



## WP43S 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 [https://gitlab.com/Over\\_score/wp43s](https://gitlab.com/Over_score/wp43s)

Copyright © 2015 - 2020 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. 163 and bottom of p. 164 were kindly supplied by *SwissMicros* as well as the drawing on p. 219, the picture on p. 215 by *Martin Lorang*. The plots in *Appendix H* are based on material found in *Wikipedia*. The other pictures, diagrams, and graphics were created by the author.

Internet addresses are specified as found and verified at 2019-06-26. 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 (*IOI*)

13

0 - 9	17
A	18
B	22
C	24
D	31
E	35
F	38
G	40
H	43
I	44
J	47
K	47
L	49
M	53
N	59
O	61
P	62
R	65
S	73
T	82
U	84
V	85
W	86
X	87
Y	90
Z	91

A, α	91
β	93
Γ, γ	93
Δ, δ	93
ε	94
ζ	94
λ	94
μ	95
Π, π	95
Σ, σ	95
Φ	97
X	98
ω	98
(, +, -, ×, /, ^	98
→,	99
%	102
The Rest	103
 Names of Variables and System Flags Provided	107
System Flags	112
Nonprogrammable Commands and Keys	116
Command Parameter Input and Closing It	117
Alphanumeric Input in X and Closing It	119
 <b>Section 2: Menus and Catalogs</b>	<b>123</b>
One to Find and Rule Them All – the CATALOG	124
Accessing Cataloged Items Rapidly	127
Further Menus and Their Contents	129
Constants	138
Unit Conversions	148
 <b>Section 3: Calling and Executing Operations</b>	<b>156</b>
Using XEQ for Executing Operations	156

Operations Requiring Trailing Parameters	157
Operations Changing Data Types	159
<b>Appendix A: Hardware</b>	<b>162</b>
<b>Appendix B: Memory Management</b>	<b>167</b>
Data Types	167
Statistical Summation Registers	170
Range of Real Numbers	170
Numeric Limitations	175
Special Results	176
Program Step Size	181
<b>Appendix C: Messages and Error Codes</b>	<b>182</b>
<b>Appendix D: Comparison to the Function Sets of <i>HP-42S</i>, <i>HP-16C</i>, <i>HP-21S</i>, and <i>WP 34S</i></b>	<b>188</b>
Corresponding Operations on <i>HP-42S</i>	188
Corresponding Operations on <i>HP-16C</i>	194
Corresponding Operations on <i>HP-21S</i>	196
Corresponding Operations on <i>WP 34S</i>	198
New Commands on your <i>WP 43S</i>	202
Reference Literature	207
<b>Appendix E: Emulating a <i>WP 43S</i> on Your Computer</b>	<b>210</b>
<b>Appendix F: Flashing and Updating Your <i>WP 43S</i></b>	<b>214</b>
How to Flash Your <i>WP 43S</i>	214
How to Update Your <i>WP 43S</i>	218
Overlays	219

<b>Appendix G: Troubleshooting Guide</b>	<b>220</b>
Calculator Frozen	220
Keymap Trouble	221
<b>Appendix H: Advanced Mathematical Functions and Tasks</b>	<b>222</b>
Number Generating Functions	222
Statistical Distributions	224
More Statistical Formulas, also for Fitting	233
About the Curve Fitting Models Provided	239
About Error Propagation	246
Solving Differential Equations	247
Orthogonal Polynomials	251
Even More Mathematical Functions	256
<b>Appendix I: Information for Advanced Users</b>	<b>261</b>
Recursive Programming	261
Building <i>WP 43S</i> Almost from Scratch	262
Index of Everything Provided	264
<b>Appendix J: Release Notes</b>	<b>276</b>
<b>WP 43S Quick Reference Guide</b>	<b>Q-1</b>
USING MENUS	1
MEMORY	1
DATA TYPES	1
MODES	3
DISPLAY FORMATS	3
EXECUTING FUNCTIONS AND PROGRAMS	4
CLEARING AND DELETING	5
PROGRAMMING	5
MATRIX OPERATIONS	7

PROBABILITY	8
STATISTICS	9
ADVANCED OPERATIONS	10
OPERATIONS ON SHORT INTEGERS	12
OPERATIONS ON ALPHANUMERIC STRINGS	13
<b>Background Considerations and Facts</b>	<b>B-1</b>
Accessing Items	1
Alpha Register	3
Angles	4
Backward Compatibility	4
Calculation Internals	5
Character Sets	5
Complex Infinities	7
Complex Notation and Storage	7
Display Limits	9
Display Segmentation	15
Echo and Fallback	16
Equations	17
Layouting	17
Menus	18
Number Range	19
Plotting?	19
Precision and Accuracy	21
Prefixes	22
Sorting in Detail	22
Stack Size	28
Stack Lift Disabling Functions	28
Structured Programming	29
UNDO	29

# 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 over a third 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.

The 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 text font is Arial. Emphasis is added by underlining or **bold** printing. Calculator COMMANDS, MENUS, PREDEFINED VARIABLES and SYSTEM FLAGS are generally called by their *names*, printed capitalized in running text (*menus* underlined). Quoted text is printed blue (as well as translator's footnotes). 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 .pdf file – it cannot work in a printed copy for obvious reasons; thus such a link generally refers to a page number, to the [Table of Contents](#)

, or to a fully specified external address.

- Bold italic Arial letters such as *n* are used for variables; bold normal letters for constant **sample values** (e.g. labels, numbers, or characters).
- Courier is used for file names, binary and hexadecimal codes, and describing numeric formats.
- Times New Roman regular letters are for unit symbols and for mathematical functions. Italics are for *unit names* in running text.
- 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 stored therein.
- This **[KEY]** font (created by Luiz Vieira of Brasil) is taken for references to calculator keys, including **(SOFTKEYS)** in general. For shifted operations like **GTO** or **LBL**, the respective color is used. **Alphanumeric** and numeric calculator outputs (like  $1.234 \times 10^{-56}$  or  $7,089 \cdot 10^{-12}$ ) are printed as you see them on the calculator screen.
- We will use decimal points in most parts of this manual (but you may set your WP 43S to decimal 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*).

*BCD* = binary coded decimal.

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

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

*HP* = Hewlett-Packard.

*IOI* = *Index of Items* (see pp. 13ff).

*LCD* = liquid crystal display.

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

*OH* = Owner's Handbook.

*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*).

*SRS* = subroutine return stack (see *App. B* on pp. 167ff).

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

Some more abbreviations may be used and explained locally.

DRAFT

## **SECTION 1: INDEX OF ITEMS (IO)**

All the *items* provided on your WP 43S (more than 850) are listed below with their *names* (as they are displayed and printed in routines) in column 1 and the keystrokes necessary to call them. Most *items* shall be picked from *menus* (see pp. 123ff). For such *items*, we list the keys calling the respective *menu*, the *prefix* of the respective *menu* row (if applicable), and the label as shown therein; we are confident you will find the corresponding softkey. *Items* stored in CONST are listed with their *names* only, however, since they are sorted alphabetically and will be explained in detail in a separate chapter below.

