sets

For standard numeric output in the rows covering the *stack registers* on the screen, the browser FBR displays the respective characters provided as designed and implemented, sorted according to their hexadecimal codes (most of them following Unicode).

This so-called ‘numeric’ font uses a matrix of up to 16×32 px (variable width, fixed height). Therein, the punctuation space (2008_{16} , 8 px wide) is employed for separating groups of digits in longer numbers – so we follow ISO 31 for an unambiguous numeric display.

In total, six blank characters are provided allowing for any spacing wanted (standard / em / figure, punctuation, four-per-em, and hair space being 16, 8, 4 and 1 px wide).

Most of the elevated characters are for exponents or numerators of fractions. The digits below are for denominators.

Numeric indices (all shown overleaf) are for indicating bases of short integers.

2019-03-24 10:41 R \odot_4° /max 64:2	$\overline{ }$	S L
0 1 2 3 4 5 6 7 8 9 A B C D E F		
0020 ! " # % ' () * + , - . /		
0030 0 1 2 3 4 5 6 7 8 9 : ; < = ?		
0040 ABCDEFGHIJKLMNOP		
0050 PQRSTUWXYZ[] ^ _		
0060 abcdefghijklmno		
Numeric font. Press ↓ or EXIT		1/16

2019-03-24 10:52 R \odot_4° /max 64:2	$\overline{ }$	S L
0 1 2 3 4 5 6 7 8 9 A B C D E F		
0070 p q r s t u v w x y z [] ~ { }		
0080 ± 2 3 μ . 1		
0090 ×		
0100 ÿ		
0110 ÿ		
0120 ÿ		
Numeric font. Press ↑, ↓ or EXIT		2/16

2019-03-24 10:54 R \odot_4° /max 64:2	$\overline{ }$	S L
0 1 2 3 4 5 6 7 8 9 A B C D E F		
0370 Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο		
0380 Π Ρ Σ Τ Υ Φ Χ Ψ Ω		
0390 α β γ δ ε ζ η θ ι κ λ μ ν ξ ο		
0400 π ρ σ τ υ φ χ ψ ω		
Numeric font. Press ↑, ↓ or EXIT		3/16

2019-03-24 10:59 R \odot_4° /max 64:2	$\overline{ }$	S L
0 1 2 3 4 5 6 7 8 9 A B C D E F		
1D60 ×		
2000		
2060		
2070 0 1 -1 4 5 6 7 8 9 + -		
2080 0 1 2 3 4 5 6 7 8 9 + - *		
Numeric font. Press ↑, ↓ or EXIT		4/16

Non-numeric indices are mainly provided for CNST.

Optionally, narrow digits can be used in complex numbers or in matrices or in *short integers* of small base where space may be scarce (see App. K).

This font is also employed for echoing numeric input unless too long. It may be used for echoing command input as well.

The pictures following on the next two pages display

2019-03-24 12:08 R⊕4° /max 64:2 ↑
0 1 2 3 4 5 6 7 8 9 A B C D E F
24C0 K L M N O P Q R S T U V W X Y Z
Numeric font. Press ↑, ↓ or EXIT 8/16

the entire standard (or small) font of alphanumeric characters as designed and implemented.

All characters of this font live in a matrix of up to 14×20 px (variable width, fixed height again). Herein, characters usually start at column one and feature two empty pixel columns at their right side. There are exceptions, however: see e.g. the multiplication dot at B₇₁₆ and the roots in row 2210₁₆:

Characters with codes < 20₁₆ (on picture 9/16) are for control purposes; some of them (4, 10₁₀, 27₁₀) may be useful for printer control (e.g. of HP 82240 A/B).

Many characters are 8 px wide as digits – they will help where a constant character spacing is wanted.

There is a number of super- and subscripts provided. They allow for displaying all the *items* featured on the WP 43S. Arbitrary numeric indices or exponents are possible as well.

2019-03-24	12:38	RØ4°	/max	64:2	4
0	1	2	3	4	5
0140	ø	ł	ń	ø	ň
0150	ø	ł	ę	ŕ	ř
0160	š	ś	ł	ť	ť
0170	ø	ł	ý	ŵ	ŷ
0230	ø	ł	ú	ý	ÿ
0370	ø	ł	ł	ł	ł
0390	č	ă	в	ѓ	ѓ
03A0	п	р	ս	տ	յ
Standard font. Press ↑, ↓ or EXIT					
11/16					

2019-03-24	12:41	R \odot 4°	/max	64:2	↑	S L									
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
03B0	Ü	α	β	γ	δ	ε	ζ	η	ι	κ	λ	μ	ν	ξ	ο
03C0	π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	ι	ύ	ό	ύ	ώ
1D60	[d]	x	[d]												
2000	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]
2010	[20]	[19]	[18]	[13]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]
2020	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]	[20]
2060	[20]	[61]	[20]	[20]	[20]	1	g	δ	μ	ο	⊕	[20]	[20]	[20]	[20]
2070	0	1	-1	E	4	5	6	7	8	9	+	-	[20]	[20]	[20]

Eight blank characters are provided and listed with their hexadecimal addresses and their widths in the table below. Using them, any spacing is feasible:

	Character code	px
Standard space	20_{16}	10
m space	2003_{16}	12
m/3 space	2004_{16}	4
m/4 space	2005_{16}	3
m/6 space	2006_{16}	2
Figure space	2007_{16}	8
Punctuation sp.	2008_{16}	4
Hair space	$200A_{16}$	1

This character set allows for correctly spelling the languages of more than 3.5×10^9 people using either Greek or Latin alphabets. This makes the WP 43S the most versatile calculator available worldwide. If you know of further living languages covered (with one million speakers or more) beyond the ones listed below, please tell us:

2019-03-24	13:20	R \odot 4°	/max	64:2	↑	S L										
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
22A0	88	81	88	88	T	1	88	81	88	88	88	88	88	88	88	
22B0	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	
2310	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	
2390	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	
23A0	88	[L]]	88	88	88	88	88	88	88	88	
2420	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	
2460	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2490	88	10	88	88	88	88	88	88	88	88	88	88	a	b	c	d
Standard font.	Press ↑, ↓ or EXIT	15/16														

2019-03-24 13:22 R \odot ^{4°} /max 64:2 ↑
S L
0 1 2 3 4 5 6 7 8 9 A B C D E F
24A0 e f g h i j k l m n o p q r s T
24B0 u v w x y z A B C D E F G H I J
24C0 K L M N O P Q R S T U V W X Y Z

Afrikaans, Aymara, Bahasa Indonesia, Bahasa Melayu, Basa Jawa, Basa Sunda, Bosanski, Català, Cebuano, Česky, Cymraeg, Dansk, Deutsch, Eesti, Ελληνικά, English, Español, Euskara, Français, Gaeilge, Galego, Hrvatski, Italiano, Kiswahili, Kreyòl, Kurdî, Lietuvių, Magyar, Malagasy, Nāhuatl, Nederlands, Nihongo (Rōmanji), Norsk, Özbek tili, Polski, Português, Quechua, Română, Shqip, Slovenščina, Slovensky, Srpski, Suomi, Svenska, Tagalog, Tatarça, Türkçe, Türkmençe, and Zhōngwén (hànyǔ pīnyīn).

Turn to the OM for examples where these characters are used.⁵³ See here some sample strings in either font, approximately to a common realistic scale:

-1.602 22 $\times 10^{-19}$ C -1,602 22 $\cdot 10^{-19}$ C
-1.602 22 $\times 10^{-19}$ As -1,602 22 $\cdot 10^{-19}$ As

DRAFT

⁵³ Some characters shown in the pictures are not found in any *menu* of your WP 43S. They are not required for any *item* provided so far and may be for future use.



APPENDIX E: COMPARISON TO THE FUNCTION SETS OF HP-42S, HP-16C, HP-21S, AND WP 34S

In the *IOI*, the corresponding functions of vintage *HP* calculators were mentioned under the respective entry of your *WP 43S*. The tables below revert this in a way. The first table shows the functions of the *HP-42S* and the corresponding ones of your *WP 43S* unless they carry identical names and are either both keyboard accessible or both stored in a *catalog* or *menu*. There is an analog table for *HP-16C* functions starting on p. 182, one for the *HP-21S* on p. 184, and another one for the *WP 34S* on p. 186. Functions newly introduced with *WP* calculators are compiled on pp. 190ff.

Functional differences of homonymous commands are covered in the *IOI* (on pp. 12ff).

Corresponding Operations on *HP-42S*

Remarks printed on light grey indicate commands being either default settings or keyboard accessible on your *WP 43S* while you must use a *menu* on the *HP-42S*.

<i>HP-42S</i>	<i>WP 43S</i>	Remarks
ACOSH	arcosh	In <u>EXP</u>
ADV	■ADV	In <u>I/O</u>
AIP	Superfluous	You can easily merge text and numeric data as described in <i>Section 2</i> of the <i>OM</i> .
ALENG	αLENG	In <u>α.FN</u>
ALLΣ	Superfluous	Your <i>WP 43S</i> runs in ALLΣ mode always.
ALPHA	α	See the description of <i>A/M</i> in <i>Sect. 2</i> of the <i>OM</i> .
AOFF	αOFF	In <u>α.FN</u>
AON	αON	

HP-42S	WP 43S	Remarks
ARCL	Superfluous	You can easily merge text and numeric data as described in <i>Section 2</i> of the OM.
AROT	αRL or αRR	In $\underline{\alpha.FN}$
ASHF	αSL	
ASINH	$\alpha rsinh$	In \underline{EXP}
ASTO	Superfluous	Any <i>register</i> or variable can take an <i>alpha string</i> . Simply press STO instead.
ATANH	$\alpha rtanh$	In \underline{EXP}
ATOX	$\alpha \rightarrow x$	In $\underline{\alpha.FN}$
AVIEW	Superfluous	Any <i>register</i> or variable can take an <i>alpha string</i> . Simply press VIEW instead.
BASE	INTS or BITS	
BASE+	Superfluous	Your WP 43S executes these arithmetic commands automatically for integer inputs.
BASE-		
BASE \times		
BASE \div		
BASE $+\!-\!$		
BINM	Superfluous	Press # 2 for converting any closed integer number or integer part in x to binary.
BIT?	BS?	In BITS
BST	$\equiv \Delta$ (\blacktriangle)	Shortcut works if no <i>multi-view menu</i> is open.
CLA	0 STO K	
CLD	Superfluous	Any keystroke will clear <i>temporary information</i> .
CLEAR	CLR	
CLKEYS	n/a	See <i>Section 6</i> of the OM.
CLRG	CLREGS	In CLR
CLST	CLSTK	Press 0 FILL in run mode.
CLV	See remark	Variables are cleared as specified in <i>Section 6</i> of the OM.

HP-42S	WP 43S	Remarks
COMPLEX	CC	You can enter complex numbers directly as well as explained in <i>Section 2</i> of the OM.
CONVERT	L\leftrightarrow & PARTS	
CUSTOM	n/a	You can create as many <i>menus</i> as memory will hold – not just one CUSTOM menu. See <i>Section 6</i> of the OM.
DECM	Superfluous	Any input featuring a D or an E is interpreted as a real (decimal) number.
DEL	n/a	Not featured. Too dangerous, in our opinion.
DELAY	DLAY	In <u>I/O</u>
DELR	M.DELR	In <u>MATX</u>
DET	 M 	
DIM	M.DIM	
DIM?	M.DIM?	
DISP	DISP	
EDIT	M.EDI	In <u>MATX</u>
EDITN	M.EDIN	
FCSTX	\hat{x}	In <u>STAT</u>
FCSTY	\hat{y}	
FLAGS	FLAGS	
FNRM	ENORM	In <u>MATX</u> . Euclid is older than Frobenius.
GAMMA	$\Gamma(x)$	In <u>PROB</u>
GETKEY	KEY?	In <u>P.FN</u>
GETM	M.GET	In <u>MATX</u>
GROW	M.GROW	
HEXM	Superfluous	Press # H for converting any closed integer number or integer part in x to hexadecimal.
H.MS+	Superfluous	Your WP 43S executes these arithmetic commands automatically if sexagesimal times are in x and y and + or - is pressed, respectively.
H.MS-		

HP-42S	WP 43S	Remarks
INSR	M.INSR	In <u>MATX</u>
INTEG	ʃ	In <u>ADV</u>
INVRT	[M]⁻¹	In <u>MATX</u>
KEYASN	Superfluous	Not needed since no CUSTOM menu is featured (see CUSTOM).
LASTx	RCL L	
LBL		Press LBL .
LCLBL	Superfluous	Obsolete since no CUSTOM menu is featured (see CUSTOM). Nevertheless, your WP 43S provides local labels (cf. Section 3 of the OM).
LINΣ	Superfluous	Your WP 43S runs in ALLΣ mode always. Since statistical registers are separate from general purpose registers, there is no benefit in saving statistical registers.
LIST	n/a	Use PROG instead.
LOG	LOG₁₀	Press Ig .
MAN	CF T	Manual print mode is <i>startup default</i> on your WP 43S.
MAT?	MATR?	In <u>TEST</u>
MEAN	Ȑ	In <u>STAT</u>
MODES	MODE	
N!	x!	Press x! .
NEWMAT	M.NEW	In <u>MATX</u>
NORM	n/a	Not featured.
OCTM	Superfluous	Press # 8 for converting any closed integer number or integer part in <i>x</i> to octal.
OLD	M.OLD	In <u>MATX</u>
ON	n/a	Programmable ON is not featured.
PGM.FCN	P.FN	GTO , RTN , and VIEW are on the keyboard.

HP-42S	WP 43S	Remarks
PI	π	Press π .
POSA	α POS	In <u>α.FN</u>
PRA	✉ r K	In <u>I/O</u>
PRGM	P/R	
PRINT	I/O	PRx is on the keyboard.
PRLCD	✉ LCD	In <u>I/O</u>
PROFF	CF T	
PROMPT	Superfluous	Use VIEW , STOP instead.
PRON	SF T	
PRP	✉ PROG	In <u>I/O</u>
PRSTK	✉ STK	
PRUSR	✉ USER	
PRV	✉ r	
PRX	✉ r X	Press ✉ X .
PRΣ	✉ Σ	In <u>I/O</u>
PUTM	M.PUT	In <u>MATX</u>
PWRF	PowerF	In <u>STAT</u>
RAN	RAN#	In <u>PROB</u>
RND	ROUND	In <u>PARTS</u>
RNRM	RNORM	In <u>MATX</u>
ROTXY	RL , RLC , RR , and RRC	In <u>BITS</u>
RTN		Press RTN .
SDEV	s	In <u>STAT</u>
SIZE	Superfluous	There are 100 global general purpose <i>registers</i> always.
SLOPE	L.R.	In <u>STAT</u>
SOLVE	SLV	In <u>ADV</u>

HP-42S	WP 43S	Remarks
SQRT	\sqrt{x}	
SST	$\equiv \nabla$ (∇)	Shortcut works if no <i>multi-view menu</i> is open.
STR?	STRI?	In <u>TEST</u>
TOP.FCN	Superfluous	Obsolete since no top functions are overwritten.
TRACE	SF T	
TRANS	$[M]^T$	In <u>MATX</u>
UVEC	UNITV	In <u>MATX</u> and <u>CPX</u>
VARMENU	VARMNU	Truncated to 6 characters to fit the <i>menu</i> space.
VIEW		Press VIEW .
WMEAN	\bar{x}_w	In <u>STAT</u>
WRAP	M.WRAP	In <u>MATX</u>
XTOA	$x \rightarrow a$	The conversion is done in X.
X<>		Press x<> .
X<0?, X<Y?	$x < ?$	
X≤0?, X≤Y?	$x \leq ?$	
X=0?, X=Y?	$x = ?$	In <u>TEST</u>
X≠0?, X≠Y?	$x \neq ?$	
X≥0?, X≥Y?	$x \geq ?$	
X>0?, X>Y?	$x > ?$	
YINT	L.R.	In <u>STAT</u>
y^x		Press y^x .
ΣREG	Superfluous	There are 100 global general purpose <i>registers</i> always.
ΣREG?		
→DEC	→INT 10	Press # 1 0 or # D
→HR		Press .d ... for closed input.
→H.MS		Press h.ms
→OCT	→INT 8	Press # 8

HP-42S	WP 43S	Remarks
→POL		Press [→P] .
→REC		Press [R↔] .
%CH	Δ%	Press [Δ%] .

Corresponding Operations on *HP-16C*

The table for the functions of the *HP-16C* is sorted following their appearance on its keyboard, starting top left. As for the *HP-42S*, only functions carrying different names on both calculators are listed.

HP-16C	WP 43S	Remarks
RL , RLn	RL	
RR , RRn	RR	In BITS
RLC , RLCn	RLC	
RRC , RRCn	RRC	
DBL÷	DBL/	In INTS
x^{>}(i)		
x^{>}I		Any register may be used for indirection.
SHOW HEX		
SHOW DEC		
SHOW OCT		
SHOW BIN		
B?	BS?	In BITS
GSB	XEQ	
HEX	# [H]	
DEC	# [D]	
OCT	# [8]	
BIN	# [2]	

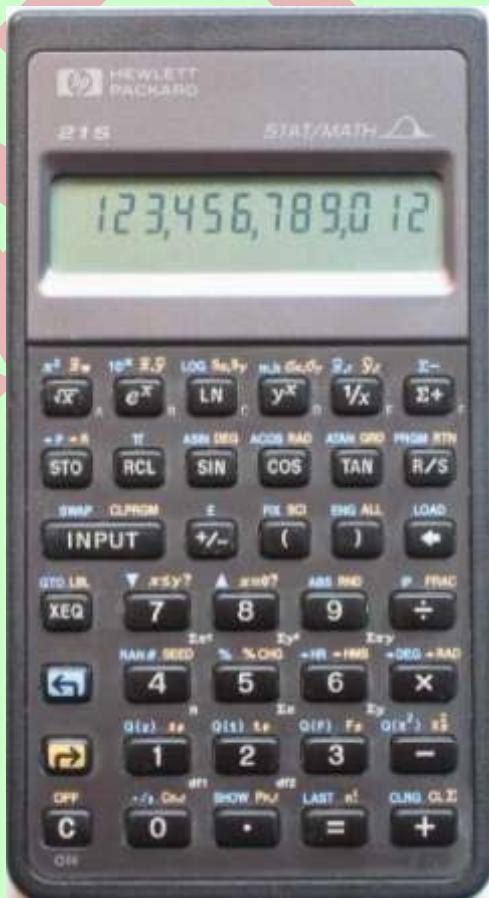
HP-16C	WP 43S	Remarks
SF 3 , CF 3	LZ0N, LZ0FF	In <u>DISPL</u> . Control display of leading zeros.
SF 4 , CF 4	SF C, CF C	Carry.
SF 5 , CF 5	SF B, CF B	Overflow. SF and CF live in <u>FLAGS</u> .
F?	FS?	In <u>FLAGS</u>
(i)	Superfluous	Any <i>register</i> may be used for indirection.
I		
CLEAR PRGM	CLP	In <u>CLR</u> . Note here is also CLPALL.
CLEAR REG	CLREGS	In <u>CLR</u>
CLEAR PREFIX	Superfluous	See Section 2 of the OM.
WINDOW	Superfluous	64 <i>bits</i> can be displayed in one row.
SET COMPL 1S	1COMPL	In <u>MODE</u> and <u>BITS</u> . Note here is also SIGNMT.
SET COMPL 2S	2COMPL	
SET COMPL UNSGN	UNSIGN	
SST	≡▼ (▼)	▼ works if no <i>multi-view menu</i> is open.
BSP	►	
BST	≡▲ (▲)	▲ works if no <i>multi-view menu</i> is open.
x≤y	x≤ ?	In <u>TEST</u> . Note more tests are covered here.
x<0	x< ?	
x>y	x> ?	
x>0		
FLOAT	FIX	In <u>DISP</u>
MEM	STATUS	In <u>FLAGS</u>
CHS	[+/-]	
◀	Superfluous	64 <i>bits</i> can be displayed in one row.
▶		
LSTX	RCL L	
x≠y	x≠ ?	In <u>TEST</u> . Note more tests are covered here.
x≠0		

HP-16C	WP 43S	Remarks
x=y	x= ?	
x=0		

Corresponding Operations on HP-21S

The table for the functions of *HP-21S* (starting overleaf) follows the same rules as the one for *HP-16C*. The *HP-21S*, however, is an algebraic calculator; hence its keys **INPUT**, **(**, **)**, and **=** have no direct equivalent on your *WP 43S*.