There is an important difference between the *names* of *items* and their labels as printed on the bezel or displayed in menus:

Each item provided is identified by its unique reserved name of up to 7 characters – it may be accessible under one or more different labels, featuring less or more characters than its name (see some unit conversions, for example). These labels are not required to be unique.<sup>1</sup>

On your WP 43S, sorting (e.g. of *names*) works in the following order:  
2

1



Accented letters follow their parents, as do superscripts and subscripts.

In principle, *WP 43S* operations work as the corresponding ones did on the *WP 34S* where applicable (see App. E). Referring to vintage

<sup>1</sup> Actually, there are two separate sets of *items*: set 1 contains commands, constants, *menus*, program labels, and reserved symbols; set 2 is for *registers*, *system flags* and variables. The *name* of an *item* must be unique in its set.

<sup>2</sup> Characters printed on grey background are inaccessible for users for the time being. The entire sorting table is printed in an appendix.

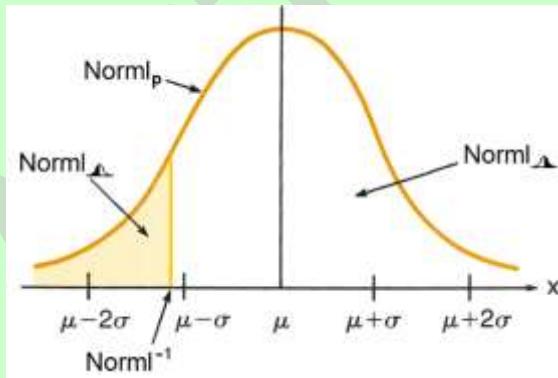
calculators, most functions and keystroke-programming will work as they did on the *HP-42S*, bit and integer functions as on the *HP-16C*, unless specified otherwise. Also for functions inspired by other vintage calculators as mentioned in the index below, their manuals may contain helpful additional information.

Some 300 functions featured in your *WP 43S* are new compared to *HP's RPN* pocket calculators. Operations carrying familiar *names* but deviating in their functionality from previous *HP RPN* calculators or the *WP 34S* are marked **light red**.<sup>3</sup>

Operations working with the accumulated statistical data are marked **light blue**. For probability distributions, the naming rules are as pictured here.

Operations asking you for confirmation are printed **red**.

All operations may be also entered in *PEM* unless marked **violet** – on the other hand, the majority of functions contained in P.FN and TEST carry most use in *PEM*.



For the vast majority of operations, their remarks in the table 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*.

<sup>3</sup> We did not compare the *RPL* calculators of the last three decades nor the *HP Prime*. They are exceeding the realm of shirt pocket calculators.

Note some functions overwrite two stack levels instead of pushing two values on it: e.g. →POL and →REC, as you may have expected.<sup>4</sup>

Operation or function **parameters** will be taken from the lowest stack register(s) unless mentioned explicitly in second column of the *IOI* – 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 addressing tables in *Section 1* of the *OM*); and
  - n** stands for the respective number itself;

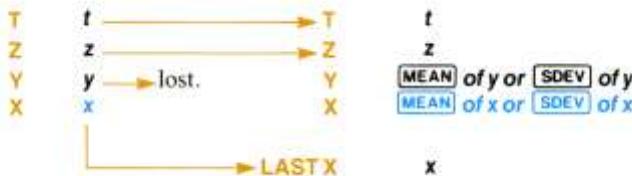
**Example:** RSD 12 rounds  $x$  to 12 significant digits, while

---

<sup>4</sup> 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...

... from this ...      ... to this.



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).

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 program 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 input parameter, you can enter XYZ → ST.X and it will take its parameter from X instead – like a good old traditional RPN command.

The *data types* a particular function operates on are listed in { } under “remarks” if there are restrictions – cf. *App. B* on pp. 167ff. 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.<sup>5</sup>

---

<sup>5</sup> This applies for °C → °F, for instance: For *real*/number input, output will stay *real*. For integer input, however, output will be *real*.

Some functions operating on *long integers* will return either such integers or *reals*, depending on the input value. E.g.  $\sqrt[3]{x}$  will return 3 for an input of 27, i.e. for a proper

*Automatic stack lift* is enabled after each command except after CLX, ENTER $\uparrow$ ,  $\Sigma+$ , and  $\Sigma-$  (cf. Section 1 of the OM); thus, numeric input immediately following one of these four operations will overwrite  $x$  instead of pushing it on the *stack* as usual.<sup>6</sup>

Below, the functions checked already are highlighted green, those which didn't work yet (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*.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$ etc.	(1) {2}; {1} $\rightarrow$ {2}
$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$		Convert temperatures. See pp. 148ff.
$10^x$		(1) {1, 2, 3, 8*, 9*, 10} Returns $10^x$ , the inverse of $\lg(x)$ .
1COMPL	1COMPL 1COMPL	(0) Sets 1's complement mode for operations on <i>short integers</i> . Indicated in the <i>status bar</i> . See Sect. 2 of the OM.
$1/x$		(1) {2, 3, 8*, 9*}; {1} $\rightarrow$ {2} Inverts the number $x$ or all elements of the matrix $x$ .

cube, but will return a *real* for an input of 28 although this is a *long integer* as well. The same function operating on a *short integer* will return 3 for both cases, in whatever base applicable. See the OM, Section 2, *Integers: Summary of Functions*.

<sup>6</sup> Some reasoning why *automatic stack lift* is disabled for these four:

- a) CLX is for clearing **X** to make room for a corrected value. This value shall overwrite  $x$  – an extra zero on the stack would make no sense.
- b) ENTER $\uparrow$  is a *stack lift* manually initiated by the user. An additional *automatic stack lift* immediately after this command would make no sense.
- c)  $\Sigma+$  and  $\Sigma-$  are dedicated commands for adding or subtracting data points (see the chapter about *Statistical Calculations* in Section 2 of the OM). These two commands were exclusively designed for data input since their first appearance on the HP-45 and are not really meant to be mixed with calculations.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
2COMPL	f BITS ▼ 2COMPL g INTS ▲ 2COMPL	(0) Sets 2's complement mode for operations on <i>short integers</i> . Indicated in the <i>status bar</i> . See Sect. 2 of the OM.
$2^x$	g EXP $2^x$	(1) {1, 2, 3, 8*, 9*, 10} Returns $2^x$ .
$\sqrt[3]{x}$	g EXP f $\sqrt[3]{x}$	(1) {1, 2, 3, 8*, 9*, 10}; ({1} → {2}) Returns the cube root of $x$ . Roots of non-cube <i>long integers</i> will return <i>reals</i> .
a	f CONST a	(-1) {} → {2}
$a_0$	f CONST $a_0$	<i>Gregorian year in days</i> and <i>Bohr radius in meter</i> .
ABS	f CATALOG FCNS ABS	Points to $ x $ on p. 102. Maintained for backward compatibility only.
ACOS	f CATALOG FCNS ACOS	Points to arccos on p. 20.
$ac \rightarrow m^2$	f U→ f A: acre → $m^2$	(1) {2}; {1} → {2} Convert areas. See pp. 148ff.
$ac_{us} \rightarrow m^2$	f U→ f A: acre <sub>us</sub> → $m^2$	
ADV	f ADV	<i>Menu</i> . See p. 130.
AGM	g X.FN AGM	(2) {2, 3}; {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. 256 for more.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
AGRAPH	<b>g P.FN P.FN2</b> <b>g AGRAPH s</b>	(0) Alpha graphics. Displays a graphics image. Each character in the source <b>s</b> specifies an 8-dot-1-column pattern. The <b>X</b> - and <b>Y</b> -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 valid. 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	<b>g DISP ALL n</b>	(0) Sets the numeric display format to show all decimals of <i>real</i> or <i>complex</i> numbers whenever displayable (trailing decimal zeros will not be shown). ALL 0 works like ALL in <i>HP-42S</i> almost. For $x \geq 10^{16}$ (or earlier for <i>complex</i> numbers), display will switch to SCI or ENG with the maximum number of necessary decimals displayable using the large font (see ALLSCI). The same will happen if $x < 10^{-n}$ and more than 16 digits are required to show $x$ entirely (see examples in Section 2 of the OM). The limits differ in RBR – see p. 66.
a <sub>Moon</sub>	<b>f CONST a<sub>Moon</sub></b>	(-1) {} → {2} Semi-major axis of the Moon's orbit around the earth in <i>meter</i> .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
AND	<b>f [BITS] AND</b>	(2) {10} Works bitwise as in <i>HP-16C</i> (see the <i>OM, Section 2</i> ).
		(2) {1, 2} → {1} Works like AND in <i>HP-28S</i> , i.e. <i>x</i> and <i>y</i> are interpreted before executing this operation. Zero is ‘false’; any other <i>real</i> number is ‘true’.
ANGLES	<b>f [CAT.] VARS ANGLES</b>	<b>Submenu</b> of tagged angular variables defined at execution time. See pp. 124f.
arccos	<b>[TRI] arccos</b>	(1) {3, 8*, 9*}; {1, 2} → {4}; Returns the tagged angle $\text{arccos}(x)$ . <sup>7</sup>
arcosh	<b>[g EXP] [g] arcosh</b>	(1) {2, 3, 8*, 9*}
	<b>[TRI] [g] arcosh</b>	Returns $\text{arcosh}(x)$ .
arcsin	<b>[TRI] arcsin</b>	(1) {3, 8*, 9*}; {1, 2} → {4}; Returns the tagged angle $\text{arcsin}(x)$ . <sup>8</sup>
arctan	<b>[TRI] arctan</b>	(1) {3, 8*, 9*}; {1, 2} → {4}; Returns the tagged angle $\text{arctan}(x)$ . <sup>9</sup>

<sup>7</sup> Precisely, ARCCOS returns the principal value of  $\text{arccos}(x)$ , i.e. a *real* part  $\in [0, \pi]$  in  $\cancel{\pi^r}$ , or  $\in [0^\circ, 180^\circ]$  in  $\cancel{\pi^\circ}$  or  $\cancel{\pi''}$ , or  $\in [0^g, 200^g]$  in  $\cancel{\pi^g}$ , or  $\in [0, 1]$  in  $\cancel{\pi\pi}$ . Cf. ISO/IEC 9899.

<sup>8</sup> Precisely, ARCSIN returns the principal value of  $\text{arcsin}(x)$ , i.e. a *real* part  $\in [-\pi/2, \pi/2]$  in  $\cancel{\pi^r}$ , or  $\in [-90^\circ, 90^\circ]$  in  $\cancel{\pi^\circ}$  or  $\cancel{\pi''}$ , or  $\in [-100^g, 100^g]$  in  $\cancel{\pi^g}$ , or  $\in [-0.5, 0.5]$  in  $\cancel{\pi\pi}$ . Cf. ISO/IEC 9899.