Consult the *HP-21S OM* for additional information about the four most important continuous statistical distributions and their applications.



HP-21S	WP 43S	Remarks
\bar{x}_w	\bar{x}_w	In <u>STAT</u>
\bar{x}, \bar{y}	\bar{x}	
$S_{x,y}$	S	
$m.b$	L.R.	
σ_x, σ_y	σ	
$\hat{x}.r$	r, \hat{x}	
$\hat{y}.r$	r, \hat{y}	
PRGM	P/R	
SWAP	$x \gtrless y$	
CLPRGM	CLP	In <u>CLR</u>
LOAD	n/a	Loads predefined programs in the <i>HP-21S</i> . Also your <i>WP 43S</i> features a command called LOAD but this recalls data from backup.
ABS	$ x $	In <u>PARTS</u>
RND	ROUND	
Frac	FP	
SEED	SEED	In <u>PROB</u>
%CHG	$\Delta \%$	
Q(z)	$\Phi_e(x)$	In <u>submenus of PROB</u> . Note the implementation of <u>inverse</u> distribution functions deviates on both calculators: your <i>WP 43S</i> calculates with the probability P while the <i>HP-21S</i> calculates with the error probability $Q = 1 - P$ as input (cf. the OM, Section 2). Labeling these functions on the <i>HP-21S</i> using the letter p may add some confusion.
z_p	$\Phi^{-1}(p)$	
Q(t)	$t_e(x)$	
t_p	$t^{-1}(p)$	
Q(F)	$F_e(x)$	
F_p	$F^{-1}(p)$	
Q(x^2)	$\chi^2_e(x)$	
(x^2)_p	$(\chi^2)^{-1}$	
Cn.r	COMB	In <u>PROB</u>
Pn.r	PERM	In <u>PROB</u>

HP-21S	WP 43S	Remarks
LAST	RCL L	
n!	x!	
CLRG	CLREGS	In CLR

Corresponding Operations on WP 34S

The WP 34S and your WP 43S share over 90% of their function sets. It was our objective that your WP 43S is equal or better than the WP 34S in every aspect. Most of the discrepancies between both calculators are caused by their different displays and by your WP 43S supporting various *data types*. Thus, your WP 43S allows for *softkeys* – the WP 34S can only carry four *hotkeys* instead for hardware reasons. Also dealing with matrices is greatly eased by the large high resolution dot matrix display of your WP 43S; thus some of the elementary matrix commands of the WP 34S are not required anymore by your WP 43S.

Remarks printed on light grey indicate commands being either default settings or obsolete on your WP 43S while you must use them on the WP 34S.

WP 34S	WP 43S	Remarks
ANGLE	4	
Binom _u	Binome	
Cauch _u	Cauch _e	
CL _a	0 [STO] K	Check the OM for the conditions when this register is used.
CONST	CNST	For keyboard space reasons.
DBLOFF	Superfluous	Your WP 43S features DP <i>data types</i> – it does neither need nor feature a DP mode. Use →DP to convert individual data to DP; use →SP to reconvert DP data to SP.
DBLON		

WP 34S	WP 43S	Remarks
Expon _u	Expon _e	
F _u (x)	F _e (x)	
dRCL	Superfluous	Your WP 43S features various <i>data types</i> .
gCLR, gDIM, gDIM?, gFLP, gPIX?, gPLOT, gSET	n/a	The LCD of your WP 43S features 240 pixel rows compared with 6 rows of HP-30b – the graphic paradigm of WP 34S makes no sense on your WP 43S anymore. On the other hand, it was not our objective to design a graphing calculator. Thus, we include just the basic graphic support of the HP-42S (AGRAPH, CLLCD, and PIXEL) plus POINT.
Geom _u	Geom _e	
GTO _a	Superfluous	Use GTO with an appropriate parameter instead.
H.MS+, H.MS-	Superfluous	Your WP 43S features a dedicated <i>data type</i> for times, so + and - suffice for adding or subtracting sexagesimal times, respectively.
INTM?	Superfluous	Your WP 43S features dedicated <i>data types</i> for integers – it does neither need nor feature an integer mode.
iRCL	Superfluous	Your WP 43S features various <i>data types</i> .
I _x	I _{xyz}	This is a triadic function after all.
Lgnrm _u	LgNrm _e	
Logis _u	Logis _e	
MROW+ \times , MROW \times	Superfluous	Obsolete matrix commands.
MROW \Leftarrow	M.R _R	
M+ \times	Superfluous	Obsolete matrix command.
M ⁻¹	[M] ⁻¹	
M-ALL, M-COL, M-DIAG, M-ROW	Superfluous	Obsolete matrix commands.

WP 34S	WP 43S	Remarks
M×	Superfluous	Your WP 43S features two dedicated <i>data types</i> for matrices. Thus you can simply multiply two matrices using X and copy matrices like any other objects.
M.IJ, M.REG	Superfluous	Obsolete matrix commands.
nBITS	#B	
nCOL, nROW	Superfluous	Obsolete matrix commands.
Norml _u	Norml _e	
Poiss _u	Poiss _e	
REALM?	Superfluous	Your WP 43S features a dedicated <i>data type</i> for reals – it does not need a real mode.
REGS, REGS?	Superfluous	The number of global general purpose <i>registers</i> is fixed to 100 on your WP 43S.
SENDA, SENDP, SENR, SENDΣ	SEND	SEND combines all those four commands of the WP 34S.
SEPOFF, SEPON	GAP	
SHOW	RBR	
sRCL	Superfluous	Your WP 43S features various <i>data types</i> .
TRANSP	[M] ^T	
TSOFF	GAP 0	
TSON	GAP 3	
t _u (x)	t _e (x)	
VIEWa, VWa+	Superfluous	Simply use VIEW instead; <i>alpha strings</i> are just another <i>data type</i> . You can combine text and numeric data easily using + as shown in Sect. 2 of the OM.
Weibl _u	Weibl _e	

WP 34S	WP 43S	Remarks
XEQ α	Superfluous	Use XEQ with an appropriate parameter instead.
XTAL?	Superfluous	A quartz crystal is installed by default.
YDOFF, YDON	Superfluous	Your WP 43S displays <i>y</i> whenever possible and wanted.
α DATE, α DAY	Superfluous	You can combine text and numeric data easily using + as shown in <i>Section 2</i> of the <i>OM</i> .
α GTO	Superfluous	Use GTO with an appropriate parameter instead.
α IP, α MONTH	Superfluous	You can combine text and numeric data easily using + as shown in <i>Section 2</i> of the <i>OM</i> .
α RCL, α RC#	Superfluous	Your WP 43S features various <i>data types</i> and ‘knows’ which type is in the <i>register</i> specified. Appending <i>alpha strings</i> is done by + .
α STO	Superfluous	Any <i>register</i> can take an <i>alpha string</i> . Simply press STO instead.
α TIME	Superfluous	You can combine text and numeric data easily using + as shown in <i>Section 2</i> of the <i>OM</i> .
α XEQ	Superfluous	Use XEQ with an appropriate parameter instead.
β	$\beta(x,y)$	
Γ	$\Gamma(x)$	
Δ DAYS	Superfluous	Simply subtract two <i>dates</i> .
ζ	$\zeta(x)$	
$\Phi_u(x)$	$\Phi_e(x)$	
$\chi^2_u(x)$	$\chi^2_e(x)$	
\rightarrow H	\rightarrow HR	
\blacksquare PLOT	n/a	See gCLR.
$\blacksquare^C r_{xy}$	Superfluous	Use $\blacksquare r$ instead. [$\blacksquare x$] is on the keyboard.

WP 34S	WP 43S	Remarks
a , a^+ , $\text{a}^{\text{+}}$	Superfluous	You can combine text and numeric data easily using + as shown in Section 2 of the OM. Then use a . a is on the keyboard.
?	Superfluous	A quartz crystal and the proper firmware for printing are installed by default.

New Commands on your WP 43S

The following table lists the commands and pseudo-commands created for your *WP 43S* (and for preceding *WP* calculators, if applicable), offering new or extended functionality compared to earlier *HP RPN* and algebraic pocket calculators. In total, these are more than 340 operations, excluding the unit conversions and constants provided; 55 are even new or extended compared to earlier *WP* calculators. The commands are printed below as spelled on your *WP 43S*.

Command	WP 43S	WP 31S	WP 34S
2^x AGM	●	—	new
ALL	●	●	extended
AND ASR NOT OR XOR	●	—	extended
BACK CASE SKIP	●	—	new
BATT?	●	●	new
BC? FB	●	—	new
Binom Binom _p (of Binomial distribution)	●	●	new
B_n B_n^* CEIL FLOOR	●	—	new
Cauch Cauch _p Cauch _e Cauch ⁻¹	●	●	new
CLCVAR	new	—	—
CLFall CLK12 CLK24 CLPall CONJ CONVG? COV	●	—	new

Command	WP 43S	WP 31S	WP 34S
CPXi CPXj CX→RE RE→CX	new	—	—
DATE TIME	●	—	(●)
DATE→ DAY MONTH YEAR →DATE	●	—	new
DBL?	modified	—	new
DEC DSL INC ISE	●	—	new
DECOMP	●	●	new
DEG→ D.MS→ GRAD→ RAD→	●	—	new
DROP	●	—	new
DROPy DSTACK	new	—	—
D→J J→D	●	—	new
EIGVAL EIGVEC	new	—	—
ENGOVR SCIOVR	●	—	new
ENTRY?	●	—	new
EQ.DEL EQ.EDI EQ.NEW	new	—	—
erf erfc ERR MSG	●	—	new
EVEN? ODD?	●	—	new
Expon Expon _p Expon _e Expon ⁻¹	●	●	new
EXPT MANT	●	—	new
FAST SLOW	●	—	new
FBR	new	—	—
FC?F FC?S FF FS?F FS?S	●	—	new
FIB	●	—	new
FILL	●	●	new
FLASH? FP?	●	—	new
F _p (x) F(x) (of F distribution)	●	●	new
f' f''	new	—	—
f'(x) f''(x)	extended	—	new

Command	WP 43S	WP 31S	WP 34S
GAP	extended	●	new
GCD LCM	●	●	new
$g_d \ g_d^{-1}$	●	—	new
Geom Geom _p Geom _e Geom ⁻¹	●	—	new
H _n H _{nP} L _n L _{na} P _n T _n U _n	●	—	new
Hyper Hyper _p Hyper _e Hyper ⁻¹	new	—	—
IDIV	●	—	new
IDIVR IM RE	new	—	—
IMPFRC PROFRC	●	●	new
INT? I _{xyz} IΓ _p IΓ _q	●	—	new
J _y (x)	new	—	—
J/G	extended	●	new
KEY? KTYP? LBL? LEAP?	●	—	new
LgNrm LgNrm _p LgNrm _e LgNrm ⁻¹	●	—	new
LNβ LNΓ LOADP LOADR LOADSS LOADΣ LocR LocR? LOG ₂ LOG _{xy}	●	—	new
LOAD SAVE	●	●	new
Logis Logis _p Logis _e Logis ⁻¹	●	●	new
max min MIRROR	●	—	new
MOD	●	●	new
MULT _x MULT _· MULT _π MULT _{π→}	new	—	—
M.LU M.SQR? NAND NaN? NEIGHB NOR	●	—	new
NBin NBin _p NBin _e NBin ⁻¹	new	—	—
NEXTP PRIME?	●	●	new
Norml Norml _p Norml _e Norml ⁻¹	●	●	new
nΣ (callable by name)	●	●	new
OrthoF PLOT POINT	new	—	—
PAUSE	●	—	extended

Command	WP 43S	WP 31S	WP 34S
Poiss Poiss _p Poiss _e Poiss ⁻¹	●	●	new
PopLR PRCL PSTO PUTK	●	—	new
RBR	●	●	new
RCLCFG STOCFG	extended	—	new
RCLS STOS	●	—	new
RCL \uparrow RCL \downarrow ST0 \uparrow ST0 \downarrow	●	—	new
RDP RECV SEND	●	—	new
Re \gtrless Im	new	—	—
RJ	●	—	new
RL RLC RR RRC	●	—	extended
RMDR	●	●	extended
RM RM? ROUNDI RSD RTN+1 R-CLR R-COPY R-SORT R-SWAP	●	—	new
SDIGS? SETSIG	new	—	—
SDL SDR SETCHN SETEUR SETIND SETJPN SETUK SETUSA	●	—	new
SETDAT SETTIM	●	—	(●)
SIGNMT sinc	●	—	new
SL SR	●	—	extended
SLVQ SMODE? SPEC?	●	—	new
S _{mw} S _w	●	●	new
SSIZE4 SSIZE8 SSIZE?	●	●	new
STATUS	extended	—	extended
S _{xy}	●	—	new
TDISP	new	—	—
TICKS	●	—	new
TIMER	●	—	(●)
TOP? ULP?	●	—	new
t _p (x) t(x) (of t distribution)	●	●	new
t _x y _x z _x ζ	●	—	new

Command	WP 43S	WP 31S	WP 34S
undo (UNDO)	●	new	—
VANGLE	new	—	—
VERS? WDAY WHO?	●	●	new
Weibl Weibl _p Weibl _e Weibl ⁻¹	●	●	new
W _m W _p W ⁻¹ WSIZE? X _G XNOR	●	—	new
x→DATE	new	—	—
x< ? x≤ ? x= ? x≠ ? x≥ ? x> ?	●	—	extended
x=+0? x=-0? x≈ ?	●	—	new
Y.MD	●	●	new
αLENG?	extended	—	●
αPOS?	extended	—	—
αRL αRR αSL αSR	●	—	extended
β(x,y) γ _{xy} Γ _{xy} ε ε _m ε _p ζ(x) Π Σ G _w	●	—	new
Σln ² x Σln ² y Σlnx Σlnxy Σlny Σx Σx ² Σx ² y Σxlny Σxy Σy Σylnx Σy ² (callable by names)	●	●	new
Φ(x) φ(x) (of Gaussian distribution)	●	—	new
χ ² _p (x) χ ² (x) (of chi-square distribution)	●	●	new
(-1) ^x xMOD ^MOD	●	—	new
±∞?	new	—	—
→DEG →RAD	●	●	new
→DP →SP	new	—	—
→D.MS →MULπ	new	—	—
→GRAD	●	—	new
→INT →REAL	new	—	—
█ADV █CHAR █r █REGS █TAB █# █MODE	●	—	(new)
█WIDTH	extended	—	(new)
	●	●	new

The statements in parentheses in the rightmost column refer to the *WP 34S* with optional quartz and capacitors installed.

Reference Literature

As mentioned above, some advanced functionality of your *WP 43S* is taken over from previous *HP* calculators. The following vintage *HP* material is recommended as source of in-depth information (as far as calculating, programming, and applications are concerned) about the topics listed, from a calculator point of view. The manuals listed below are entirely contained in a document set distributed by the *Museum of HP Calculators* (see <http://www.hpmuseum.org/cd/cddesc.htm>).

Depending on your educational background and professional qualification, textbooks about various mathematical, scientific, or engineering topics may be helpful in addition. Ensure you know enough about what you compute (and check footnote 56 on p. 212 below as well as the last paragraph on p. 16 of the OM).