<sup>9</sup> Precisely, ARCTAN returns the principal value of  $\text{arctan}(x)$ , i.e. a *real* part  $\in [-\pi/2, \pi/2]$  in  $\cancel{\pi^r}$ , for example (cf. ASIN), if SPCRES is set. Else the result interval for ATAN becomes  $(-\pi/2, \pi/2)$  in  $\cancel{\pi^r}$ , for example. Cf. ISO/IEC 9899.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
arsinh	<b>g EXP g arsinh</b>	(1) {2, 3, 8*, 9*}
	<b>TRI g arsinh</b>	Returns $\text{arsinh}(x)$ .
artanh	<b>g EXP g artanh</b>	(1) {2, 3, 8*, 9*}
	<b>TRI g artanh</b>	Returns $\text{artanh}(x)$ .
ASIN	<b>f CATALOG FCNS ASIN</b>	Points to ARCSIN above. Maintained for backward compatibility only.
ASR	<b>f BITS f ASR n</b>	(1) {10} 
		Works like <b>n</b> ( $\leq 63$ ) consecutive ASR commands in HP-16C, corresponding to a division of $x$ by $2^n$ . ASR <b>0</b> executes as NOP, but loads <b>L</b> . See Section 2 of the OM.
ASSIGN	<b>f ASN item, location</b>	(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.
ATAN	<b>f CATALOG FCNS ATAN</b>	Points to ARCTAN above. Maintained for backward compatibility only.
atm→Pa	<b>f U→ F&amp;p: atm→Pa</b>	(1) {2}; {1} → {2}
au→m	<b>f U→ x: au→m</b>	Convert pressures and distances. See pp. 148ff.
A:	<b>f U→ f A:</b>	Submenu. See p. 148.
a <sub>⊕</sub>	<b>f CONST a<sub>⊕</sub></b>	(-1) {} → {2} Semi-major axis of the Earth's orbit around the sun in <i>meter</i> .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
BACK	<b>g [P.FN] P.FN2</b> <b>g BACK n</b>	(0) Jumps $n$ 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.  <b>ATTENTION:</b> 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	<b>f [U→</b> <b>F&amp;p: bar→Pa</b>	(1) {2}; {1} → {2}  Converts pressures. See pp. 148ff.
BATT?	<b>g [INFO] f BATT?</b>	(-1) {} → {2}  Measures the battery voltage in the range between 1.9V and 3.4 V and returns this value. Measurement resolution is 1mV
bbl→m <sup>3</sup>	<b>f [U→</b> <b>f V:</b> <b>f barrel → m<sup>3</sup></b>	(1) {2}; {1} → {2}  Converts volumes. See pp. 148ff.
BC?	<b>f [BITS] g BC? n</b>	(-1) {10}  Tests if the specified bit in $x$ is clear.
BEEP	<b>f [I/O] BEEP</b>	(0) Sounds a sequence of four tones. See also TONE.
BeginP	<b>g [FIN] TVM</b> <b>f Begin</b>	(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. 13ff for general information)
<b>BestF</b>		<p>(0) Instructs your <i>WP 43S</i> to select the ‘best’ curve fit model for the current statistical data by picking the one with maximum <i>correlation</i> out of the models allowed (almost like <b>BEST</b> in <i>HP-42S</i>).</p> <p>Relevant for L.R., CORR, COV, <math>s_{XY}</math>, <math>\hat{x}</math>, and <math>\hat{y}</math>. You can accelerate computation of these functions significantly by excluding fit models making no sense for your data (e.g. for physical or technical reasons). The parameter <b>n</b> carries this information. Each fit model corresponds to a number as listed:</p> <ul style="list-style-type: none"> <li>• LINF 1</li> <li>• EXPF 2</li> <li>• LOGF 4</li> <li>• POWERF 8</li> <li>• ROOTF 16</li> <li>• HYPF 32</li> <li>• PARABF 64</li> <li>• CAUCHF 128</li> <li>• GAUSSF 256</li> </ul> <p>Take the numbers of all models you can exclude and sum them up – the result is <b>n</b>.</p> <p><b>Example:</b> Excluding the three 3-parameter models leads to <b>n = 64 + 128 + 256 = 448</b>. So call <b>BESTF 448</b> to look for the best-fitting 2-parameter model.</p> <p>Note ORTHOF is not part of the set of models under investigation. See pp. 224ff for more.</p>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
<b>BestF?</b>	<b>g INFO</b> <b>▼ BestF?</b>	(-1) {} → {1} Returns the 'best' curve fit model for the current statistical data by picking the one with maximum <i>correlation</i> out of the models allowed, encoded as an integer according to the list at BESTF.
<b>Binom<sub>p</sub></b>	<b>g PROB</b> <b>g Binom:</b> <b>Binom</b> etc.	(1) {2} <i>Binomial distribution</i> with the number of successes <b>g</b> in <b>X</b> , the probability of a success <b>p<sub>o</sub></b> in <b>I</b> , and the sample size <b>n</b> in <b>J</b> . See p. 224 for more.
<b>Binom<sub>▲</sub></b>		
<b>Binom<sub>△</sub></b>		
<b>Binom<sup>-1</sup></b>		Binom <sup>-1</sup> returns the maximum number of successes <b>m</b> for a given probability <b>p</b> in <b>X</b> , <b>p<sub>o</sub></b> in <b>I</b> and <b>n</b> in <b>J</b> .
<b>Binom:</b>	<b>g PROB</b> <b>g Binom:</b>	<i>Submenu</i> . See p. 133.
<b>BITS</b>	<b>f BITS</b>	<i>Menu</i> . See p. 127.
<b>B<sub>n</sub></b>	<b>g X.FN</b> <b>B<sub>n</sub></b>	(1) {1, 2}
<b>B<sub>n</sub>*</b>	<b>g X.FN</b> <b>B<sub>n</sub>*</b>	B <sub>n</sub> and B <sub>n</sub> * return the Bernoulli number for an integer <b>n &gt; 0</b> given in <b>X</b> , working with different definitions (see both formulas on p. 222).
<b>BS?</b>	<b>f BITS</b> <b>g BS? n</b>	(0) {10} Tests if the specified bit in <b>x</b> is set.
<b>Btu→J</b>	<b>f U→ E: Btu→J</b>	(1) {2}; {1} → {2} Converts energies. See pp. 148ff.
<b>c</b>	<b>f CONST</b> <b>c</b> etc.	(-1) {} → {2}
<b>c<sub>1</sub></b>		Speed of light in vacuum in <i>meter/second</i> ;
<b>c<sub>2</sub></b>		first and second radiation constants in <i>Planck's Law</i> (see p. 140).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
<b>cal→J</b>	<b>f [U→] E: cal→J</b>	(1) {2}; {1} → {2} Converts energies. See pp. 148ff.
<b>CASE</b>	<b>g P.FN P.FN2 g CASE s</b>	<p>(0) Works like SKIP below but takes the number of steps to skip from <i>s</i>.</p> <p><b>Example:</b> Assume a program section:</p> <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 <math>r12 \leq 1</math> then the program will proceed to step 101 and continue with a jump to step 105, for <math>r12 = 2</math> the program will go to step 102, etc., resulting in a nice controlled dispatcher for <math>1 \leq r12 \leq 4</math>.</p> <p><b>ATTENTION:</b> CASE might surprise you for <math>r12 &gt; 4</math> 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 <b>manually maintain all those statements individually</b>.</p>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
CATALOG	f CATALOG	Catalog of everything. See pp. 124ff.
Cauch <sub>p</sub>	g PROB f Cauch: Cauch etc.	(1) {2}  Cauchy-Lorentz (a.k.a. Lorentz or Breit-Wigner) distribution with the location $x_0$ specified in I and the shape $\gamma$ in J. See p. 227 for more.
Cauch <sub>Δ</sub>		
Cauch <sub>Δ</sub>		
Cauch <sup>-1</sup>		Cauch <sup>-1</sup> returns $x$ for a given probability $p$ in X, with $x_0$ in I and $\gamma$ in J.
Cauch:	g PROB f Cauch:	Submenu. See p. 133.
CauchF	f STAT ▼ f CauchF	(0) Selects the Cauchy (a.k.a. Lorentz) peak fit model. Relevant for CORR, COV, L.R., S <sub>XY</sub> , $\hat{x}$ , and $\hat{y}$ . See pp. 233ff for more.
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} → {1}  Returns the smallest integer $\geq x$ . Cf. FLOOR.
CF	g FLAGS CF n g MODE CF n g CLR CF n	(0) Clears the flag specified, i.e. sets it to 0.
CHARS	f CATALOG CHARS	
CLALL	g CLR g CLall	
		(0) Clears all registers, user flags, variables, and programs in RAM. Modes will stay as they are. Cf. CLCVAR, CLFALL, CLPALL, CLREGS, CLSTK, and RESET.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
<b>CLCVAR</b>	<b>g CLR CLCVAR</b>	(0) Clears all variables used in the <i>current program</i> , i.e. sets all such <i>real</i> and <i>complex</i> variables to <b>0</b> , all <i>integer</i> ones to <b>0</b> , all <i>time</i> variables to <b>0:00:00</b> , all <i>date</i> variables to <b>January 1<sup>st</sup></b> of year <b>0</b> , all character strings to zero length, and all the elements of all matrix variables used to <b>0</b> .
<b>CLFALL</b>	<b>g FLAGS</b> <b>g CLFall</b>	(0) Clears all global and local <i>user flags</i> . Compare CF.
	<b>g CLR f CLFall</b>	
<b>CLK</b>	<b>g CLK</b>	<b>Menu</b> . See p. 123.
<b>CLLCD</b>	<b>g CLR f CLLCD</b>	(0) Clears the <i>LCD</i> in the rectangular window north and west of the point <i>x</i> , <i>y</i> . I.e. all pixels $\geq x$ and $\geq y$ are cleared.
<b>CLMENU</b>	<b>g CLR CLMENU</b>	(0) Clears all <i>menu</i> key definitions for the programmable <i>menu</i> . See MENU.
	<b>g P.FN P.FN2 ...</b>	
<b>CLP</b>	<b>g CLR CLP</b>	(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.
<b>CLPALL</b>	<b>g CLR f CLPall</b>	(0) Clears all programs in <i>RAM</i> . Cf. CLP.
<b>CLR</b>	<b>g CLR</b>	<b>Menu</b> . See p. 130.
<b>CLREGS</b>	<b>g CLR f CLREGS</b>	(0) Clears all global and local general purpose <i>registers</i> allocated (see also LOC), i.e. sets all these registers to <b>0</b> . The contents of the <i>stack</i> and <b>L</b> are kept.
<b>CLSTK</b>	<b>g CLR f CLSTK</b>	Clears all <i>stack registers</i> currently allocated (i.e. either <b>X ... T</b> or <b>X ... D</b> ). All other <i>register</i> contents are kept. Cf. CLREGS.
	<b>0 f FILL</b>	