Topic	Recommended literature
General calculation examples and applications	All vintage <i>HP</i> calculator manuals can be recommended
Root finding and numeric integration	<i>HP-34C Owner's Handbook and Programming Guide</i> <i>HP-15C Owner's Handbook</i> <i>HP-15C Advanced Functions Handbook</i> <i>HP-42S Programming Examples and Techniques</i>
Statistical distributions and their application	<i>HP-21S Owner's Manual</i>
Financial calculations	<i>HP-17BII+ User's Guide</i>
Manipulating short integers	<i>HP-16C Owner's Handbook</i>

Topic	Recommended literature
Programming	<i>HP-42S Owner's Manual</i> <i>HP-42S Programming Examples and Techniques</i>

DRAFT

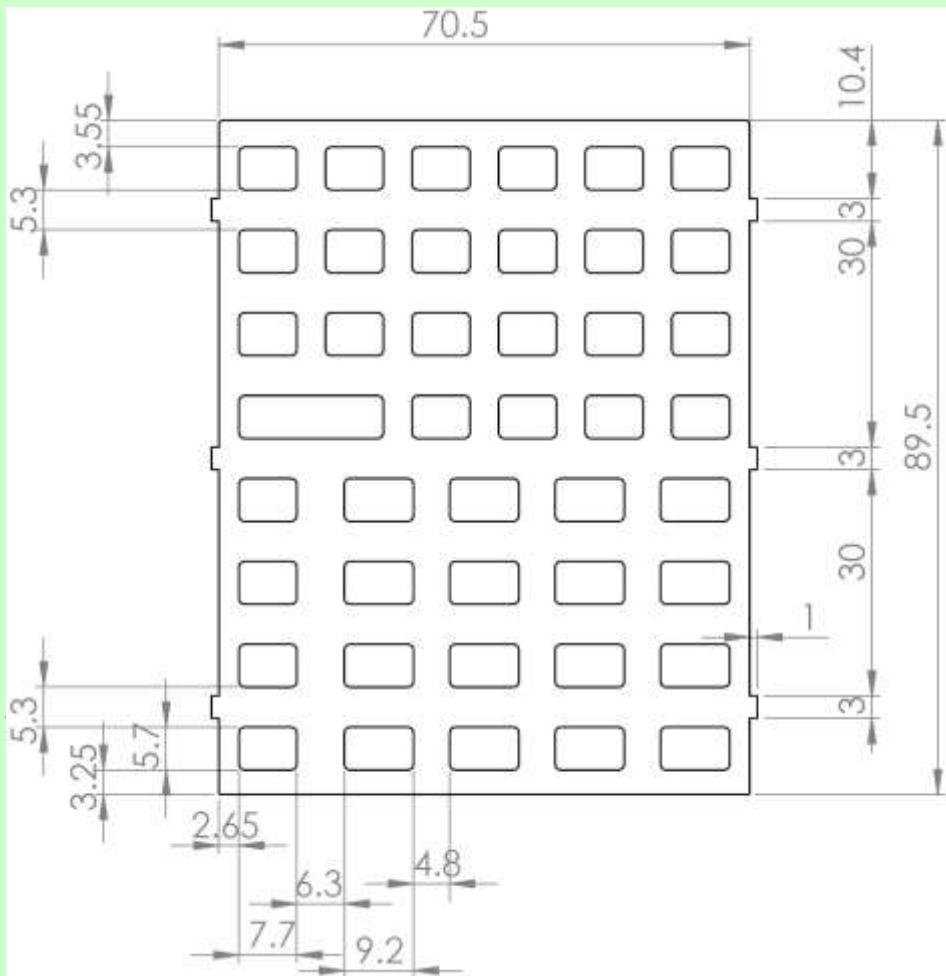
APPENDIX F: FLASHING AND UPDATING YOUR WP 43S

xxx

DRAFT

APPENDIX G: OVERLAYS

See below the drawing for an overlay – all dimensions are given in mm.
Note the overall width is 72.5 mm.



APPENDIX H: TROUBLESHOOTING GUIDE

XXX

DRAFT

APPENDIX I: EMULATING A WP 43S ON YOUR COMPUTER

Xxx

DRAFT

APPENDIX J: ADVANCED MATHEMATICAL FUNCTIONS AND TASKS

Your *WP 43S* contains several operations covering advanced mathematics. Most of them are taken over from *WP 34S*, some are implemented here for the first time on an *RPN* calculator. Find those functions collected here and described in more detail than in the *IOI*, together with a few traditional pocket calculator functions matching the topic.

For reasons explained in *Section 1*, we assume you are able to read and understand mathematical formulas for real domain functions.

Wherever complex numbers may be valid input or output, the command name is printed on light yellow background. Ensure you understand the respective fundamental mathematical concepts; else leave these functions aside. By experience, it is only beneficial to use something you overview and know the background of – otherwise it may even become dangerous for you and your fellow men.

Number Generating Functions

The following are all *monadic* functions except COMB and PERM.

Name	Remarks (see pp. 12ff for general information)
B_n	B_n returns the Bernoulli number for an integer $n > 0$ given in X : $B_n = (-1)^{n+1} \cdot n \cdot \zeta(1 - n)$ B_n^* works with the old definition instead:
B_n^*	See p. 233 for $\zeta(x)$. $B_n^* = 2 \cdot \frac{(2n)!}{(2\pi)^{2n}} \cdot \zeta(2n)$

Name	Remarks (see pp. 12ff for general information)
COMB, PERM	<p>For $y \geq x \geq 0$ and $x, y \in \mathbb{N}$, $C_{y,x} = \binom{y}{x} = \frac{y!}{x!(y-x)!}$ is the number of <i>combinations</i> and $P_{y,x} = \frac{y!}{(y-x)!} = x! \cdot C_{y,x}$ the number of <i>permutations</i> of x and y as explained in the IOP (see pp. 25 and 60, respectively).</p> <p>Note $C_{y,0} = 1$, $C_{y,1} = y$, and $C_{y,2} = \frac{1}{2}y(y-1)$.</p> <p>$C_{y,x}$ applies to the <i>binomial distribution</i> (see p. 203): In a <i>Galton box</i>⁵⁴ (a.k.a. <i>bean machine</i>) featuring y rows of pins and fed with 2^y balls, $C_{y,x}$ is the number of balls expected in column x of that box (start column counting with zero).</p> <p>$P_{y,x} = \frac{\Gamma(y+1)}{\Gamma(y-x+1)}$ and $C_{y,x} = \frac{\Gamma(y+1)}{\Gamma(x+1) \cdot \Gamma(y-x+1)}$ work also for non-integer numbers and in complex domain, too.</p>
FIB	<p>For integers, FIB returns the Fibonacci number f_n with $n = x$. These numbers are defined as $f_0 = 0$, $f_1 = 1$, and $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$. With UNSIGNED, f_{93} is the maximum before an overflow occurs.</p> <p>Else FIB returns the extended Fibonacci number</p> $F_x = \frac{1}{\sqrt{5}} [\Phi^x - \Phi^{-x} \cos(x\pi)]$ <p>for an arbitrary real or complex number x, with $\Phi = \frac{1+\sqrt{5}}{2}$ denoting the golden ratio.</p>

⁵⁴ Translator's note: This is called «Planche de Galton» in French, “Galtonbrett” in German, and “macchina di Galton” in Italian. Note the subtle differences in naming. Galton invented his box in 1889.

Statistical Distributions

Stack-wise, the following are all *monadic* functions, stored in **PROB**. Actually, they feature more parameters though. Those are supplied in the *registers I, J, and K* as applicable and mentioned below.

In the following text, the five discrete distributions are covered first, the continuous ones thereafter. Typical plots are shown for the *PMFs* or *PDFs*.

Binom: *Binomial distribution* with the *number of successes g* in X, the *gross probability of a success p₀* in I and the *sample size n* in J.

Binom_P returns

$$p_B(g; n; p_0) = \binom{n}{g} \cdot p_0^g \cdot (1 - p_0)^{n-g}$$

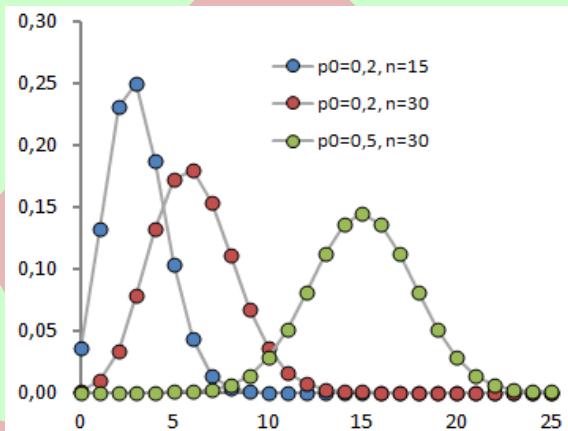
$= C_{n,g} \cdot p_0^g \cdot (1 - p_0)^{n-g}$ (see COMB on p. 202).

Binom returns $F_B(m; n; p_0) = \sum_{g=0}^m p_B(g; n; p_0)$ with the maximum number of successes m in X.

The *binomial distribution* is fundamental for error statistics in industrial sampling, e.g. for designing test plans.

If you want to know, for example, the probability for finding no faulty items in a sample of 15 items drawn from a batch of 300 wherein you expect 3% defective items overall, this will tell you:

.03 [STO] J 15 [STO] K 0 [PROB] g Binom: Binom



returns 0.633 – so the odds are almost two out of three that you will not detect any defect in your sample! ⁵⁵

Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda366i.htm>.

Geom: Geometric distribution:

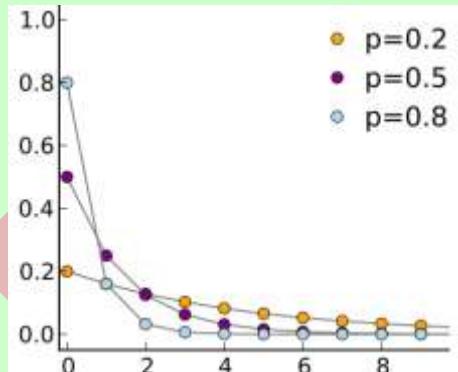
Geom_P returns

$$p_{Ge}(n) = p_0(1 - p_0)^n$$

Geom returns

$$F_{Ge}(m) = 1 - (1 - p_0)^{m+1}$$

being 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 I.



Start reading here for more:

http://en.wikipedia.org/wiki/Geometric_distribution.

Hyper: Hypergeometric distribution with the number of successes g in X, gross probability of a success p_0 in I, sample size n in J, and batch size n_0 in K.

Hyper_P returns $p_H(g; n; p_0; n_0) = \frac{\binom{n_0}{g} \cdot \binom{n_0(1-p_0)}{n-g}}{\binom{n_0}{n}}$ (see COMB on p. 202 for the explanation of the notation).

⁵⁵ The exact result for said probability is 0.626, calculated using the hypergeometric distribution. These results show nicely that two significant digits are a typical accuracy of theoretical statistical statements – frequently the (simplified) statistical model used matches reality no better than that.

While the *binomial distribution* is based on the assumption that each sample part is returned to the batch after drawing and checking, the *hypergeometric distribution* lets you keep your samples out of the batch. This is found more often in real life, but may be neglected in 'large' batches ($n_0 > 10$) and small sample sizes (<10% of n_0).

Start reading here for more:

http://en.wikipedia.org/wiki/Hypergeometric_distribution.

NBin: Negative binomial distribution with the *total number of failures f* (in n draws) in X, the *gross probability of a success in a single draw p₀* in I, and n in J.

$$\text{NBin}_P \text{ returns } p_{NB}(f; n; p_0) = \binom{n-1}{f-1} \cdot p_0^f \cdot (1-p_0)^{n-f} \\ = C_{n-1, f-1} \cdot p_0^f \cdot (1-p_0)^{n-f} \text{ (see COMB on p. 202).}$$

Start reading here for more:

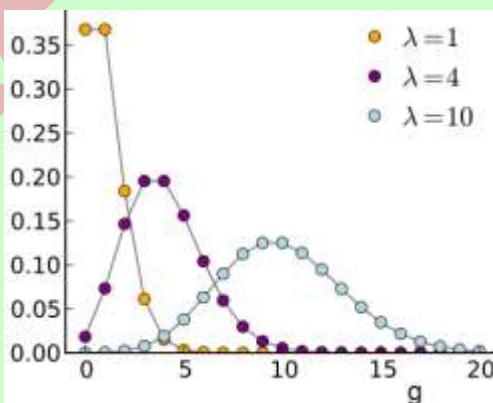
http://en.wikipedia.org/wiki/Negative_binomial_distribution.

Poiss: Poisson distribution with the *number of successes g* in X and the *Poisson parameter λ* in J.

Poiss_P computes

$$p_P(g; \lambda) = \frac{\lambda^g}{g!} e^{-\lambda}$$

and Poiss returns the corresponding *CDF* for the maximum number of successes m in X.

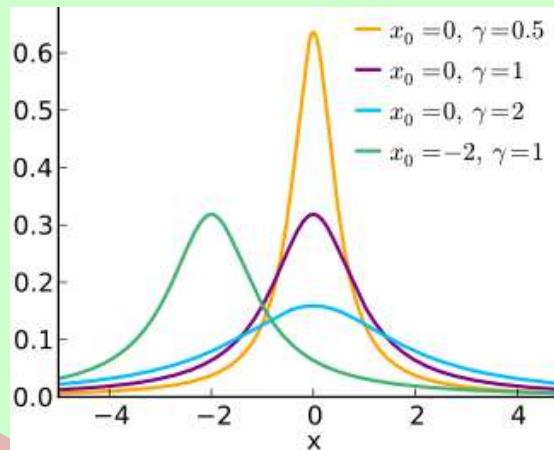


The *Poisson distribution* provides the mathematically simplest model for industrial sampling tests – use $\lambda = np_0$ with the gross error probability p_0 and the sample size n (cf. BINOM). For the example introduced with BINOM above, we get 0.638 here.

Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda366j.htm>.

Cauch: *Cauchy-Lorentz distribution* (also known as *Lorentz* or *Breit-Wigner distribution*) with the *location* x_0 specified in I and the *shape* γ in J.



Cauch_P returns $f_{Ca}(x) = \left\{ \pi\gamma \cdot \left[1 + \left(\frac{x-x_0}{\gamma} \right)^2 \right] \right\}^{-1}$,

Cauch returns $F_{Ca}(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x-x_0}{\gamma}\right)$,

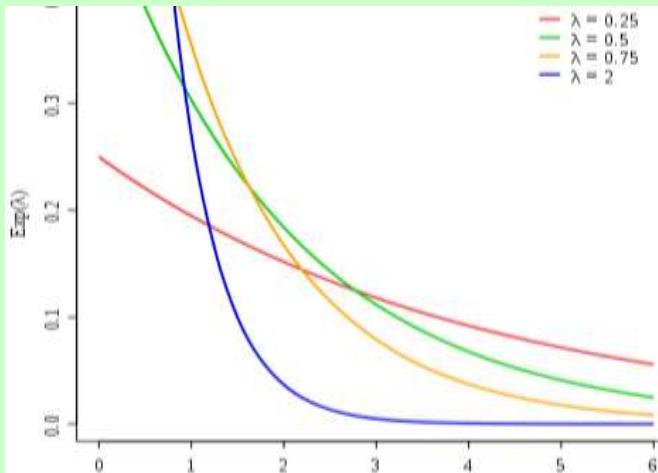
Cauch⁻¹ returns $F_{Ca}^{-1}(p) = x_0 + \gamma \tan\left[\pi \cdot \left(p - \frac{1}{2}\right)\right]$.

This distribution is quite popular in physics. It is a special case of *Student's t distribution*. Start reading here for more:
http://en.wikipedia.org/wiki/Cauchy_distribution.

Expon: *Exponential distribution* with the *rate* λ in I.

Expon_P returns $f_{Ex}(x) = \lambda \cdot e^{-\lambda x}$. See the PDF plotted overleaf.

Expon returns $F_{Ex}(x) = 1 - e^{-\lambda x}$.

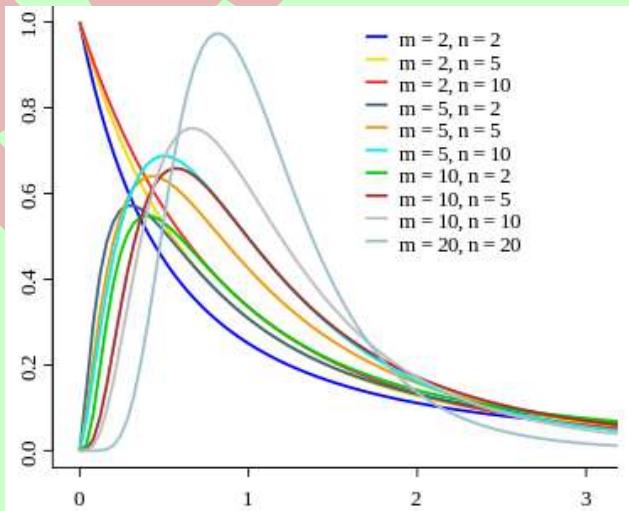


Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda3667.htm>

F(x): Fisher's F distribution with the *degrees of freedom* in **I** and **J**.
It is used e.g. for analyses of variance (ANOVA).

The picture shows the *PDF* plotted for different *degrees of freedom* **m** and **n** corresponding to **i** and **j**.



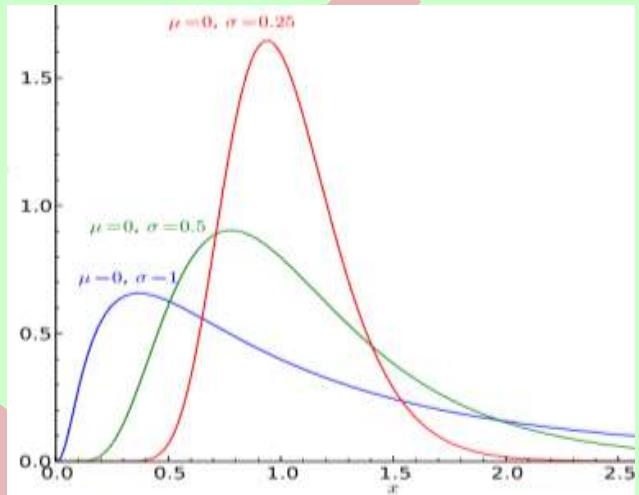
Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda3665.htm>

LgNrm: *Log-normal distribution* with the parameters $\mu = \ln \bar{x}_g$ in **I** and $\sigma = \ln \varepsilon$ in **J** (see some PDF plots below).

LgNrm_{P} returns $f_{Ln}(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{[\ln(x)-\mu]^2}{2\sigma^2}}$.

LgNrm returns $F_{Ln}(x) = \Phi\left(\frac{\ln(x)-\mu}{\sigma}\right)$ with $\Phi(z)$ denoting the *standard normal CDF* as presented on p. 211.



Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda3669.htm>

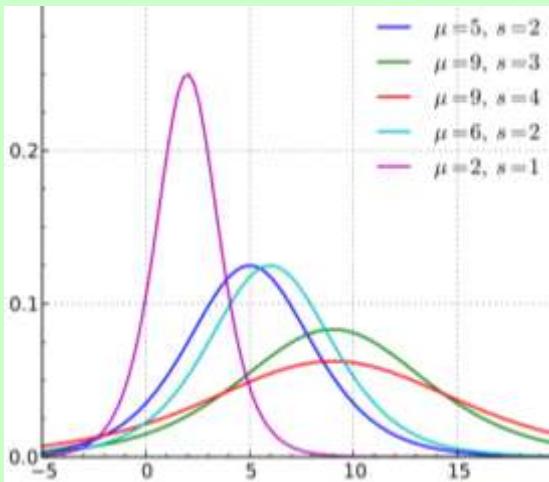
Logis: *Logistic distribution* with an arbitrary *mean* μ given in **I** and a *scale parameter* s in **J**.

Substituting $\xi = \frac{x-\mu}{s}$,

Logis_{P} returns $f_{Lg}(x) = \frac{e^{-\xi}}{(1+e^{-\xi})^2 s}$ and

Logis returns $F_{Lg}(x) = \frac{1}{1+e^{-\xi}}$.