Item	Keystrokes	Remarks (see pp. 13ff for general information)
CLX	<b>g CLR CLX</b>  <b>⬅</b> (for closed $x$ )	(1) Clears stack register <b>X</b> , disabling <i>automatic stack lift</i> . Cf. CLREGS and CLSTK.
CLΣ	<b>g CLR CLΣ</b>  <b>f STAT g CLΣ</b>	(0) Clears the statistical summation registers and releases the memory allocated for them (see p. 170).
CNST	<b>g P.FN</b>  <b>f CNST n</b>	(-1) {} → {2}  Returns the constant stored at position <b>n</b> in <b>CONST</b> (see below and pp. 138ff). Allows for indirectly addressing these constants, also beyond $\infty$ .
COMB	<b>g PROB C<sub>yx</sub></b>	(2) {1}  Returns the number of possible <u>subsets</u> of $x$ items taken out of a set of $y$ items (i.e. choose $x$ out of $y$ ). No item occurs more than once in a subset, and <u>different orders</u> of the same $x$ items are <u>not counted</u> separately. Cf. PERM.  (2) {2, 3}  See pp. 222f for the formula.
CONJ	<b>g CPX conj</b>	(1) [3, 9]  Returns the <i>complex conjugate</i> of $x$ .
CONST	<b>f CONST</b>	Menu. See CNST above and pp. 138ff.
CONVG?	<b>g TEST</b>  <b>g CONVG? r</b>	(0) {2}  Checks for convergence by comparing $x$ and $y$ as determined by the lowest five bits of $r$ .  a) The very lowest 2 bits set the tolerance limit: $0 = 10^{-14}, \quad 1 = 10^{-24}, \quad 2 = 10^{-32}.$

Item	Keystrokes	Remarks (see pp. 13ff for general information)
		<p>b) The next two bits determine the comparison mode using the tolerance limit set:</p> <p>0 = compare the numbers <math>x</math> and <math>y</math> relatively,      1 = compare them absolutely.</p> <p>c) The top bit tells how special numbers are treated:</p> <p>0 = NaN and <math>\pm\infty</math> are considered converged,      1 = they are not considered converged.</p> <p>Now, <math>r = a + 4b + 16c</math>.</p>
<b>CORR</b>	<b>f [STAT] ▲ r</b>	$(-1) \{ \} \rightarrow \{2\}$ <p>Returns the coefficient of correlation for the current statistical data and curve fit model. See pp. 233ff for more.</p>
<b>cos</b>	<b>[TRI] cos</b>	$(1) \{2, 3, 8^*, 9^*\}; \{1, 4\} \rightarrow \{2\}$ <p>Returns the cosine of the angle in X (see Section 2 of the OM for details).</p>
<b>cosh</b>	<b>g EXP g cosh</b>	$(1) \{2, 3, 8^*, 9^*\}$ <p>Returns the hyperbolic cosine of <math>x</math>.</p>
	<b>[TRI] g cosh</b>	
<b>COV</b>	<b>f [STAT] ▲ cov</b>	$(-1) \{ \} \rightarrow \{2\}$ <p>Returns the population covariance for the two data sets entered via <math>\Sigma+</math>, depending on the curve fit model selected. See <math>s_{XY}</math> for the sample covariance and pp. 224ff for more.</p>
<b>CPX</b>	<b>g CPX</b>	<i>Menu.</i> See p. 130.
<b>CPXS</b>	<b>f [CATALOG] VARS CPXS</b>	<i>Submenu</i> of complex variables defined at execution time. See pp. 124f.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
CPX?	<b>g TEST ▲ CPX?</b>	(0) Checks if $x$ is <i>complex</i> . Returns <b>true</b> if $X$ contains data of type 3 or 9 with nonzero <i>imaginary</i> part.
CROSS	<b>f MATX f cross</b>	(2) {8} Requires two <i>real</i> 2D or 3D vectors in $\mathbf{X}$ and $\mathbf{Y}$ and returns their cross product. Crossing of 2D vectors works as for <i>complex</i> numbers.
	<b>g CPX cross</b>	(2) {3} → {2} When two <i>complex</i> numbers are crossed, your WP 43S simply returns a <i>real</i> number that is equal to the signed <i>magnitude</i> of the resulting moment vector. See an example in the OM.
ct→kg	<b>f U→ m: f carat → kg</b>	(1) {2}; {1} → {2}
cwt→kg	<b>f U→ m: cwt→kg</b>	Convert masses. See pp. 148ff.
CX→RE	<b>CC</b> (works in run mode only)	(-1) {3} → {2}; {9} → {8}
	<b>g CPX f CX→RE</b>	Cuts a closed <i>complex</i> object $x$ , putting either <ul style="list-style-type: none"> <li>(for <b>L</b>) its <i>real</i> part in <math>\mathbf{Y}</math> and its <i>imaginary</i> part in <math>\mathbf{X}</math> or</li> <li>(for <b>G</b>) <i>magnitude</i> in <math>\mathbf{Y}</math> and <i>phase</i> in <math>\mathbf{X}</math>.</li> </ul>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
DATE	<b>g CLK DATE</b>	(-1) { } → {6} Recalls the date 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	<b>f CATALOG VARS f DATES</b>	Submenu of date variables defined at execution time. See pp. 124f.
DATE→	<b>g CLK DATE→</b>	(-2) {2, 6} → {1} Assumes $x$ containing a date in the format selected (or a real number in corresponding format) and pushes its three components as integers on the stack. Reversible by →DATE.
DAY	<b>g CLK f DAY</b>	(1) {2, 6} → {1} Assumes $x$ containing a date in the format selected (or a real number in corresponding format) and extracts the day.
DBLR	<b>g INTS g DBLR</b> etc.	(10) Double word length commands for remainder, multiplication and division (cf. the HP-16C OH, Section 4, pp. 52ff).
DBLx		DBLR and DBL / accept a double size dividend in <b>Y</b> and <b>Z</b> (most significant bits in <b>Y</b> ), the divisor in <b>X</b> as usual, and return the result in <b>X</b> .
DBL/		DBLx takes $x$ and $y$ as factors as usual but returns the product in <b>X</b> and <b>Y</b> (most significant bits in <b>X</b> ).
dB→fr	<b>f U→ ▲ dB → field ratio</b>	(1) {2}; {1} → {2}
dB→pr	<b>f U→ ▲ dB → power ratio</b>	Convert ratios. See pp. 148ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
DEC	f LOOP f DEC r	(0) {1, 2, 10} Decrements $r$ by 1. Does not load L even for target address X.
DECOMP	f PARTS DECOMP	(-1) {1, 2} → {1} Decomposes $x$ (after converting it to an <i>improper fraction</i> , if applicable), returning a stack [denominator( $x$ ), numerator( $x$ ), ...]. <b>Example:</b> If X contains 2.25 then DECOMP will return $x = 4$ and $y = 9$ .
DEG	g MODE DEG	(0) Sets the ADM to <i>decimal degrees</i> . Indicated in the <i>status bar</i> .
DEG→	g L→ f DEG→	(1) {1, 2} → {4} Converts angles as described on p. 154.
DENMAX	g MODE f 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. Indicated in the <i>status bar</i> . For $x = 1$ , the current DENMAX setting is recalled, replacing $x$ .
DET	f CATALOG FCNS DET	Points to  M  explained on p. 102. Maintained for backward compatibility only.
DIGITS	f CAT. f DIGITS	Submenu of digits defined. See pp. 124ff.
DISP	g DISP	Menu. See p. 130.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
DOT	<b>g CPX</b> dot	(2) {3} → {2} Returns $Re(x) \cdot Re(y) + Im(x) \cdot Im(y)$
	<b>f MATX</b> 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 the OM, Sect. 2.
DROP	<b>g DROP</b> ↓	Drops $x$ ... from the stack. See Section 1 of the OM for details.
DROPy	<b>g STK</b> DROPy	Drops $y$
DSE	<b>f LOOP</b> DSE r	(0) {1, 2, 10} Given $cccccc.ffffii$ in the source, DSE decrements $r$ by $ii$ , skipping next program step if then $cccccc \leq ffff$ (cf. the backside of your WP 43S). 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	<b>f LOOP</b> DSL r	(0) {1, 2, 10} Works like DSE but skips if $cccccc < ffff$ .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
DSTACK	 	<p>(0) Sets the maximum number of <i>stack registers</i> displayed. For an input of 1, only <i>x</i> will be shown directly above the <i>menu section</i>; for 2, <i>x</i> and <i>y</i> will be displayed; maximum input is 4.</p> <p>Expanded views of e.g. matrices and multi-level returns like SUM will work as described in the OM regardless of the DSTACK parameter set. In any case, command input will be echoed directly below the <i>status bar</i>.</p> <p>This command is for old-school calculator users who feel distracted by a multitude of <i>stack registers</i> displayed changing simultaneously while only the lowest ones are really relevant.</p>
DSZ	 	<p>(0) {1, 10}</p> <p>Decrement <i>r</i> by 1 and skips the next step if <i>r</i> = 0 thereafter. Does not load <b>L</b> even for target address <b>X</b>. Cf. HP-16C.</p>
D.MS	 (for closed input)	<p>(0) Sets the ADM to <i>sexagesimal degrees</i>. Indicated in the <i>status bar</i>.</p>
D.MS→	 	<p>(1) {1, 2} → {4}</p>
D.MS→D	 	Convert angles as described on pp. 154f.
D.MY	 	(0) Sets the format dd.mm.yyyy for <i>dates</i> .
D→D.MS	 	<p>(1) {1, 2} → {4}</p> <p>Converts angles as described on p. 154.</p>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
D→J	<b>g CLK f D→J</b>	(1) {2, 6} → {1} Assumes $x$ containing a <i>date</i> in the format selected (or a <i>real number</i> in corresponding format) and converts it to a <i>Julian day number</i> <sup>10</sup> according to the J/G setting.
D→R	<b>g L→ g D→R</b>	(1) {1, 2} → {4} Converts angles as described on p. 154.
e	<b>f CONST e</b>	(-1) {} → {2}
e <sub>E</sub>	<b>f CONST e<sub>E</sub></b>	Elementary charge in <i>coulomb</i> and <i>Euler's e</i> .
EIGVAL	<b>f MATX ▲ g EIGVAL</b>	(-1) {8, 9} Evaluates the matrix $x$ and pushes a diagonal matrix containing its eigenvalues on the stack.
EIGVEC	<b>f MATX ▲ g EIGVEC</b>	(-1) {8, 9} Evaluates the matrix $x$ and pushes a matrix containing its eigenvectors on the stack.
END	<b>g P.FN END</b>	(0) Last command in a program and terminal for searching local labels as described in the OM, Section 3. Works like RTN in all other aspects.
ENDP	<b>g FIN TVM f End</b>	(0) Sets “End” mode in TVM: payments occur at the end of each period. Typical for loans and investments. Cf. BEGINP.
ENG	<b>g DISP ENG n</b>	(0) Sets engineer’s display format (see Section 2 of the OM).

<sup>10</sup> 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. 13ff for general information)
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. Compare  x  on p. 102.
ENTER↑	[ENTER↑]	(-1) Separates two entries in input. Copies x into Y, disabling <i>automatic stack lift</i> . See p. 120 and the OM, Section 1, for details.
ENTRY?	g [TEST] g ENTRY?	(0) Checks the (internal) entry flag. It is set if: <ul style="list-style-type: none"> <li>any character is entered in A/M, or</li> <li>any command is accepted for entry (be it via [ENTER↑], a function key, or [R/S] with a partial command line).</li> </ul> Useful in routines, e.g. after PAUSE.
EQN	g [EQN]	Menu. See p. 131.
EQ.DEL	g [EQN] f DELETE	Deletes an equation.
EQ.EDI	g [EQN] EDIT	Opens the Equation Editor to edit an existing equation.
EQ.NEW	g [EQN] NEW	Opens the Equation Editor to enter a new equation.
erf	g [X.FN] erf etc.	(1) {2}; {1} → {2} Returns the error function or its complement. See pp. 256ff for more.
erfc		