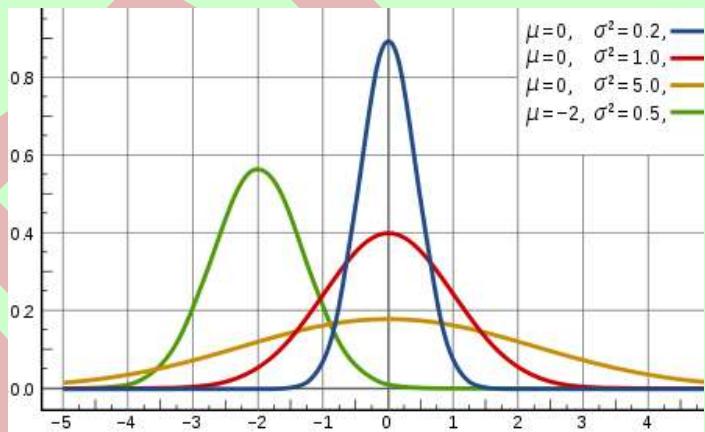
Logis^{-1} returns $F_{Lg}^{-1}(p) = \mu + s \ln\left(\frac{p}{1-p}\right)$.



Start reading here for more:

http://en.wikipedia.org/wiki/Logistic_distribution.

Norml: Normal distribution with an arbitrary mean μ given in I and an arbitrary standard deviation σ in J.



Norml_P returns $f_N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ and

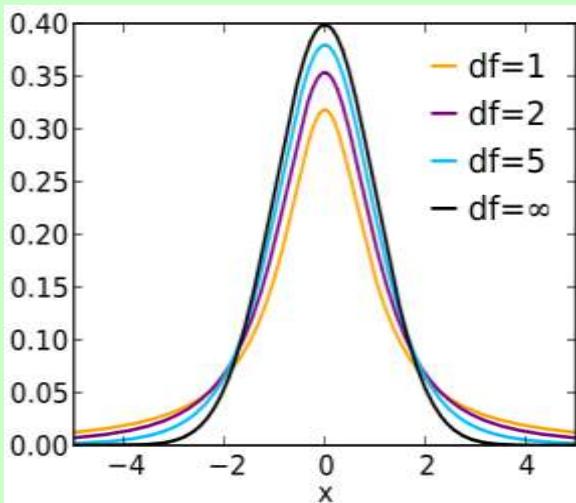
Norml returns $F_N(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$ with $\Phi(z)$ denoting the standard normal (or Gaussian) CDF as presented on p. 211.

Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm>

$t(x)$: Standardized Student's t distribution with its *degrees of freedom* in **I**.

It is used for hypothesis testing and calculating confidence intervals e.g. for means. The picture shows its *PDF* plotted for different degrees of freedom. For $df \rightarrow \infty$, the shoulders of $t(x)$ shrink and it approaches the *PDF* of the *standard normal distribution* (compare the red *Gaussian curve* at NORML on p. 209).



Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda3664.htm>

Weibl: Weibull distribution with its *shape parameter b* in **I** and its *characteristic lifetime T* in **J**.

Weibl_P returns $f_W(t) = \frac{b}{T} \cdot \left(\frac{t}{T}\right)^{b-1} e^{-(t/T)^b}$ for $t \geq 0$ (else it returns 0).

This is a very flexible function – see the curves plotted on next page.

Weibl returns $F_W(t) = 1 - e^{-(t/T)^b}$

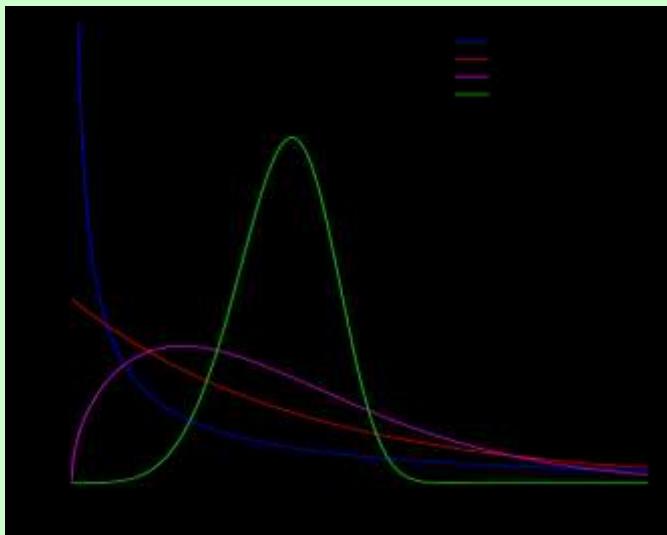
This distribution is widely used e.g. for analyzing tool and product lifetimes.

Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda3668.htm> .

You may check also

http://en.wikipedia.org/wiki/Weibull_distribution#Uses for some more application fields.

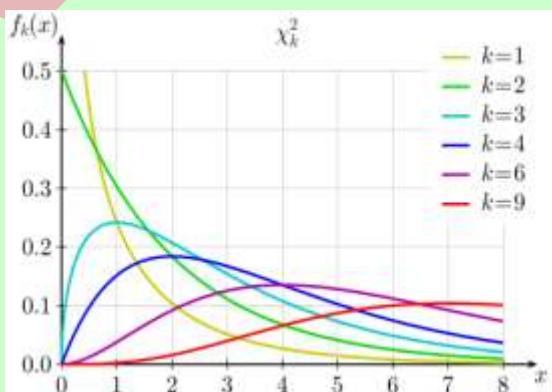


$\varphi(x)$ and $\Phi(x)$: $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ is the *standard normal PDF* (the famous *Gaussian bell curve*, pictured as red curve under NORML on p. 209), while $\Phi(x) = \int_{-\infty}^x \varphi(\tau) d\tau$ is the corresponding *CDF* (cf. the *error function* on p. 230). Take NORML instead for arbitrary means and standard deviations. See the link there for more information.

$\chi^2(x)$: *Chi-square distribution* with its *degrees of freedom* given in I. It is used for calculating confidence intervals for standard deviations, variances, process and machine capabilities, and the like. The diagram shows *PDF plots* for different *degrees of freedom*.

Read here for more information:

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda3666.htm>



More Statistical Formulas, also for Fitting

The following equations are for data measured at samples of n specimens (i.e. n is the *sample size*). Note that complete measurement results must include both: information about the expected value and about its uncertainty.

- For samples drawn out of a *Gaussian* (additive) process, the expected value is the *arithmetic mean* (or *average*) and its uncertainty is given by its *standard error* (see \bar{x} and s_m).
- For samples drawn out of a *log-normal* (multiplicative) process, the expected value is the *geometric mean* and its uncertainty is given by its *scattering factor* (see \bar{x}_g and ε_m).
- For samples drawn out of other kinds of processes other measures apply.

Be assured not everything is *Gaussian* in real world!⁵⁶ Process features can be detected (and should be checked well in advance of calculating e.g. means) using suitable tests – turn to applicable statistical reference literature.

The following functions as named in the left column are all found in STAT:

⁵⁶ Generally, the statistical model shall be chosen that matches observations best – within their statistical errors. In real life cases, however, dramatic deviations from the model distribution are frequently found – then you cannot expect the calculated consequences matching reality any better.

As mentioned in the main text, we recommend you look deeply into statistics textbooks to ensure you fully understand what you do with the functions provided in your WP 43S. The real world shows lots of sad examples where people full of good will caused large damages by applying tools they did not know sufficiently – or applied standard tools in areas where those are not applicable. “*Wenn Dumme fleißig werden, wird's gefährlich*” (i.e. “*It's getting dangerous when ignorant become busy*”), a former boss of mine used to say (cf. e.g. D.T. recently).

By the way: Since the *PDF* of the *Gaussian* distribution will never reach zero, this statistical model tells you to expect individual items far, far away from the mean value when your sample becomes large enough. This, however, does not match reality. So we must conclude nothing at all is really *Gaussian* in real world. Nevertheless, the *Gaussian* distribution is a very successful model describing a lot of real world observations very well. Just never forget the limits of such models.

Name	Remarks (see pp. 12ff for general information)
CORR	<p>For an arbitrary fit model $R(x)$, the <i>coefficient of correlation squared</i></p> $r^2 = 1 - \frac{\sum [R(x_i) - y_i]^2}{\sum (\bar{y} - y_i)^2}$ <p>is its <i>coefficient of determination</i> indicating the fraction of the variation of the dependent data y determined by the variation of the independent data x. For $r^2 = 1$, y is fully determined by x; for $r^2 = 0$, y is completely independent of x; and e.g. $r^2 = 0.85$ means 85% of the variation of y is due to x.</p> <p>Note BESTF picks the fit model showing the maximum r^2 out of LINF, EXPF, LOGF, and POWERF.</p> <p>A two-parameter regression (like the fit models provided on your WP 43S) is said being (statistically) <i>significant</i> if</p> $\sqrt{\frac{r^2}{1 - r^2} (n - 2)} > t_{n-2}^{-1}(0.99)$ <p>with the right side being the inverse of the <i>t distribution</i> for the <i>degrees of freedom</i> $n - 2$ and <i>confidence level</i> 99% (see p. 210).</p> <p>For a <u>linear</u> fit model, the <i>coefficient of correlation</i> is $r = \frac{s_{xy}}{s_x \cdot s_y}$.</p> <p>See s_{XY} and s below.</p>
COV	<p>For a linear fit model, the <i>population covariance</i> is</p> $COV_{xy} = \frac{1}{n^2} \left(n \sum x_i y_i - \sum x_i \sum y_i \right)$ <p>Compare s_{XY} below.</p>

Name	Remarks (see pp. 12ff for general information)
ExpF	Selects the exponential curve fit model $R(x) = a_0 e^{a_1 x}$ for least squares regression. ⁵⁷ Generally, this will be a good choice if the measured data follow the shape of one of the curves pictured here. ⁵⁸
LinF	Selects the linear fit model $R(x) = a_0 + a_1 x$ for least squares regression. ⁵⁷ Generally, this will be a good choice if the measured data follow a straight line, raising or falling (but cf. ORTHOF below).
LogF	Selects the logarithmic curve fit model $R(x) = a_0 + a_1 \ln(x)$ for least squares regression. ⁵⁷ Generally, this will be a good choice if the measured data follow a curve looking like shown at left.
L.R.	For a <u>linear</u> fit model, the least squares regression ⁵⁷ line parameters are $a_0 = \frac{\sum x_i^2 \cdot \sum y_i - \sum x_i \cdot \sum x_i y_i}{n \cdot (n-1) \cdot s_x^2} \quad \text{and} \quad a_1 = \frac{s_{xy}}{s_x^2} = r \cdot \frac{s_y}{s_x}$ (see CORR above, s_{XY} and s below).

⁵⁷ Note that *least squares regression* is best for data point errors in direction y being significantly greater than errors in direction x .

⁵⁸ Plots on this and next page are taken from the HP-27 manual; $b = a_1$ on your WP 43S.

Name	Remarks (see pp. 12ff for general information)
	<p>Their <i>standard errors</i> can be calculated using the formulas</p> $s(a_1) = \frac{s_y}{s_x} \sqrt{\frac{1-r^2}{n-2}} \text{ and } s(a_0) = s(a_1) \cdot \sqrt{\frac{n-1}{n} s_x^2 + \bar{x}^2} .$ <p>For all curve fit models provided on your WP 43S, a regression parameter is (statistically) <i>significant</i> if</p> $\left \frac{a_i}{s(a_i)} \right > t_{n-2}^{-1}(0.995) ,$ <p>with the right side being the inverse of the <i>t distribution</i> for the <i>degrees of freedom</i> $n - 2$ and 99% <i>confidence level</i> (see p. 210).</p>
OrthoF	<p>Selects the linear fit model $R(x) = a_0 + a_1x$ like LINF, though assuming data point errors in x are equal to those in y (precisely: their variances are equal). The sum of squared distances of the data points to the fit line will be minimized. This model is called <i>orthogonal regression</i>. The line parameters are</p> $a_1 = \frac{1}{2s_{xy}} \left[s_y^2 - s_x^2 \pm \sqrt{(s_y^2 - s_x^2)^2 + 4s_{xy}^2} \right] \text{ and } a_0 = \bar{y} - a_1 \bar{x}$ <p>(see s_{XY} and s below).</p>
PowerF	<p>Selects the power curve fit model $R(x) = a_0 x^{a_1}$ for least squares regression.⁵⁷</p> <p>Generally, this will be a good choice if the measured data follow the shape of one of the curves shown here.⁵⁸</p>

Name	Remarks (see pp. 12ff for general information)
s_{XY}	For any linear fit model (LinF or OrthoF), the <i>sample covariance</i> is $s_{xy} = \frac{1}{n(n-1)} \left(n \sum x_i y_i - \sum x_i \sum y_i \right)$ <p style="text-align: right;">Compare COV above.</p>
s, s_m	The <i>sample standard deviation (SD)</i> is the positive square root of the <i>sample variance</i> $s_x^2 = \frac{1}{n(n-1)} \left[n \sum x_i^2 - \left(\sum x_i \right)^2 \right] = \frac{1}{n-1} \left(\sum x_i^2 - n \bar{x}^2 \right)$ <p>And the <i>standard error</i> (i.e. the <i>SD</i> of the <i>mean</i> \bar{x}) is $s_{mx} = s_x / \sqrt{n}$</p>
s_w, s_{mw}	The <i>sample SD</i> for <u>weighted</u> data (where the weight y_i of each data point x_i was entered via $\Sigma+$) is $s_w = \sqrt{\frac{\sum y_i \sum y_i x_i^2 - (\sum y_i x_i)^2}{\sum y_i (\sum y_i - 1)}}$ <p>And the corresponding <i>standard error</i> (the <i>SD</i> of the <i>mean</i> \bar{x}_w) is $s_{mw} = \frac{1}{\sum y_i} \sqrt{\frac{\sum y_i \sum y_i x_i^2 - (\sum y_i x_i)^2}{\sum y_i - 1}}$</p>
\bar{x}	The <i>arithmetic mean</i> is calculated as $\bar{x} = \frac{1}{n} \sum x_i$
\bar{x}_G	The <i>geometric mean</i> is calculated as $\bar{x}_G = \sqrt[n]{(\prod x_i)} = e^{\left[\frac{1}{n} \sum \ln(x_i) \right]}$.
\bar{x}_w	The <i>arithmetic mean</i> for <u>weighted</u> data (where the weight y_i of each data point x_i was entered via $\Sigma+$) is $\bar{x}_w = \frac{\sum x_i y_i}{\sum y_i}$

Name	Remarks (see pp. 12ff for general information)
ε	The <i>scattering factor</i> ε_x for a sample of <i>log-normally</i> distributed data is calculated via: $\ln(\varepsilon_x) = \sqrt{\frac{1}{n-1} \left[\sum \ln^2(x_i) - 2n \ln(\bar{x}_G) \right]}$ <p style="text-align: right;">Compare s.</p>
ε_m	The <i>scattering factor</i> of the <i>geometric mean</i> is $\varepsilon_m = \varepsilon^{\frac{1}{\sqrt{n}}}$. <p style="text-align: right;">Compare s_m.</p>
ε_p	The <i>scattering factor</i> ε_p for a population of <i>log-normally</i> distributed data is calculated via: $\ln(\varepsilon_p) = \sqrt{\frac{n-1}{n}} \ln(\varepsilon_x)$ <p style="text-align: right;">Compare σ.</p>
σ	The <i>SD</i> of a population of <i>normally</i> distributed data is calculated via: $\sigma_x = \frac{1}{n} \sqrt{\sum x_i^2 - n \bar{x}^2} = \sqrt{\frac{n-1}{n}} s_x$ <p style="text-align: right;">Compare ε_p.</p>
σ_w	The <i>SD</i> of the population for <u>weighted</u> data (where the weight y_i of each data point x_i was entered via $\Sigma+$) is $\sigma_w = \sqrt{\frac{\sum y_i (x_i - \bar{x}_w)^2}{\sum y_i}}$

About Error Propagation

As mentioned in footnote 32 on p. 124, experimental data are always attended with errors, caused by e.g. the uncertainty of the measuring method, the instrument used, and environmental variations. Even under controlled measuring conditions, random errors remain. These errors must be taken into account for a proper estimation of the uncertainty of your results computed using those experimental data. For about 200 years, Gauß' *least squares method* can be employed for this task.

Assume you know that your result R depends on several experimental parameters x_1 through x_n . Each such parameter x_i has an uncertainty or error Δx_i . Now, if $R = f(x_1, \dots, x_n)$ then

$$\begin{aligned}\Delta R &= f(x_1 \pm \Delta x_1, \dots, x_n \pm \Delta x_n) - f(x_1, \dots, x_n) \\ &= \pm \sqrt{\left(\frac{df}{dx_1}\right)^2 \Delta x_1^2 + \dots + \left(\frac{df}{dx_n}\right)^2 \Delta x_n^2}\end{aligned}$$

Often, however, the differential terms under the square root are tedious to determine analytically.

But this root can be written simpler: $\Delta R = \pm \sqrt{\Delta f_1^2 + \dots + \Delta f_n^2}$.

And with your *WP 43S*, the following algorithm will do for computing ΔR , even if f is 'strongly curved':

1. Program the function $R = f(x_1, x_2, \dots, x_n)$ in a way you can vary its parameters easily.
2. Let your *WP 43S* compute $= f(x_1, x_2, \dots, x_n)$.
3. Let it compute $R_{1+} = f(x_1 + \Delta x_1, x_2, \dots, x_n)$ and $\Delta R_{1+} = R_{1+} - R$.
4. Let it compute $R_{1-} = f(x_1 - \Delta x_1, x_2, \dots, x_n)$ and $\Delta R_{1-} = R_{1-} - R$.
5. Let it compute $R_{2+} = f(x_1, x_2 + \Delta x_2, \dots, x_n)$ and $\Delta R_{2+} = R_{2+} - R$.
6. Let it compute $R_{2-} = f(x_1, x_2 - \Delta x_2, \dots, x_n)$ and $\Delta R_{2-} = R_{2-} - R$.
7. Repeat the last two steps for each remaining parameter.

Being through with all n parameters, you will end with

$$\Delta R = \pm \sqrt{\frac{1}{2} (\Delta R_{1+}^2 + \Delta R_{1-}^2 + \Delta R_{2+}^2 + \Delta R_{2-}^2 + \cdots + \Delta R_{n+}^2 + \Delta R_{n-}^2)}$$

So the terms under the square root have become simple differences which are determined most easily with the help of your *WP 43S*.

For ‘small’ errors or less curvature, the following simpler algorithm will do, requiring down to half as many steps only:

1. Program the function $R = f(x_1, x_2, \dots, x_n)$ in a way you can vary its parameters easily.
2. Let your *WP 43S* compute $= f(x_1, x_2, \dots, x_n)$.
3. Let it compute $R_1 = f(x_1 + \Delta x_1, x_2, \dots, x_n)$ and $\Delta R_1 = R_1 - R$.
4. Let it compute $R_2 = f(x_1, x_2 + \Delta x_2, \dots, x_n)$ and $\Delta R_2 = R_2 - R$.
5. Repeat the last step for each remaining parameter.

Being through with all n parameters, you will end with

$$\Delta R = \pm \sqrt{\Delta R_1^2 + \Delta R_2^2 + \cdots + \Delta R_n^2}$$

You might know this formula from your university or lab classes.

More about Curve Fitting

Actually, a proper linear regression is computed for LINF and ORTHOF only. For the other three models (EXPF, LOGF, and POWERF) the same method is applied to transformed data. Your data might follow a straight line if you plot...

- the logarithm of your y -data over your x -data (then EXPF will fit);
- the logarithm of your y -data over the logarithm of your x -data (then POWERF will fit);
- your y -data over the logarithm of your x -data (then LOGF will fit).

This is what your WP 43S does when you enter statistical data points and compute a fit curve thereafter:

1. It accumulates the 13 sums listed on p. 93 and increments n .
2. The evaluation depends on the model you select:
 - a. If you choose LINF then the least squares regression line parameters a_0 and a_1 will be computed following the formulas on p. 214. Their standard errors can be calculated using the formulas on p. 215 with

$$r = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \cdot \sqrt{n \sum y_i^2 - (\sum y_i)^2}}$$

- b. If you choose EXPF then the least squares regression line parameters for the transformed data $x_i, \ln(y_i)$ will be computed using

$$a_{0,tEXP} = \frac{\sum x_i^2 \cdot \sum \ln(y_i) - \sum x_i \cdot \sum x_i \ln(y_i)}{n \sum x_i^2 - (\sum x_i)^2}$$

$$a_{1,tEXP} = \frac{n \sum x_i \ln(y_i) - \sum x_i \sum \ln(y_i)}{n \sum x_i^2 - (\sum x_i)^2}$$

The correlation coefficient for the transformed data will be

$$r_{tEXP} = \frac{n \sum x_i \ln(y_i) - \sum x_i \sum \ln(y_i)}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \cdot \sqrt{n \sum \ln^2(y_i) - [\sum \ln(y_i)]^2}}$$

The standard errors of $a_{0,tEXP}$ and $a_{1,tEXP}$ can be calculated using the formulas on p. 215 with the transformed results.

The parameters of the fit curve $R(x) = a_0 e^{a_1 x}$ turn out being $a_0 = e^{a_{0,tEXP}}$ and $a_1 = a_{1,tEXP}$.

- c. If you choose POWERF then the least squares regression line parameters for the transformed data $\ln(x_i)$, $\ln(y_i)$ will be computed in analogy to the method shown for EXPF. Thus they will be

$$a_{0,tPOW} = \frac{\sum \ln^2(x_i) \cdot \sum \ln(y_i) - \sum \ln(x_i) \cdot \sum \ln(x_i) \ln(y_i)}{n \sum \ln^2(x_i) - [\sum \ln(x_i)]^2}$$

$$a_{1,tPOW} = \frac{n \sum \ln(x_i) \ln(y_i) - \sum \ln(x_i) \sum \ln(y_i)}{n \sum \ln^2(x_i) - [\sum \ln(x_i)]^2}$$

$$r_{tPOW} = \frac{n \sum \ln(x_i) \ln(y_i) - \sum \ln(x_i) \sum \ln(y_i)}{\sqrt{n \sum \ln^2(x_i) - [\sum \ln(x_i)]^2} \cdot \sqrt{n \sum \ln^2(y_i) - [\sum \ln(y_i)]^2}}$$

The standard errors of $a_{0,tPOW}$ and $a_{1,tPOW}$ can be calculated using the formulas on p. 215 with the transformed results.

The parameters of the fit curve $R(x) = a_0 x^{a_1}$ turn out being $a_0 = e^{a_{0,tPOW}}$ and $a_1 = a_{1,tPOW}$.

- d. If you choose LOGF then the least squares regression line parameters for the transformed data $\ln(x_i)$, y_i will be computed in analogy to the method shown for EXPF. Thus they will be

$$a_{0,tLOG} = \frac{\sum \ln^2(x_i) \cdot \sum y_i - \sum \ln(x_i) \cdot \sum y_i \ln(x_i)}{n \sum \ln^2(x_i) - [\sum \ln(x_i)]^2}$$

$$a_{1,tLOG} = \frac{n \sum y_i \ln(x_i) - \sum \ln(x_i) \sum y_i}{n \sum \ln^2(x_i) - [\sum \ln(x_i)]^2}$$

$$r_{tLOG} = \frac{n \sum y_i \ln(x_i) - \sum \ln(x_i) \sum y_i}{\sqrt{n \sum \ln^2(x_i) - [\sum \ln(x_i)]^2} \cdot \sqrt{n \sum y_i^2 - [\sum y_i]^2}}$$

The standard errors of $a_{0,tLOG}$ and $a_{1,tLOG}$ can be calculated using the formulas on p. 215 with the transformed results.

The parameters of the fit curve $R(x) = a_0 + a_1 \ln(x)$ are just $a_0 = a_{0,tLOG}$ and $a_1 = a_{1,tLOG}$.

- e. If you choose BESTF then the correlation coefficient will be computed with your data for model a and with the transformed data for models b, c, and d. The model delivering the greatest absolute r value will be selected.

Solving Differential Equations

The method applied to the examples in the respective chapter in Section 3 of the OM develops as explained below:

First, we solve one-dimensional problems of the kind

$$\frac{d^2f}{dt^2} = a - b \left(\frac{df}{dt} \right)^2$$

This is the equation for a body (of mass M) falling through a medium featuring drag proportional to the velocity squared of said body. For earthly problems, take $a = 9.81 \frac{m}{s^2} = g$ and $b = \delta/M$ with the constant parameter δ taking care of the viscosity of the medium as well as size and shape of the falling body as a whole.

For a first guess, let us assume $b = 0$. So there will be no drag at all, the body will be just accelerated by $a = g$. Then for two arbitrary subsequent points in time,

- vertical velocity will develop like $\left(\frac{df}{dt} \right)_{i+1} \approx \left(\frac{df}{dt} \right)_i + a\Delta t$ and
- position over ground like $f_{i+1} \approx f_i + \left(\frac{df}{dt} \right)_i \Delta t$.

Proceeding from time zero in small, constant time steps $\Delta t = t_{i+1} - t_i$:

- $f_1 = f_0 + \left(\frac{df}{dt} \right)_0$ and $\left(\frac{df}{dt} \right)_1 = \left(\frac{df}{dt} \right)_0 + a\Delta t$,

- $f_2 = f_1 + \left(\frac{df}{dt}\right)_1$ and $\left(\frac{df}{dt}\right)_2 = \left(\frac{df}{dt}\right)_1 + a\Delta t$, etc.

Principally, a better approximation of the slope of f is achieved using the so-called *half-step method*:

$$\left(\frac{df}{dt}\right)_{1/2} \approx \left(\frac{df}{dt}\right)_0 + a \frac{\Delta t}{2}$$

$$\left(\frac{df}{dt}\right)_{i+\frac{1}{2}} \approx \left(\frac{df}{dt}\right)_{i-\frac{1}{2}} + a\Delta t$$

$$f_{i+1} \approx f_i + \left(\frac{df}{dt}\right)_{i+1/2} \Delta t$$

Proceeding from time zero in small steps Δt again, we get

- $\left(\frac{df}{dt}\right)_{1/2} = \left(\frac{df}{dt}\right)_0 + a \frac{\Delta t}{2}$
- $f_1 = f_0 + \left(\frac{df}{dt}\right)_{1/2}$ and $\left(\frac{df}{dt}\right)_{3/2} = \left(\frac{df}{dt}\right)_{1/2} + a\Delta t$,
- $f_2 = f_1 + \left(\frac{df}{dt}\right)_{3/2} \Delta t$, etc.

Let us drop the restriction for b now. Replacing a in the previous set of equations by the right side of the differential equation on p. 222, we will get the following new set:

$$\frac{df}{dt}_{1/2} \approx \frac{df}{dt}_0 + \left[a - b \left(\frac{df}{dt}\right)_0^2 \right] \frac{\Delta t}{2}$$

$$\left(\frac{df}{dt}\right)_{i+1/2} \approx \left(\frac{df}{dt}\right)_{i-\frac{1}{2}} + \left[a - b \left(\frac{df}{dt}\right)_{i-\frac{1}{2}}^2 \right] \Delta t$$

$$f_{i+1} \approx f_i + \left(\frac{df}{dt}\right)_{i+1/2} \Delta t$$

Proceeding from time zero in small steps Δt again, we get

- $\left(\frac{df}{dt}\right)_{1/2} = \left(\frac{df}{dt}\right)_0 + \left[a - b \left(\frac{df}{dt}\right)_0^2\right] \frac{\Delta t}{2}$
- $f_1 = f_0 + \left(\frac{df}{dt}\right)_{1/2}$ and $\left(\frac{df}{dt}\right)_{3/2} = \left(\frac{df}{dt}\right)_{1/2} + \left[a - b \left(\frac{df}{dt}\right)_{1/2}^2\right] \Delta t,$
- $f_2 = f_1 + \left(\frac{df}{dt}\right)_{3/2} \Delta t$, etc.

This half-step method as explained above can be applied easily to all ordinary differential equations of second order which can be written like

$$\frac{d^2f}{dt^2} = h\left(t, f, \frac{df}{dt}\right)$$

with an arbitrary real function h depending on time, the function itself and its first derivative. The equations applicable in this general case are

- $\left(\frac{df}{dt}\right)_{1/2} = \left(\frac{df}{dt}\right)_0 + h\left(t_0, f_0, \left[\frac{df}{dt}\right]_0\right) \frac{\Delta t}{2}$
- $\left(\frac{df}{dt}\right)_{i+1/2} = \left(\frac{df}{dt}\right)_{i-1/2} + h\left(t_{i-1/2}, f_{i-1/2}, \left[\frac{df}{dt}\right]_{i-1/2}\right) \Delta t$
- $f_{i+1} = f_i + \left[\frac{df}{dt}\right]_{i-1/2} \Delta t$

For solving a 2D problem like e.g. finding the orbit of a satellite in the gravitational field of the earth, we need two differential equations, one for x and one for y :

$$\frac{d^2x}{dt^2} = \frac{F_x}{m} = -\frac{F}{m} \frac{x}{\sqrt{x^2 + y^2}} \quad \text{and} \quad \frac{d^2y}{dt^2} = \frac{F_y}{m} = -\frac{F}{m} \frac{y}{\sqrt{x^2 + y^2}}.$$

And we know $F = G m M / (x^2 + y^2)$, thus

$$\frac{d^2x}{dt^2} = -\frac{GM}{(x^2 + y^2)^{3/2}} x = K_x \quad \text{and} \quad \frac{d^2y}{dt^2} = -\frac{GM}{(x^2 + y^2)^{3/2}} y = K_y$$

This is a pair of coupled differential equations. It is solved as follows:

$$\left(\frac{dx}{dt}\right)_{1/2} \approx \left(\frac{dx}{dt}\right)_0 + K_x \frac{\Delta t}{2} \quad \left(\frac{dy}{dt}\right)_{1/2} \approx \left(\frac{dy}{dt}\right)_0 + K_y \frac{\Delta t}{2}$$

$$\left(\frac{dx}{dt}\right)_{i+\frac{1}{2}} \approx \left(\frac{dx}{dt}\right)_{i-\frac{1}{2}} + K_x \Delta t \quad \left(\frac{dy}{dt}\right)_{i+\frac{1}{2}} \approx \left(\frac{dy}{dt}\right)_{i-\frac{1}{2}} + K_y \Delta t$$

$$x_{i+1} \approx x_i + \left(\frac{dx}{dt}\right)_{i+1/2} \Delta t \quad y_{i+1} \approx y_i + \left(\frac{dy}{dt}\right)_{i+1/2} \Delta t$$

Orthogonal Polynomials

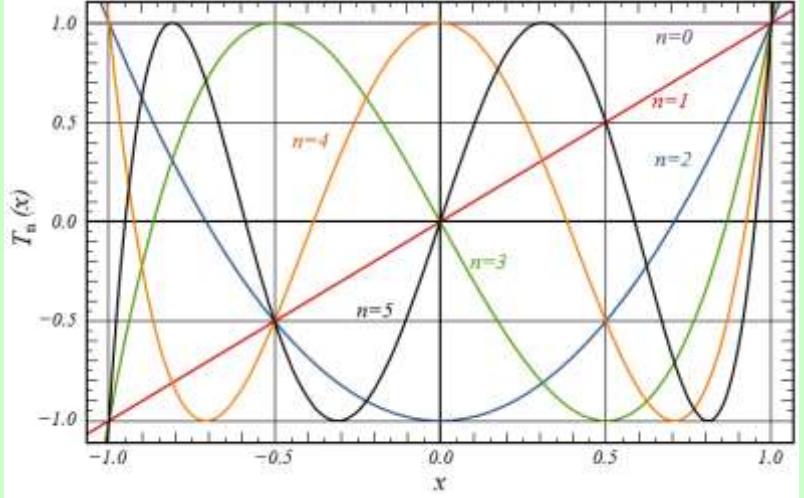
The following polynomials are all collected in X.FN'ORTHO.

Name	Remarks (see pp. 12ff for general information)
H_n	<p>Hermite polynomials for probability: $H_n(x) = (-1)^n \cdot e^{\frac{x^2}{2}} \cdot \frac{d^n}{dx^n} \left(e^{-\frac{x^2}{2}} \right)$</p> <p>with n in Y, solving the differential equation</p> $f''(x) - 2x \cdot f'(x) + 2n \cdot f(x) = 0.$ <p>See the first five polynomials plotted overleaf.</p>

Name	Remarks (see pp. 12ff for general information)
H_{np}	<p>Hermite polynomials for physics: $H_{np}(x) = (-1)^n \cdot e^{x^2} \cdot \frac{d^n}{dx^n}(e^{-x^2})$</p> <p>with n in Y, solving the same differential equation. See the first five polynomials plotted below.</p>

Name	Remarks (see pp. 12ff for general information)
L_n	<p>Laguerre polynomials (compare $L_{n\alpha}$ below):</p> $L_n(x) = \frac{e^x}{n!} \cdot \frac{d^n}{dx^n} (x^n e^{-x}) = L_n^{(0)}(x)$ <p>with n in \mathbb{Y}, solving the differential equation $x \cdot f''(x) + (1 - x) \cdot f'(x) + n \cdot f(x) = 0$.</p> <p>See the first five Laguerre polynomials plotted here.</p>
$L_{n\alpha}$	<p>Laguerre's generalized polynomials (compare L_n above):</p> $L_n^{(\alpha)}(x) = \frac{x^{-\alpha} e^x}{n!} \cdot \frac{d^n}{dx^n} (x^{n+\alpha} e^{-x})$ <p>with n in \mathbb{Y} and α in \mathbb{Z}. Some of them are plotted below ($k = \alpha$).</p>

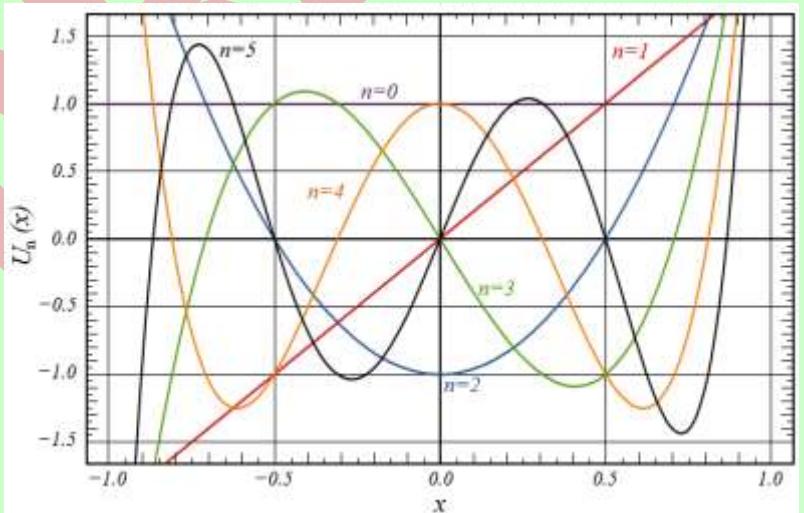
Name	Remarks (see pp. 12ff for general information)
P_n	<p>Legendre polynomials: $P_n(x) = \frac{1}{2^n n!} \cdot \frac{d^n}{dx^n} [(x^2 - 1)^n]$ with n in \mathbb{Y}, solving the differential equation</p> $\frac{d}{dx} \left[(1 - x^2) \cdot \frac{d}{dx} f(x) \right] + n(n+1)f(x) = 0.$ <p>See the first six polynomials plotted here.</p>
T_n	<p>Chebyshev (a.k.a. Čebyšev, Tschebyschow, Tschebyscheff) polynomials of first kind</p> $T_n(x) = \begin{cases} \cos(n \arccos(x)) & \text{for } -1 \leq x \leq 1 \\ \cosh(n \operatorname{arcosh}(x)) & \text{for } x > 1 \\ (-1)^n \cosh(n \operatorname{arcosh}(-x)) & \text{for } x < -1 \end{cases} \quad \text{with } n \text{ in } \mathbb{Y},$ <p>solving the differential equation</p> $f''(x) - \frac{x}{1-x^2} f'(x) + \frac{n^2}{1-x^2} f(x) = 0$ <p>The plot overleaf shows $T_0(x) \dots T_5(x)$.</p>

Name	Remarks (see pp. 12ff for general information)
	 <p>The plot shows the Chebyshev polynomials of the first kind, $T_n(x)$, for $n = 0, 1, 2, 3, 4, 5$. The x-axis ranges from -1.0 to 1.0, and the y-axis ranges from -1.0 to 1.0. The curves are symmetric about the y-axis. T_0 is a constant 1.0. T_1 is a blue line from $(-1,0)$ to $(1,1)$. T_2 is a red parabola opening upwards. T_3 is a green curve with a local maximum at $x=0$. T_4 is an orange curve with a local minimum at $x=0$. T_5 is a black curve with a local maximum at $x=0$.</p>