See Section 4 of  
the OM.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
ERR	<b>g P.FN</b> 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. 181ff for the respective error codes. Cf. MSG.
EVEN?	<b>g TEST</b> f EVEN?	(0) Checks if $x$ is integer and even.
$e^x$	<b>e<sup>x</sup></b>	(1) {2, 3, 8*, 9}; {1, 10} → {2} Returns $e^x$ .
EXITALL	<b>g P.FN</b> P.FN2 EXITall	(0) Exits all menus.
EXP	<b>g EXP</b>	Menu. See p. 131.
ExpF	<b>f STAT</b> ▼ ExpF	(0) Selects the exponential curve fit model. Relevant for CORR, COV, L.R., $s_{XY}$ , $\hat{x}$ , and $\hat{y}$ . See pp. 224ff for more.
Expon <sub>p</sub>	<b>g PROB</b>	(1) {2}
Expon <sub>λ</sub>	<b>f Expon:</b> Expon etc.	<i>Exponential distribution</i> with the rate $\lambda$ in <b>J</b> . See pp. 224ff for more.
Expon <sub>Δ</sub>		
Expon <sup>-1</sup>		Expon <sup>-1</sup> returns the survival time $t_s$ for a given probability $p$ in <b>X</b> , with $\lambda$ in <b>J</b> .
Expon:	<b>g PROB</b> f Expon:	Submenu. See p. 133.
EXPT	<b>f PARTS</b> EXPT	(1) {1, 2} → {1} Returns the exponent $h$ of the number $x = m \cdot 10^h$ displayed. Cf. MANT.
$e^{x-1}$	<b>g EXP</b> f $e^{x-1}$	(1) {2, 8*} For $x \approx 0$ , this returns a more accurate result for the fractional part than $e^x$ does.
E:	<b>f U→ E:</b>	Submenu. See p. 146.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
F	f CONST F	(-1) {} → {2} Faraday constant in <i>coulomb per mol.</i>
FB	f BITS g FB n	(1) {10} Inverts ('flips') the specified bit in <i>x</i> .
FBR	g a.FN g FBR	(0) Font browser. Shows all characters implemented in the fonts designed for your WP 43S.
FCNS	f CATALOG FCNS	Submenu of all provided functions. See pp. 124ff.
FC?	g FLAGS FC? n	(0) Tests if the specified <i>flag</i> is clear.
FC?C	g FLAGS f FC?C n	(0) Tests if the specified <i>flag</i> is clear. Clears, flips, or sets this <i>flag</i> after testing, respectively. etc.
FC?F	etc.	
FC?S		
F <sub>p</sub> (x)	g PROB F: F <sub>e</sub> (x)	(1) {2} <i>Fisher's F distribution.</i> F <sub>e</sub> (x) equals Q(F) on HP-21S. The degrees of freedom are specified in <b>I</b> and <b>J</b> . See pp. 224ff for more. etc.
F <sub>△</sub> (x)		
F <sub>▲</sub> (x)		
F <sup>-1</sup> (p)		
FF	g FLAGS FF n	(0) Flips the <i>flag</i> specified.
FIB	g X.FN f FIB	(1) {1} Returns the Fibonacci number (see pp. 222f). (1) {2, 3} Returns the extended Fibonacci number.
FILL	f FILL	Copies <i>x</i> to all stack registers.
FIN	g FIN	Menu. See p. 131.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
FIX	<b>g DISP FIX <i>n</i></b>	(0) Sets fixed point display format (see the OM, Sect. 2).
FLAGS	<b>g FLAGS</b>	Menu. See p. 131.
FLASH	<b>f CATALOG PROGS FLASH</b>	Submenu of global labels defined at execution time. See pp. 124f.
FLASH?	<b>g INFO f FLASH?</b>	(-1) {} → {1} Returns the number of free bytes in FM.
FLOOR	<b>g INTS g FLOOR</b>	(1) {8*} {1, 2} → {1} Returns the greatest integer $\leq x$ . Cf. CEIL.
fm.→m	<b>f U- x: ▲ fathom → m</b>	(1) {2, 11; {1} → {2}} Converts distances. See pp. 148ff.
FP	<b>f PARTS FP</b>	(1) {1, 2, 8*, 10} Returns the fractional part of $x$ . Cf. IP.
FP?	<b>g TEST f FP?</b>	(0) Tests $x$ for having a fractional part $\neq 0$ .
fr→dB	<b>f U- ▲ field ratio → dB</b>	(1) {2}; {1} → {2} Converts ratios. See pp. 148ff.
FS?	<b>g FLAGS FS? <i>n</i></b>	(0) Tests if the specified flag is set.
FS?C	<b>g FLAGS</b>	(0) Tests if the specified flag is set. Clears, flips, or sets this flag after testing, respectively. etc.
FS?F	<b>f FS?C <i>n</i></b>	
FS?S		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$ft \rightarrow m$	<b>f</b> [U→] <b>x:</b> <b>f</b> <b>ft.</b> → <b>m</b>	
$ft_{us} \rightarrow m$	<b>f</b> [U→] <b>x:</b> <b>▲ survey foot<sub>us</sub> → m</b>	(1) {2}; {1} → {2}
$fz_{UK} \rightarrow m^3$	<b>f</b> [U→] <b>f</b> <b>V:</b> <b>f</b> <b>floz<sub>UK</sub> → m<sup>3</sup></b>	Convert distances and volumes, respectively. See pp. 148ff.
$fz_{us} \rightarrow m^3$	<b>f</b> [U→] <b>f</b> <b>V:</b> <b>f</b> <b>floz<sub>us</sub> → m<sup>3</sup></b>	
$F_\alpha$	<b>f</b> <b>CONST</b> <b>F<sub>α</sub></b>	(-1) {} → {2}
$F_\delta$	etc.	Feigenbaum's $\alpha$ and $\delta$ .
$F:$	<b>g</b> <b>[PROB]</b> <b>F:</b>	_submenu_. See p. 133.
$f'$	<b>g</b> <b>[EQN]</b> <b>f'</b>	Submenus for calculating the first or second derivative of a given equation. See the OM (Sect. 4) for more.
$f''$	<b>g</b> <b>[EQN]</b> <b>f''</b>	
$f'(x)$	<b>f</b> <b>[ADV]</b> <b>f'(x)</b> <b>lbl</b>	(1, 2) → {2}  $f(x)$ [ $f'(x)$ ] returns the 1 <sup>st</sup> [2 <sup>nd</sup> ] derivative of the function $f(x)$ at position $x$ . This $f(x)$ must be specified in a routine starting with LBL <i>lbl</i> . On return, <b>Y</b> , <b>Z</b> , and <b>T</b> will be cleared and the position $x$ will be in <b>L</b> . See Section 4 of the OM for more.
$f''(x)$	<b>f</b> <b>[ADV]</b> <b>f</b> <b>f''(x)</b> <b>lbl</b>	<b>ATTENTION:</b> $f(x)$ and $f'(x)$ fill all stack registers with $x$ before calling the routine specified.
$F&p:$	<b>f</b> [U→] <b>F&amp;p:</b>	_submenu_. See p. 148.
$G$	<b>f</b> <b>CONST</b> <b>G</b>	(-1) {} → {2}
		Newtonian constant of gravitation in $m^3/kg\ s^2$ ;
$G_0$	<b>f</b> <b>CONST</b> <b>G<sub>0</sub></b>	also called $\gamma$ by other authors. Conductance quantum in <i>siemens</i> .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
GAP		(0) Defines the interval for inserting digit group separators in <i>reals</i> . 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 <i>reals</i> . After GAP 2, 1, or GAP 0, no group separators will be displayed in any numbers at all. See Sect. 2 of the OM.
GaussF		(0) Selects the Gauss peak fit model. Relevant for CORR, COV, L.R., $s_{XY}$ , $\hat{x}$ , and $\hat{y}$ . See pp. 233ff for more.
$G_c$		(-1) {} → {2} <i>Catalan's (mathematical) constant.</i>
GCD		(2) {1; 10} Returns the Greatest Common Divisor of <i>x</i> and <i>y</i> . <sup>11</sup> This will always be positive.
$g_d$		(1) {2, 3} ; {1} → {2}
$g_d^{-1}$	etc.	Returns the Gudermannian function or its inverse. See p. 257 for details.
$g_e$		(-1) {} → {2} <i>Landé's electron g-factor.</i>

<sup>11</sup>  $\text{GCD}(x, y) = \left| \left( x / \text{LCM}(x, y) \right) \times y \right|$ . See also LCM.