U_n **Chebyshev polynomials of second kind** $U_n(x)$ with n in Y, solving the differential equation

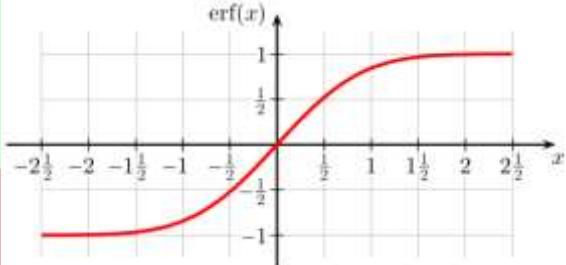
$$f''(x) - \frac{3x}{1-x^2} f'(x) + \frac{n(n+2)}{1-x^2} f(x) = 0$$

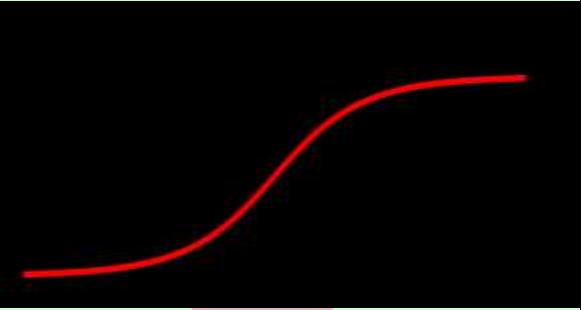
The plot below shows $U_0(x) \dots U_5(x)$:



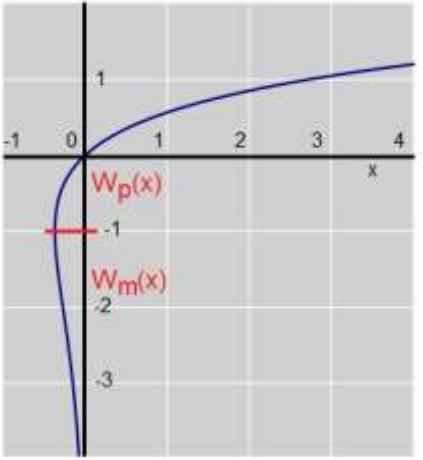
Even More Mathematical Functions

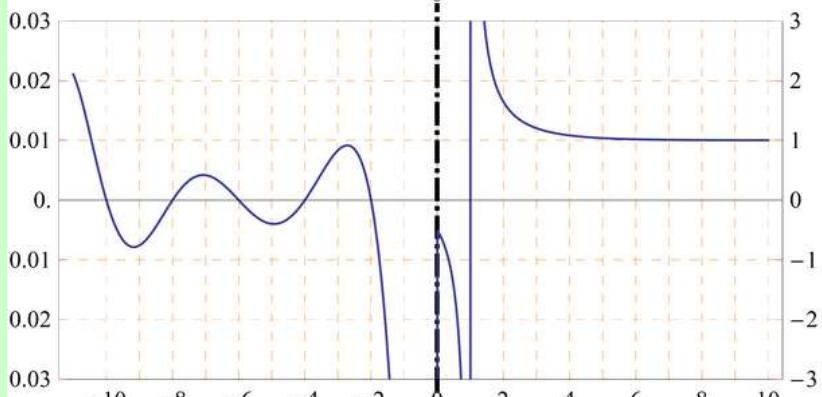
All the following functions are found in X.FN. Some of them are for pure mathematics only but were useful at some stages of the *WP 34S* or *WP 43S* projects, so we made them accessible for the public.

Name	Remarks (see pp. 12ff for general information)
AGM	Returns the <i>arithmetic-geometric mean</i> . Find more about it here: http://mathworld.wolfram.com/Arithmetic-GeometricMean.html .
erf	Returns the <i>error function</i> $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.  <p>Note that $\text{erf}\left(\frac{x}{\sqrt{2}}\right) = 2 \cdot \Phi(x) - 1$ with $\Phi(x)$ representing the standard normal CDF as described on p. 211. Beyond statistics, the error function may be helpful in heat conduction and diffusion problems, for instance.</p>
erfc	This command returns the <i>complementary error function</i> $\text{erfc}(x) = 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$. This function is related to the <i>error probability</i> of the standard normal distribution.

Name	Remarks (see pp. 12ff for general information)
g_d	<p>Returns the <i>Gudermannian function</i></p> $g_d(x) = \int_0^x \frac{d\xi}{\cosh \xi}$ <p>linking hyperbolic and trigonometric functions. See the plot for its real values. The <i>inverse Gudermannian function</i> is</p> $g_d^{-1}(x) = \int_0^x \frac{d\xi}{\cos \xi}.$ <p>Start reading here for more: http://en.wikipedia.org/wiki/Gudermannian_function.</p> 
I_{xyz}	<p>Returns the <i>regularized beta function</i> $\frac{\beta_x(x, y, z)}{B(y, z)}$ with</p> $\beta_x(x, y, z) = \int_0^x t^{y-1} (1-t)^{z-1} dt$ <p>being the <i>incomplete beta function</i> and $B(y, z)$ being <i>Euler's Beta function</i> (see p. 90).</p>
$I\Gamma_p$	<p>Returns the <i>regularized gamma function</i> $P(x, y) = \frac{\gamma(x, y)}{\Gamma(x)}$.</p> <p>See γ_{XY} below for $\gamma(x, y)$ and p. 90 for $\Gamma(x)$.</p>
$I\Gamma_q$	<p>Returns the <i>regularized gamma function</i> $Q(x, y) = \frac{\Gamma_u(x, y)}{\Gamma(x)}$.</p> <p>See Γ_{XY} below for $\Gamma_u(x, y)$ and p. 90 for $\Gamma(x)$.</p>

Name	Remarks (see pp. 12ff for general information)
$J_y(x)$	<p>Generally, the <i>Bessel functions</i> solve the differential equation $x^2 f''(x) + xf'(x) + (x^2 - \nu^2)f(x) = 0$ with $\nu \in \mathbb{C}$.</p> <p>$J_y(x)$ returns the <i>Bessel function of first kind</i> and order $y = \nu$. For arbitrary ν, this is</p> $J_\nu(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m! \Gamma(m+\nu+1)} \left(\frac{x}{2}\right)^{2m+\nu}$ <p>For integer ν, this is also</p> $J_\nu(x) = \frac{1}{\pi} \int_0^\pi \cos[\nu t - x \sin(t)] dt$ <p>Start reading here for more information: http://en.wikipedia.org/wiki/Bessel_function.</p>

Name	Remarks (see pp. 12ff for general information)
W_p , W_m	<p>Returns <i>Lambert's W</i> with its principal branch (called W_p here) and its negative branch (called W_m for <u>minus</u>). The connecting point is $(-1/e, -1)$. The diagram shows the real values of both branches.</p> <p>Start reading here for more information: http://en.wikipedia.org/wiki/Lambert_W_function . Learn more here: http://mathworld.wolfram.com/LambertW-Function.html .</p> 
γ_{xy}	<p>Returns the <i>lower incomplete gamma function</i> $\gamma(x, y) = \int_0^y t^{x-1} e^{-t} dt$. Required for $I\Gamma_p$ above.</p>
Γ_{xy}	<p>Returns the <i>upper incomplete gamma function</i> $\Gamma_u(x, y) = \int_y^\infty t^{x-1} e^{-t} dt$. Required for $I\Gamma_q$ above.</p>
$\zeta(x)$	<p>Returns <i>Riemann's Zeta</i> for real arguments, with $\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x}$ for $x > 1$, and its analytical continuation for $x < 1$: $\zeta(x) = 2^x \pi^{x-1} \sin\left(\frac{\pi}{2} x\right) \cdot \Gamma(1-x) \cdot \zeta(1-x).$ <p>Note the different vertical scales for negative and positive x in the plot overleaf.</p> </p>

Name	Remarks (see pp. 12ff for general information)
	 <p>Look here for more: http://mathworld.wolfram.com/RiemannZetaFunction.html.</p>

Note the *error function* as well as *Laguerre*, *Legendre*, and *Bessel functions* were already provided on the *Commodore M55* pocket calculator of 1976/77 (featuring 55 keys).

Beyond what is printed in this appendix, you will find lots of information about the special functions implemented in your *WP 43S* in the internet in addition. Generally speaking, *Wikipedia* is a good starter – we recommend checking the articles in different languages since they may well contain different material and use different approaches. For applied statistics, the *NIST Sematech* online handbook (quoted on pp. 203ff) is a competent source. And *Mathworld* (quoted on pp. 230ff) may contain more details than you ever want to know. Further references are found at these sites.

APPENDIX K: BACKGROUND CONSIDERATIONS

Alpha Register

For long I thought we could do without a dedicated *alpha register*. Some special programming functions like KEYG and KEYX, however, seem to require such a *register* – else handling these functions would become more complicated than it was on the *HP-42S*. Especially direct entry of alphanumeric constants in programs is easier when the destination is automatically defined, and people became used to this method in decades since the *HP-42S* was launched. Thus, I introduced this *register* in v0.7, taking **K** for this task.

Angles

Originally, a separate *data type* for *angles* was planned. It was removed in v0.9 since its scope is quite limited and the opinion rose that ‘*angles work like reals*’. It turned out, however, that D.MS data would need special treatment in calculations, so *data type* 4 returned with v0.10 for sake of keeping algebraic operations simple and avoiding special purpose commands like D.MS+, D.MS–, etc.

Actually, angles are displayed in five ‘modes’ (*decimal and sexagesimal degrees, radians, multiples of π , and gon or grads*) but may be represented in a fixed format internally – similar to *short integers* where a fixed bit pattern may be displayed in various *integer sign modes* and bases. *Decimal degrees or multiples of π* may be beneficial as internal representations since avoiding the necessity for high precision storage of π for modulo calculations.

Display Limits

Due to the character sizes mentioned in *Appendix D: Display Internals*, the screen could take inputs of up to 23 digits, a sign, and an 8-pixel radix mark:

-4.2345678901234567890123 ,

occupying $15 + 23 \times 16 + 8 = 391$ px. Numeric output would allow for the same 23 digits. Without digit group separators, however, this would be hardly readable. With 3-digit separators (*startup default mode*), 20 digits are displayable in one row instead:

-4.234 567 890 123 456 789 0 ,

taking $15 + 20 \times 16 + 7 \times 8 = 391$ px. This maximum precision is independent of the position of the radix mark. Scientific or engineering notation allows for a 16-digit mantissa

-4.234 567 890 123 456 $\times 10^{-925}$,

taking 395 px ($15 + 16 \times 16 + 5 \times 8 + 15 + 16 + 4 \times 13 + 1$) for displaying this number this way.

For double precision numbers, a 34-digit mantissa plus exponent can be shown using the small font:

-8.123 456 789 012 345 678 901 234 567 890 123 $\times 10^{-925}$.

Some *temporary information* may limit output precision. E.g. for linear regression, up to 8 digits are viable allowing for 2-digit exponents in SCI or ENG and up to 12 digits in FIX:

Logarithmic* $a_1: -5.234\ 567\ 8 \times 10^{-92}$
 $y = a_0 + a_1 \ln(x)$ $a_0: -1.234\ 567\ 890\ 12$.

Complex numbers in Cartesian notation require 44 px for $+i\times$ in addition to the space for two reals. Only the real part may need extra space for a 15-pixel sign. Thus, there are 343 px left for the digits, allowing for

-4.234 567 89+j $\times 4.234\ 567\ 89$,

i.e. 8 decimals per part in worst case (since $44 + 15 + 2 \times (16 + 8 + 48 + 8 + 48 + 8 + 32) = 59 + 2 \times 159 = 397$ px in total).

This applies if both real and imaginary parts are in the same order of magnitude and the multiplication cross is chosen.

With SCI or ENG, a minimum of 3 decimals can be shown:

$$-4.234 \times 10^{-925} + i \times 4.234 \times 10^{-925}$$

Using multiplication dots instead, displaying one decimal more is feasible:

$$-4,234\ 5 \cdot 10^{-925} + i \cdot 4,234\ 5 \cdot 10^{-925}$$

Else, 13 pixel wide narrow digits will allow for 4 decimals even with multiplication crosses:

$$-6.234\ 5 \times 10^{-925} + i \times 6.234\ 5 \times 10^{-925}$$

With dots instead of crosses, 5 decimals minimum are feasible for both parts here:

$$-6.234\ 56 \cdot 10^{-925} + i \cdot 6.234\ 56 \cdot 10^{-925}$$

Complex numbers in polar notation require 32 px for π plus 16 px for the angular unit in addition to the space for two signed reals. Both magnitude and angle may need a 15-pixel sign. Thus, there are 322 px left for the digits, allowing for 7 decimals output in FIX:

$$-4.234\ 567\ 8\ \pi -0.234\ 567\ 8\pi$$

With SCI or ENG, the minimum number of decimals depends on the angular display mode since output is confined to the interval -180° to $+180^\circ$ or its equivalents, e.g. $-\pi$ to $+\pi$ in *radians* or -200^g to $+200^g$ in *gon* (see Sect. 2 of the OM). Hence, the angular parts can be displayed without exponents always. This allows for a minimum of 4 decimals for *degrees* and *gon*:

$$-4.234\ 5 \times 10^{-925}\ \pi -120.234\ 5^g$$

For *radians* or *multiples of π* , however, a minimum of 5 decimals can be displayed always:

$$-4.234\ 56 \times 10^{-925}\ \pi -0.234\ 56\pi$$

Digits in **fractions** (and exponents) are 13 px wide. Thus, a 4-digit numerator and denominator take $4 \times 13 + 8 = 60$ px each; the fraction

bar takes another 16 px and the trailer 29 (= 16 + 12 + 1). The remaining 235 px would suffice for 11 digits, the 16-pixel gap between integer and fraction, and the optional sign ($15 + 12 \times 16 + 3 \times 8 = 231$ px) in a proper fraction:

-67 890 234 567 2 289/4 567 > .

So we would need 396 px in total here (cf. *Section 2* of the OM).

For ***long integers***, up to 21 digits and a sign may be displayed using large digits:

-123 456 789 012 345 678 901 .

Larger ***long integers*** are shown using the small font, allowing for 42 digits and a sign:

-123 456 789 012 345 678 901 234 567 890 123 456 789 012 .

Even larger ***long integers*** are displayed with an exponent replacing as many of their least significant digits as necessary:

-123 456 789 012 345 678 901 234 567 890 123 456 $\times 10^{21}$.

For unsigned ***short integers***, up to 21 *bits* may be shown in large digits in binary representation:

0 1100 0010 1101 0110 0000₂ .

Base 3 (with narrow blanks every three digits) allows for displaying 20 digits and a sign:

-22 211 200 201 120 001 212₃ .

Base 4 (with narrow blanks every two digits) allows for showing 19 digits representing 38 *bits*:

3 21 23 30 22 11 21 20 32 12₄ .

Also bases 5, 6, and 7 allow for displaying 20 digits and a sign like base 3, base 8 for 19 digits like base 4 (but representing 57 *bits* in base 8).

By using the narrow digits provided, up to 25 *bits* may be shown in binary representation:

0 1110 1100 0010 1101 0110 0000₂.

Then 24 digits and a sign can be displayed for bases 3, 5, 6, and 7, as well as 22 digits for bases 4 and 8.

Longer integers in bases 2 through 6 must be displayed using the small font. This allows for showing up to 44 *bits* in binary notation:

1110 1100 0101 1101 0110 1110 1100 0010 1101 0110 0000₂.

41 digits and a sign can be displayed for base 3 being already sufficient for 64 *bits*, as well as the 39 digits theoretically displayable for base 4.

For showing the maximum of 64 *bits* in binary notation, two special 5-pixel wide characters were created:

1110 1100 0000 1101 0110 1110 1100 0000 1101 0110 1110 1100 0000 1101 0110 0000₂.

Summing up, for given base and word size, the following fonts will do for *short integers*:

Base ▼	Allowable size of digits for display			
	large	narrow	small	special
2	≤ 21 bits	≤ 25 bits	≤ 44 bits	≤ 64 bits
3	≤ 31 bits	≤ 38 bits		
4	≤ 38 bits	≤ 44 bits		
5	≤ 46 bits	≤ 55 bits		
6	≤ 51 bits	≤ 62 bits		
7	≤ 56 bits			
8	≤ 57 bits			
> 8	≤ 64 bits			

One row of four arbitrary **real matrix elements** (with absolute values < 10^{100}) takes 399 px in small font, SCI 3:

$$[-6,609 \cdot 10^{-19} \quad -6,609 \cdot 10^{-19} \quad -1,609 \cdot 10^{-19} \quad -1,609 \cdot 10^{-19}]$$

using multiplication dots. Else you will lose one decimal. A slightly different notation allows for SCI 4:

$$[-6.609 \cdot 10^{-19} \quad -6.609 \cdot 10^{-19} \quad -1.609 \cdot 10^{-19} \quad -1.609 \cdot 10^{-19}]$$

Matrices with more than four columns will need ellipses added on one or both sides:

$$[\dots \quad -6,609 \cdot 10^{-19} \quad -6,609 \cdot 10^{-19} \quad -1,609 \cdot 10^{-19} \quad \dots]$$

allowing to display a section of three elements in SCI 4 format. Using multiplication crosses will cost one decimal.

Vertically, each such matrix row requires 20 px as other small font strings do. Within the space of 3 standard numeric output rows, $3 \times 32 + 2 \times 5 = 106$ px are available, allowing for 5 matrix rows ($5 \times 20 + 4$). Thus, a 5×4 real matrix can be displayed entirely always, using SCI 3 in worst case.

In consequence, any chosen 3×3 section out of a real matrix of arbitrary size can be shown in SCI 3 minimum with surrounding ellipses. In FIX format, 8 decimals can be displayed always.

In analogy, for a **complex matrix** of arbitrary size any chosen 3×2 section can be displayed in FIX 6 format maximum with surrounding ellipses like

$$[\dots \quad -6.609 \cdot 226+i \cdot 6.609 \cdot 226 \quad -1.609 \cdot 226+i \cdot 1.609 \cdot 226 \quad \dots]$$

while displaying complex matrix elements featuring large exponents may become inconvenient very soon, regardless of the symbols used:

$$[\dots \quad -6.60E^{-199}+i \cdot 6.60E^{-199} \quad -1.60E^{-199}+i \cdot 1.60E^{-199} \quad \dots]$$

Also displaying an arbitrary 3×3 section out of a larger complex matrix is viable up to FIX 3 as long as the numbers stay in a reasonable range:

$$[\dots \quad -6,609+i \cdot 6,609 \quad -6,609+i \cdot 1,609 \quad -1,609+i \cdot 1,609 \quad \dots]$$

One row of **alphanumeric text** will typically take some 40 characters. The actual number will vary depending on their individual widths as mentioned above.

The *status bar* is a good example for such an alphanumeric row: Loaded to maximum, it might look like

2017-05-08 23:49 RLE π /3 546f 64:0 A X \downarrow C \downarrow D \downarrow U \downarrow

containing 45 characters.

Putting the alphabet in a row allows for

abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOP

i.e. 41 characters.

Echoing command input requires up to 16 characters (the 17th will close input) for a 7-character command indirectly addressing a 7-character variable entered in AIM. This can be done in either font.

Command and variable names in menus were discussed on top of p. 170 already. Although seven characters are allowed for such names, six may well fill the screen space available there. Thus, it is recommended to keep such names as short as possible, though meaningful.

Plotting?

It is mentioned elsewhere we are not out for creating a graphing calculator. There is, however, a very useful application where a basic scatter plot of measured data points would support decision making significantly (see the 3rd industrial statistics application example in *Section 2* of the OM). This would require the statistical data (e.g. max. 100 data points, i.e. 100 pairs of *x* and *y* values) to be stored point by point in a matrix, not just summed up as in the WP 34S and earlier *RPN* calculators.

The plot could be called by a command PLOT stored in STAT displaying the data points collected in a quadratic diagram. Both axes shall reach from minimum value measured to maximum value measured (plus a little extension which can be calculated based on the data

points). Axis scales are not required for analysis so I omitted them. Drawing area has to be quadratic (240×240 px for data, 242×240 px incl. the vertical axis). Hence softkeys can be positioned on one side of the diagram (max. 3×2 labels). The *status bar* would be partially overwritten by the diagram. See the sketch (to scale) for general screen layout and various data point symbols for checking visibility.



CLLCD is modified for clearing just the *LCD* section required for the diagram. – Four characters are provided in small font for ‘drawing’ the vertical axis and the 45° line:



Alternatively, both lines may be created using AGRAPH and PIXEL (see also below).

Data points can then be plotted using POINT (containing 3×3 px) – positioning them properly in the diagram, however, will require some background calculations. For POINT, also some of the control modes of HP-42S shall be implemented in analogy (the picture below is copied from the *HP-42S OM*, p. 137; settings 1 and 3 should do for our plotting):

Flag 34	Flag 35	How the AGRAPH Image is Displayed
Clear*	Clear*	The image is merged with the existing display (logical OR).
Clear	Set	The image overwrites all pixels in that portion of the display.
Set	Clear	Duplicate “on” pixels get turned “off.”
Set	Set	All pixels are reversed (logical XOR).

* Default setting.

Softkey functions could be ...

- CENTRL for fitting the center line to the data points and plotting it for checking deviations from the 45° line (some background calculations in L.R. using ORTHOF for orthogonal regression are required for the plot, setting 1 in the picture above will do while plotting),
- DEL1PT for deleting the data point located most distant from the center line fitted (could be done by plotting it once more using setting 3),
- s_{mi} for calculating the minimum standard deviation of the measuring instrument (some background calculations required again), and
- CLOSE for closing the plot screen, returning to normal display.

It may be beneficial to define a general origin for graphics at a location deviating from 0, 0 (i.e. the bottom left corner of the *LCD*) – the point 158, 0 may be a useful origin. This would allow for creating also other graphics than just the correlation diagrams mentioned above, while reserving a ‘protected screen space’ for up to six softkeys. Any user may do his own thing in this almost quadratic drawing area then, employing the provided commands AGRAPH, CLLCD, CLOSE, PIXEL, and POINT.

Stack Depth

At a very early stage of this project (2013), *stack depth* was discussed. At the bottom line, eight *registers* turn out being sufficient for solving any real-world mathematical problem. An RPL-like ‘infinite’ *stack* would allow for saving (pushing) everything thereon before calling a (sub-)routine and popping it after RTN but makes traditional R↓, R↑, and top level repetition obsolete (and FILL as well); I suggested two new commands called CLOSES and OPENS for closing the bottom section (4 or 8 *registers*) of an infinite stack for the time when R↓, R↑, FILL, and top level repetition were required, and opening it thereafter.

After all, we decided sticking to *RPN* as implemented on the *WP 34S* and *WP 31S*. It covers everything needed with minimum complexity.

Stack Lift Disabling Functions

Also these functions were subject of discussion. For sake of backward compatibility, we decided to keep them as they were on the vintage *HP RPN* pocket calculators up to the *HP-42S* (and *WP 34S* and *WP 31S*):

Only *ENTER↑*, *CLX*, $\Sigma+$, and $\Sigma-$ disable *automatic stack lift*, all other functions enable it.

Structured Programming

In 2013, I suggested the following control structures:

- IF – THEN – ELSE – END,
- FOR – FROM – TO – END,
- REPEAT – UNTIL,
- WHILE – END.

Traditional END would need to be called ENDPGM then. Later, we discussed some *PASCAL*-like structures:

- IF – THEN BEGIN – END ELSE BEGIN – END;
- FOR – DO BEGIN – END;
- WHILE – DO BEGIN – END;

We refrained from implementing such commands since we had our doubts about mixing keystroke programming and structured programming features.

UNDO

In 2013, UNDO was planned as it works in *HP-48*, recalling the stack as it was before executing last command. *WP 31S*, on the other hand, features an UNDO recalling the entire calculator status as it was. It turned out that such a complete UNDO was viable here, too, so we did it.

APPENDIX L: RELEASE NOTES

	Date	Release notes
0	29.11.12	Official project start with first publication on the forum of the MoHPC. There are found, however, far older traces of a '43S' denoting a 'Super HP-42S', though in various more or less fictional cases – pure vapourware™.
0.1	2.2.14 23.5.15	Manual setup based on the one of <i>WP 34S</i> . Passed to <i>Jake Schwartz</i> , <i>Eric Smith</i> , and <i>Richard Ottosen</i> for first information.
0.2	3.10.15	Update based on Jake's feedback and further thoughts, distributed to <i>Eric</i> , <i>Jake</i> , <i>Marcus</i> , and <i>Pauli</i> .
0.3	21.3.16	Split the manual in three; moved LBL onto the keyboard, renamed STOM to STOCFG, RCLM to RCLCFG, SERR to s_m , and SERR _w to s_{mw} ; refined the <i>Key Response Table</i> . Passed to <i>Michael Steinmann</i> for information.
0.4	28.3.16	Renamed LOGS to EXP and EEX to E . Added hardware information from 2 nd manufacturer.
0.5	29.10.16	Returned EEX . Changed the keyboard layout.
0.6	22.8.17	Merged the Applications and Owner's Manual. Changed the input order of complex number parts on Pauli's request. Changed the keyboard layout introducing D.MS, SST, BST, and % while removing ÿ, RAN#, 'FRC, and 'CFIT. Put 'CFIT into 'STAT and 'FRC into 'MODE. Placed OFF below EXIT for easier customizing. Renamed cc to C5, EEX to E , STOPW to TIMER, SHOW to REGS, 'SOLVE to 'ADV, DLINES to DSTACK, 12h to CLK12, and 24h to CLK24. Replaced IND by →. Deleted %MG since covered by Δ%, added EIGVAL and EIGVEC. Swapped CNST and CONST. Defined the echo rows for alphanumeric and command input. Expanded and modified the character sets for better use of display space. Added the QRG.

Date	Release notes
0.7 2.4.18	Changed keyboard layout. Replaced the labels BST by ≡A , SST by ≡V , and UNDO by ⬅ ; added some <i>alpha input mode</i> reminders on the keyboard. Added AGRAPH, CLLCD, EQ.xxx, HYP , J/G, M.GOTO, ORTHOF, PIXEL, POINT, TDISP, and USER . Moved the background considerations out of <i>ReM App. D</i> . Introduced K as <i>alpha register</i> for alphanumeric constants in programs. Removed <i>fraction data type</i> . Extended <i>items</i> from 6 to 7 characters to match <i>HP-42S</i> . Specified <i>data types</i> more precisely in <i>ReM App. D</i> . Reduced the maximum number of <i>local registers</i> from 888 to 100. Deleted JG1582 and JG1752. Renamed two commands for TVM. Replaced the heading apostrophe for <i>menu names</i> . Put SUMS in STAT . Renamed the trigonometric and hyperbolic functions according to mathematical standards, and CHR to CHAR . Redistributed the chapter about constants. Modified STATUS display. Refined the unit conversions to ensure <i>SI</i> on one side. Specified 0 SEED. Expanded <i>ReM App. A</i> . Added formula output for L.R. Modified CPX?, DBL?, and REAL?. Changed output of binary tests for compatibility with <i>HP-42S</i> .
0.8 7.5.18 20.9.18	Changed keyboard layout: introduced TRG containing trigonometric functions, removed HYP into EXP and IT to g-shifted 7 , swapped some shifted labels. Refined the chapters about register arithmetic, <i>Command Parameter Input</i> , <i>Alphanumeric Input</i> , <i>Matrix Calculations</i> , and <i>Orthogonal Polynomials</i> . Introduced CLCVAR and more vintage examples. Rearranged <i>temporary information</i> on the screen. Renamed REGS to RBR and CLx to CLX. Deleted ANGLE. Corrected errors and inconsistencies. Added one more example. Moved the key response table into an appendix.
0.9 3.1.19	Removed the <i>angle data type</i> . Added another industrial application and many more examples. Exchanged keyboard pictures due to changed bezel. Expanded <i>App. B</i> . Added SHOW for displaying full precision of DP numbers and FBR for browsing our two fonts. Split a chapter. Expanded some titles. Added the overlay drawing. Modified functionality of EXIT and Vx to match <i>HP-42S</i> . Added a chapter about curve fitting. Modified functionalities of ENTER† and ⬅ . Moved the printer character to f R/S . Expanded <i>App. K</i> . Renamed DOUBLE to →DP. Added →SP and conversions of <i>quarts</i> . Rearranged X.FN . Replaced USR by UM . Changed keyboard moving UM , Vx , and TRI . Added XIN and XOUT. Added a chapter in <i>App. E</i> and information about infinite integers. Extended the domain of GCD and LCM. Refined and corrected.

	Date	Release notes
0.10	3.3.19	Returned <i>angle data type</i> and α SR. Added IDIVR and VANGLE. Refined FP, IP, IMPFRC, PROFRC, SDIGS?, \rightarrow DP, \rightarrow HR, \rightarrow INT, \rightarrow REAL, \rightarrow SP, explanation of ALL, the summary of integer functions, and handling of long alpha strings. Modified contents of <u>CPX</u> , <u>MATX</u> , and <u>α</u> . Added a summary of matrix functions. Removed the <u>ON</u> -key combinations. Modified MEM?. Rewrote the angular conversions. Renamed infinite and finite integers to <i>long</i> and <i>short integers</i> . Added a chapter about $\pm\infty$ and NaN. Modified RBR and the menu for STO and RCL. Removed \blacksquare from the keyboard. Renamed X_u to X_e for the distributions. Corrected.
0.11	27.3.19	Changed keyboard making CC primary and user mode shifted, removing x^2 , $x\downarrow$, and DSP, adding $ x $, DROP, and SHOW, and moving some shifted labels. Modified <u>BITS</u> , <u>CLREGS</u> , <u>CNST</u> , <u>CPX</u> , <u>DISP</u> , <u>EXP</u> , <u>INTS</u> , <u>MODE</u> , <u>PARTS</u> , <u>SHOW</u> , <u>STAT</u> , <u>$U\rightarrow$</u> , <u>αMATH</u> , the division matrix, <i>data type</i> conversions, user mode, and the <i>Quick Reference Guide</i> . Added conversions of <i>barrels</i> , <i>carats</i> , and <i>fathoms</i> . Deleted DSP. Corrected.

DRAFT

WP 43S QUICK REFERENCE GUIDE

USING MENUS

A *menu* defines the top row of keys by displaying up to three softkeys above each . If the current *menu* has more than three rows, its current view is limited by a dashed line indicating that it is a *multi-view menu* and or can be used to display the additional views of this *menu*.

MEMORY

The **stack** is a workspace for calculations. Each **stack register** may contain any type of data. Choose a **stack** of four (**X**, **Y**, **Z**, and **T**) or eight **registers** (**X**, **Y**, **Z**, **T**, **A**, **B**, **C**, and **D**). Last x is saved in **register L**.

General purpose registers: There are 100 numbered global general purpose registers (00 ... 99). And there are **I**, **J**, and **K** serving special purposes in matrix handling (see p. vii), probability distributions (see p. viii), and programming but may be used globally otherwise. Also **A**, **B**, **C**, and **D** may be used this way unless being part of **stack**. Each **register** may contain any type of data. **STO nn** stores a copy of x into **Rnn**, **RCL nn** recalls a copy of the contents of **Rnn** into **X**, and **xz nn** swaps x and the contents of **Rnn**.

Variables are named storage locations that may contain any type of data. E.g. for storing x into a variable named **XYZ**, enter **STO @ XYZ ENTER↑**. Variable names shall be unique, ≤ 7 characters long, and contain ≥ 1 letter.

Flags: There are 112 global user **flags**. Of these, the following have a special meaning if set:

- | | | |
|-----|----------|--|
| 103 | T | sets print mode to Tracing |
| 105 | B | Overflow |
| 106 | C | Carry |
| 107 | D | allows for infinite and non-numeric results ("Danger") |
| 109 | I | allows for complex results |

Programs consist of ≥ 3 program steps: LBL with a global label, at least one action step, and RTN. Each program may contain subroutines (up to 8 levels deep). See p. v for more.

Available memory: **INFO MEM?** (or **FLAGS STATUS**) displays the amount of free memory. Use CLP for clearing programs or clear variables to deallocate memory that is no longer needed.

DATA TYPES

Long integers are the simplest type of data. Any number you enter without using **0** or **E** is regarded a *long integer* of base 10.

Real numbers: Any number you enter using **.** and/or **E** is a real number.

Complex numbers: A complex number consists of two real numbers combined to represent its real and imaginary part like $1.23-i\times 4.56$ in rectangular mode (set RECT and press **1.23** **CC** **4.56** **Hz**) or its magnitude and angle like $7.89 \angle -120^\circ$ in polar mode (set POLAR and press **7.89** **CC** **120** **Hz**).

Angles: Any input of a real number trailed by **d.ms** is interpreted as an *angle* in *sexagesimal degrees*. Alternatively, *angles* may be entered as well in *decimal degrees*, *radians*, *multiples of π* , or *gradians*. Choose the appropriate angular display mode via **MODE** (see overleaf).

Times: Any input of a real number trailed by **h.ms** is interpreted as a *sexagesimal time*. It will be displayed like $23:45:43.210\ 9$ with as many decimal *seconds* as needed.

Dates: Any input of a real number trailed by **d** is interpreted as a *date* in the format selected (yyyy.mmdd for Y.MD or dd.mmyyyy for D.MY or mm.ddyyyy for M.DY).

Matrices: see the dedicated paragraph on p. vii.

Short integers: Any numeric input trailed by **#** and a legal base is interpreted as a *short integer* of the base specified. **D** and **H** are shortcuts for base 10 and 16, respectively. *Short integers* may occupy 1 ... 64 bits.

Alphanumeric strings: Any data entered in *alpha input mode* (AIM) become such strings when closed (unless they are function parameters). Enter AIM by pressing **@**. International (e.g. accented) letters are found in **g** **+**. Greek letters are accessed via **g** plus the corresponding Latin letter. Switch to lower case by **▼** and back to upper by **▲** for all these letters.

Use **f** plus one of the keys **+**, **-**, **x**, **/**, **,**, and **0** – **9** to enter the corresponding character. Special characters are found in **g** **-** and **g** **.**

f R↓ makes the subsequent character entered a subscript, **f E** makes it a superscript, if applicable.

MODES

	SETSIG				LZOFF		
	1COMPL	2COMPL	UNSIGN	SIGNMT	LZON	WSIZE	
MODE	DENMAX	DENANY	DENFAC	DENFIX	SSIZE4	SSIZE8	
	FAST	SLOW	RM	QUIET	REALRE	CPXRES	
	DEG	RAD	GRAD	MUL π	RECT	POLAR	

- DEG Selects *degrees* as angular display mode (*ADM*).
 RAD Selects *radians* as *ADM*.
 GRAD Selects *gradians*, a.k.a. *gon*, as *ADM*.
 MUL π Selects *multiples of π* as *ADM*.
 RECT Sets rectangular format for displaying complex numbers.
 POLAR Sets polar format for displaying complex numbers.
 FAST Sets processor to normal speed.
 SLOW Reduces processor speed (approx. 2 times slower than fast).
 RM Sets rounding mode.
 QUIET Disables or enables the beeper.
 REALRE Allows only real results.
 CPXRES Allows also complex results of real calculations (e.g. $\sqrt{-1}$).
 DENMAX Sets the maximum denominator for calculating with fractions.
 DENANY Any denominator up to DENMAX is legal.
 DENFAC Any integer factor of the maximum denominator is legal
 DENFIX Sets DENMAX as fixed denominator.
 SSIZE4, SSIZE8 Sets stack size to 4 (or 8) *registers*.
 SETSIG Sets calculator precision (1 ... 34 significant digits).

For the commands 1COMPL through WSIZE see *OPERATIONS ON SHORT INTEGERS* on p. xii.

Display Formats

	CHINA	EUROPE	INDIA	JAPAN	UK	USA	
	GAP					DSTACK	
	SCI0V0R	ENG0V0R	MULTx	MULT-	RDX.	RDX,	
DISP	SDL	SDR			RDP	RSD	
	FIX	SCI	ENG	ALL	ROUND1	ROUND	

FIX	Fixed number of decimals.
SCI, ENG	Scientific (or engineering) notation.
ALL	Displays all digits required as far as possible.
ROUND	Rounds a <i>time</i> , real, or complex <i>x</i> to current display format.
ROUNDI	Rounds to next integer.
RDP	Rounds <i>x</i> to <i>n</i> decimal places (1 ... 99, think of FIX)
RSD	Rounds <i>x</i> to <i>n</i> significant digits (1 ... 34, think of SCI).
SDL, SDR	Shifts digits left (right) by <i>n</i> decimal positions.
SCIOVR, ENGOVR	Specify the display format if ALL is not viable anymore.
MULT \times , MULT \cdot	Select the multiplication symbol.
RDX., RDX,	Select the radix mark.
GAP	Selects a digit group gap inserted after every <i>n</i> digits.
DSTACK	Sets the number of <i>stack registers</i> displayed (1 ... 4).
CHINA, EUROPE, INDIA, JAPAN, UK, USA	Set local display preferences.

EXECUTING FUNCTIONS AND PROGRAMS

Any function or program can be executed via **[XEQ] α name [ENTER]** where **name** is the function name or the program label. If **name** is not unique, the global label closest to the permanent end (.END.) has precedence. If **name** is a local label, WP 43S searches the current program only.

Smart program menu: **[XEQ] PROGS** displays all programs (actually: global labels) defined. Specify the required program by pressing the corresponding softkey.

Single stepping: To execute the current program step, press **$\equiv \triangleright$** (or **\blacktriangleright** if no multi-view menu is displayed).

The Run/Stop key: Pressing **R/S** runs the current program beginning with the current step or stops a running program after the current step is executed completely.

The catalog of functions: Browse **CATALOG FCNS** and execute the required function by pressing the corresponding softkey. This catalog can be searched alphabetically.

Specifying Function Parameters

Numeric parameters: Functions accepting numeric parameters prompt you with a cursor for each digit expected. To key in a numeric parameter, just enter its digits. If you provide a digit for each underscore, the function will

execute. You can also provide less digits and complete input with **ENTER↑**.

Alphanumeric parameters: Many functions accept alphanumeric parameters as well. The parameter you want will often be an object already existing, so your WP 43S will display a *menu* for quick entry. If it does not exist yet, type it. E.g. for creating a variable **ABC** just type **STO** **α** **ABC** **ENTER↑**.

Stack parameters: Any function accepting a ‘usual’ *register* as parameter also accepts a *stack register*. Just press the corresponding softkey for **X** ... **T** or the keys in second row for **A** ... **D**, if applicable.

Indirect addressing: Rather than providing an actual parameter, you can specify the variable or *register* containing the parameter. Just press the softkey **→**. E.g. to display the contents of the variable or *register* specified in **R12**, key in **VIEW** **→** **12**. This works with *stack registers* as well.

CLEARING AND DELETING

CLR	CLREGS	CLPall	CLFall	CLCVAR	CLLCD	CLall	RESET
	CLΣ	CLP	CF	CLMENU	CLSTK	CLX	

- CLΣ Clears all statistical data.
CLP Clears (deletes) the *current program*.
CF *n* Clears *flag n*.
CLMENU Clears the *programmable menu*.
CLSTK Clears the entire *stack* (i.e. fills all its *registers* with zero).
CLX Clears *stack register X*.
CLREGS Clears all *registers* (except the *stack* and statistical data).
CLPALL Clears (deletes) all programs in *RAM*.
CLFALL Clears all user *flags*.
CLCVAR Clears all *variables* used in *current program*.
CLLCD Clears the *LCD* above and to the right of pixel *x,y*.
CLALL Clears everything but the modes set.
RESET Resets the WP 43S to *startup default*.

- CATALOG** **VARS** ...   allows for deleting the variable selected.
CATALOG **MENUS** ...   allows for deleting the user *menu* selected.
CATALOG **PROGS** ...   allows for deleting the program selected.

PROGRAMMING

Program Entry

- P/R** toggles *program entry mode*.
- GTO** [] moves the *program pointer* to a new program space.
- GTO** [] *nnnn* moves it to step number *nnnn*.
- [] moves it to previous step
[] moves it to next step (use [] or [] if no *multi-view menu* is displayed).
- [] deletes the *current program step* entirely.

Labels

A program label is a marker used to identify an entire program or a section within a program. Each program must begin with a global label.

Global labels can be accessed from anywhere in memory (thus, they should be unique). Global labels are alphanumeric and up to 7 characters long.

Local labels can be accessed only within the current program (thus, they should be unique within this program). Local labels are numeric (00 ... 99).

Local registers

... are allocated via **LOCR n** with the amount of *registers* specified (≤ 100). 16 *local flags* come with them. Local data are valid in the calling routine only.

Tests (Do if True, Skip if False)

When a test step (a.k.a. a conditional function) is executed, the program step immediately following said conditional is executed if the condition is “true”; if the condition is “false”, the step following the conditional is skipped.

Looping

ISE, ISG, ISZ, DSE, DSL, and DSZ (found in LOOP) control looping. Each accesses a variable or *register* containing a loop control number in the form ccccccc.ffffii with ccccccc being the current counter value, fff the final counter value, and ii the increment (or decrement) size (default is 1). As long as the count is not complete, the step following the instruction is executed (usually a branch to the top of the loop). E.g. the program segment pictured here counts from 1 to 52 by threes (executing the loop 18 times) and then beeps.

```
...
1.05203
STO "Count"
LBL 01
...
ISG "Count"
GTO 01
BEEP
...
```

USING A VARIABLE MENU

A *variable menu* may be displayed by the *Solver* or the numeric integrator or by VARMNU within a program. Each label in the *menu* represents a variable. While the *menu* is displayed, you can:

Store a value into a variable: Key in the value and then press the softkey.

Recall the contents of a variable: Press **RCL** and then the softkey.

View the contents of a variable without recalling it: Press **VIEW** and then the softkey.

Select a variable: Press the corresponding softkey without keying in a number first. (For the *Solver*, this is how you select the unknown variable; for the integrator, this is how you select the variable of integration.)

You can call and use any function *menu* without exiting from the *variable menu*.

MATRIX OPERATIONS

A matrix is an array with **m** rows and **n** columns of real or complex elements.

To create a new $m \times n$ matrix, enter its dimensions (**m** **ENTER↑** **n**) and press

MATX NEW for a matrix in **X** or

MATX DIM **@ name** **ENTER↑** for a matrix in a variable. If the variable already exists, DIM re-dimensions it.

For editing the matrix in X, use **MATX EDIT** .

For editing a named matrix, use **MATX EDITN name**.

When a matrix is being edited it is said to be *indexed* (to index a named matrix without editing it, use INDEX). Whenever there is an indexed matrix, two pointers are used to indicate the row and column of the current element: they are stored in **I** and **J**, respectively. If **I** and **J** are pointing to the last element (bottom right) in a matrix and you press **→** then ...

- ...the pointers wrap around to the first element of the matrix (**Wrap mode**, automatically set whenever you enter or exit the *Matrix Editor*) or ...
- ...the matrix grows by one complete row and the pointers move to the new row (**Grow mode**).

WRAP and GROW are in the **f**-shifted row of the *Matrix Editor menu*.

Matrix arithmetic: **[+]**, **[-]**, **[×]**, and **[÷]** work for matrices just as for individual numbers. Advanced functions often operate on the individual matrix

elements. Anytime a matrix is used in a mathematical operation with a complex object, the result will be a complex matrix.

To solve a system of simultaneous linear equations represented by the matrix equation $(A)\vec{X} = \vec{B}$:

1. Key in **[MATX] SIM EQ n** with **n** being the number of unknowns. Your WP 43S will automatically create or re-dimension the matrix variables **Mat_A**, **Mat_B**, and **Mat_X**.
2. Press **[Mat A]**; fill the matrix; press **[EXIT]**.
3. Press **[Mat B]**; fill the matrix; press **[EXIT]**.
4. Press **[Mat X]** to compute the solution matrix.

PROBABILITY

	RAN#	SEED				$\Gamma(x)$	
PROB	Binom:	Geom:	Hyper:	NBin:		Poiss:	
	Norml:	LgNrm:	Cauch:	Expon:	Logis:	Weibl:	
	t:	$\Phi:$	C_{yx}	P_{yx}	F:	$\chi^2:$	
Binom:	Binom_p	Binom			Binom_e	Binom⁻¹	

- C_{yx} , P_{yx} Returns the number of possible combinations (or permutations, a.k.a. arrangements) of x items taken out of a set of y items.
 RAN# Returns a random real number between 0 and 1.
 SEED Stores a seed for RAN#.
 $\Gamma(x)$ Returns a *Gamma function* value.

The 15 continuous (c) and discrete (d.) distributions (d.) provided:

- | | |
|-----------|--|
| $\Phi:$ | c Standard normal (or Gaussian) d. with $\mu=0$ and $\sigma=1$. |
| $t:$ | c Student's t d. (i = degrees of freedom (dof)) |
| $F:$ | c Fisher's F d. ($i = dof_1$, $j = dof_2$) |
| $\chi^2:$ | c Chi-square d. ($i = dof$) |
| Norml: | c (General) normal d. ($i = \mu$, $j = \sigma$) |
| LgNrm: | c Log-normal d. ($i = \mu$, $j = \sigma$) |
| Cauch: | c Cauchy-Lorentz (a.k.a. Breit-Wigner) d. (i = location, j = shape) |
| Expon: | c Exponential d. (i = rate) |
| Logis: | c Logistic d. ($i = \mu$, j = scale parameter) |
| Weibl: | c Weibull d. (i = shape, j = characteristic lifetime) |

Geom:	d	<i>Geometric distribution</i>	($i = p_0$ = gross probability of a success)
Binom:	d	<i>Binomial d.</i>	($i = p_0, j = n$ = sample size)
Hyper:	d	<i>Hyperbolic d.</i>	($i = p_0, j = n, k$ = batch size)
NBin:	d	<i>Negative Binomial d.</i>	($i = p_0, j = n$)
Poiss:	d	<i>Poisson d.</i>	(i = Poisson parameter)

Following naming convention holds for most distributions, e.g. for the *normal d.*: **Norml_p** denotes the *probability density function*, **Norml_c** the *cumulated d. function*, **Norml_e** the *error probability*, and **Norml_q**⁻¹ the *quantile function*. Store the required d. parameters in **I**, **J**, and **K** as listed above; the remaining parameter must be given in **X** before calling the respective function – note the *quantile function* requires a probability in **X** ($0 \leq x \leq 1$).

STATISTICS

Statistical data are accumulated in 14 dedicated *registers*, separate from all the other *registers* introduced above.

Clear the statistical registers before doing a new stat. analysis: **STAT CLΣ**.

Then, accumulate the data:

- For each x-y data pair: **y-value [ENTER↑ x-value Σ+]**.
- For each single-point data value: **x-value Σ+**.
- For each weighted data point: **weight-value [ENTER↑ x-value Σ+]**.
- For x-y data pairs stored in a two-column matrix (x-values in column 1, y-values in column 2): Place the matrix in **X** and then press **Σ+**.

To undo input mistakes or remove erroneous data,

- either press **⬅** (for the very last data point)
- or recall the (earlier) incorrect y and x data in the *stack* and press **Σ-**.

Data Evaluation and Analysis

	...						
	LinF	ExpF	LogF	PowerF		BestF	
	OrthoF		cov				
	L.R.	r	s _{xy}	ŷ	ŷ	x ²	
STAT	CLΣ	Ȑ _G	ε	ε _p	ε _m	PLOT	
	Σ-	Ȑ _w	s _w	G _w	s _{mw}	SUM	
	Σ+	Ȑ	s	G	s _m	x ²	

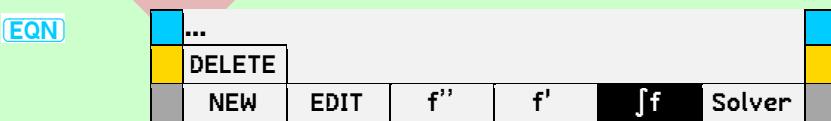
\bar{x} , s , σ , s_m	Arithmetic mean value, sample standard deviation (SD), population SD , standard error (a.k.a SD of the mean).
\bar{x}_w , s_w , σ_w , s_{mw}	Same for weighted data.
\bar{x}_g , ε , ε_p , ε_m	Geometric mean value, sample scattering factor (SF), population SF , SF of the mean.
SUM	Returns Σx and Σy .
PLOT	See the <i>ReM, App. D.</i>
L.R.	Computes the parameters a_0 and a_1 of the fit model selected (see below). For linear fits, a_0 is the y-intercept and a_1 the slope.
r	Returns the correlation coefficient.
s_{xy} , cov	Return the sample or population covariance.
\hat{x}, \hat{y}	Return the forecast for x or y following the fit model selected.
LINF	Linear fit model: $y = a_0 + a_1 x$.
EXPF	Exponential fit model: $\ln(y) = \ln(a_0) + a_1 x$ or $y = a_0 e^{a_1 x}$.
LOGF	Logarithmic fit model: $y = a_0 + a_1 \ln(x)$.
POWERF	Power fit model: $\ln(y) = \ln(a_0) + a_1 \ln(x)$ or $y = a_0 x^{a_1}$.
BESTF	Blindly selects the model returning the best correlation coefficient.
ORTHOF	Works like LINF but assumes equal errors in x and y. ORTHOF is not part of the pool BESTF investigates.

ADVANCED OPERATIONS

EQN is for interactive editing, storing, recalling, solving, integrating, and deriving equations.

ADV is for programmed summing, multiplying, solving, integrating, and deriving.

Interactive Operations on Equations



For creating a new equation, press **NEW**. The *Equation Editor menu* will open, and the blue row will display the current equation. Press **EXIT** when finished.

For browsing existing equations, press **▲** or **▼**. The equation displayed in **9**-shifted row is called the *current equation*.

For editing (or deleting) the current equation, press **[EDIT]** (or **[DELETE]**).

For operating on the current equation, press the respective softkey. A *menu* will pop up displaying the names of all variables used and more.

Using Advanced Operations in Programs

ADV	PGMSLV	f''(x)	ʃ f dx		
SOLVE	SLVQ	f'(x)	Π	Σ	

SLVQ solves the quadratic equation $ax^2 + bx + c = 0$ with its parameters on the input stack [**c**, **b**, **a**, ...]. It returns two real or complex solutions.

Π label calculates the product of the terms given by the routine specified, using the loop control number given in **X** (cf. p. vi).

Σ label calculates the sum of the terms given by the routine specified, using the loop control number given in **X** (cf. p. vi).

SOLVE var solves for an unknown variable in an expression, given values for all the other variables. The expression $f(x_1, x_2, \dots)$ shall be written as a program (let's call it **AB**, for example):

- **AB** must begin with a global label.
- The body of **AB** shall evaluate the expression. For an expression to be solved, it must be coded that $f(x_1, x_2, \dots) = 0$ is fulfilled. Recall the variables of the expression as they are required and calculate f .
- **AB** must logically end with RTN.

Then write a program calling the *Solver* (let's call it **CD**, for example). At the position where you need the expression solved, press **[ADV]**:

1. Press **PGMSLV** and specify **AB**.
2. Store a value into each known variable, e.g. using **STO**. Optionally store a guess into the unknown variable to direct the *Solver* to a solution.
3. Press **SOLVE** and specify the unknown variable.

When running **CD** later on, **SOLVE** will solve for the unknown.

f'(x) (or **f''(x)**) calculates the first (or second) derivative of $f(x)$ at location x . The function $f(x)$ shall be written as a program (e.g. called **EF**); it must begin with a global label, take care of all variables used, and evaluate $f(x)$.

Then write a program calling the derivator (let's call it **GH**, for example).

1. Store a value into each of the variables that shall remain constant under derivation.
2. At the position where you need the derivative, put the respective location into **X**, then press **ADV** $f'(x)$ (or $f''(x)$) specifying **EF**.

When running **GH** later on, the derivative will be returned in **X**.

ffd var numerically computes a definite integral. The integrand $f(x)$ shall be written as a program (e.g. called **IJ**); it must begin with a global label, recall all integration constants used, and evaluate $f(x)$.

Then write a program calling the integrator (let's call it **KL**, for example). At the position where you need the integral, press **ADV**:

1. Press **[fdx]**. A submenu will open.
2. Press **PGMINT** and specify **IJ**.
3. Store a value into each of the variables that shall remain constant under integration, e.g. using **STO**.
4. Store the lower limit ($\downarrow LIM$), the upper limit ($\uparrow LIM$), and the accuracy factor (ACC).
5. Press **J** and specify the variable of integration.

When running **KL** later on, the integral will be returned in **X** and the uncertainty of computation will be returned in **Y**.

OPERATIONS ON SHORT INTEGERS

						LZOFF	
						WSIZE	
						RJ	
	LJ						
	SL	RL	RLC	RRC	RR	SR	
BITS	SB	BS?	#B	FB	BC?	CB	
	NAND	NOR	XNOR		MIRROR	ASR	
	AND	OR	XOR	NOT	MASKL	MASKR	

AND, OR, XOR, NAND, NOR, XNOR Boole's binary operators.

NOT Inverts every bit in **X**.

MASKL, MASKR Create masks of x bits on the left (or right) side.

MIRROR Reflects all bits.

ASR n Arithmetic shift x right by n places.

SB, FB, or CB n Sets, flips, or clears bit # n in x .

BS?, BC?	Checks if bit # <i>n</i> in <i>x</i> is set (or clear).
#B	Returns the number of bits set in <i>x</i> .
SL, SR <i>n</i>	Shift <i>x</i> left (or right) by <i>n</i> places.
RL, RR <i>n</i>	Rotate <i>x</i> left (or right) by <i>n</i> places.
RLC, RRC <i>n</i>	Rotate <i>x</i> left (or right) by <i>n</i> places through Carry.
LJ, RJ	Adjust the bits set in <i>x</i> to the left (or right).
1COMPL, 2COMPL	One's (two's) complement mode.
UNSIGN	Unsigned mode.
SIGNMT	Sign-and-mantissa mode.
WSIZE	Sets the word size (1 ... 64 bits).

INTS	DBL /	DBLR	DBL ×	^MOD	CEIL	GCD	
	IDIV	RMDR	MOD	×MOD	FLOOR	LCM	
	A	B	C	D	E	F	

A ... F	Digits for bases >10.
IDIV	$\mathbb{R} \mathbb{Z}$ Integer divide – works for real numbers (\mathbb{R}) and long integers (\mathbb{Z}) as well.
RMDR, MOD	$\mathbb{R} \mathbb{Z}$ Remainder and modulo.
×MOD, ^MOD	$\mathbb{R} \mathbb{Z}$ Returns $(z \cdot y) \bmod x$ or $(z^y) \bmod x$, respectively.
FLOOR	\mathbb{R} Returns the greatest integer $\leq x$.
CEIL	\mathbb{R} Returns the smallest integer $\geq x$.
LCM	\mathbb{Z} Returns the least common multiple of <i>x</i> and <i>y</i> .
GCD	\mathbb{Z} Returns the greatest common divisor of <i>x</i> and <i>y</i> .
DBL /, DBLR, DBL×	Double word length commands for division, remainder, and multiplication.

OPERATIONS ON ALPHANUMERIC STRINGS

Connect strings by pressing . Then *x* will be appended to the string *y*. With numeric data in X, their current display format is taken into account.