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

Item	Keystrokes	Remarks (see pp. 13ff for general information)
<b>Geom<sub>p</sub></b>	<b>g PROB</b> <b>g Geom:</b> <b>Geom<sub>p</sub></b> etc.	(1) {2}  <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 <b>J</b> . See pp. 224ff for more.
<b>Geom<sub>A</sub></b>		
<b>Geom<sub>A</sub></b>		
<b>Geom<sup>-1</sup></b>		Geom <sup>-1</sup> returns the number of failures <b>f</b> before 1 <sup>st</sup> success for given probabilities <b>p</b> in <b>X</b> , <b>p<sub>0</sub></b> in <b>J</b> .
<b>Geom:</b>	<b>g PROB</b> <b>f Geom:</b>	<b>Submenu.</b> See p. 133.
<b>gl<sub>uk</sub>→m<sup>3</sup></b>	<b>f U→</b> <b>f</b> <b>V:</b> <b>gl<sub>uk</sub>→m<sup>3</sup></b> etc.	(1) {2}; {1} → {2}
<b>gl<sub>us</sub>→m<sup>3</sup></b>		Convert volumes. See pp. 148ff.
<b>GM<sub>⊕</sub></b>	<b>f CONST</b> <b>GM<sub>⊕</sub></b>	(-1) {} → {2}  Newtonian constant of gravitation times the Earth's mass with its atmosphere included according to WGS84. <sup>12</sup> Displayed in $m^3/s^2$
<b>GRAD</b>	<b>g MODE</b> <b>GRAD</b>	(0) Sets the ADM to <i>grad</i> (a.k.a. <i>gradian</i> or <i>gon</i> ). Indicated in the <i>status bar</i> .
<b>GRAD→</b>	<b>g L→</b> <b>f GRAD→</b>	(1) {1, 2} → {4}  Converts angles as described on pp. 154f.
<b>GTO</b>	<b>f GTO</b> <b>labl</b>	(0) In <i>PEM</i> , inserts an unconditional branch to <b>labl</b> . Else positions the program pointer to <b>labl</b> .

<sup>12</sup> See [http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350\\_2.html](http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
GTO.	<b>f GTO</b> <b>. n</b>	to step <b>n</b> (specify up to four digits until becoming unambiguous in used program memory).
	<b>f GTO</b> <b>. labl</b>	to the global label specified.
	<b>f GTO</b> <b>.▲</b>	(0) Puts the program pointer ... directly after <u>previous</u> END, going to the top of <i>current program</i> (see Sect. 3 of the OM).
	<b>f GTO</b> <b>.▼</b>	directly after <u>next</u> END, going to the top of next program.
	<b>f GTO</b> <b>..</b>	to the end of used program memory in RAM (i.e. to the final END statement).
<b>g⊕</b>	<b>f CONST</b> <b>g⊕</b>	(-1) { } → {2} Standard earth acceleration in $m/s^2$ .
<b>h</b>	<b>f CONST</b> <b>h</b>	(-1) { } → {2} <i>Planck constant in joule-second.</i>
<b>ha→m²</b>	<b>f U→</b> <b>f A: ha → m²</b>	(1) {2}; {1} → {2} Converts areas. See pp. 148ff.
<b>Hₙ</b>	<b>g X.FN</b> <b>Orthog Hₙ</b>	(2) {2}; {1} → {2}
<b>Hₙₚ</b>	... <b>Orthog f Hₙₚ</b>	<i>Hermite polynomials for probability (<math>H_n</math>) and physics (<math>H_{np}</math>). See p. 251 for details.</i>
<b>hp_E→W</b>	<b>f U→</b> <b>P: hp_E→W</b> etc.	(1) {2}; {1} → {2} Convert powers. See pp. 148ff.
<b>hp_M→W</b>		
<b>hp_UK→W</b>		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
Hyper <sub>p</sub>	<b>g PROB</b> <b>g Hyper:</b> Hyper etc.	(1) {2} <i>Hypergeometric distribution</i> with the number of successes <b>g</b> in <b>X</b> , the probability of a success <b>p<sub>o</sub></b> in <b>I</b> , the sample size <b>n</b> in <b>J</b> , and the batch size <b>n<sub>o</sub></b> in <b>K</b> . See pp. 224ff for the formula.
Hyper <sub>-1</sub>		Hyper <sup>-1</sup> returns the maximum number of successes <b>m</b> for a given probability <b>p</b> in <b>X</b> , <b>p<sub>o</sub></b> in <b>I</b> , <b>n</b> in <b>J</b> , and <b>n<sub>o</sub></b> in <b>K</b> .
Hyper:		<b>Submenu.</b> See p. 133.
HypF		(0) Selects the hyperbolic fit model. Relevant for CORR, COV, L.R., s <sub>XY</sub> , $\hat{x}$ , and $\hat{y}$ . See pp. 233ff for more.
$\hbar$	<b>f CONST</b> $\hbar$	(-1) { } → {2} $= \frac{\hbar}{2\pi}$ , reduced Planck constant in <i>joule-second</i> (a.k.a. <i>Dirac constant</i> ).
IDIV	<b>g INTS</b> <b>f IDIV</b>	(2) {1, 10}; {2} → {1} Integer division, working like <b>/</b> <b>IP</b> . See the OM, Sect. 2, for the <i>data type</i> of the quotient.
IDIVR	<b>f CATALOG</b> FCNS IDIVR	{1, 2, 10} Like IDIV but also returns the remainder in <b>Y</b> . See the OM, Section 2, for the resulting <i>data types</i> of quotient and remainder.
iHg→Pa	<b>f U→</b> F&p: <b>f in.Hg → Pa</b>	(1) {2}; {1} → {2} Converts pressures. See pp. 148ff.
Im	<b>g CPX</b> <b>f Im</b> <b>f PARTS</b> <b>g Im</b>	(1) {3} → {2}; {9} → {8}; Returns the imaginary part of <b>x</b> . Cf. RE.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
INC	f [LOOP] f INC r	(0) {1, 2, 10} Increments $r$ by 1. Does not load L even for target address X.
INDEX	f [MATX] f INDEX name	(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	g [INFO]	Menu. See p. 131.
INPUT	g [P.FN] INPUT r	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.
INTS	g [INTS]	Menu. See p. 131.
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] <sup>-1</sup> on p. 102. Maintained for backward compatibility only.
in.→m	f [U→] x: g in.→m	(1) {2}; {1} → {2} Converts distances. See pp. 148ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
IP	f PARTS IP	(1) {1, 8*, 10}; {2} → {1} Returns the integer part of $x$ . Cf. FP.
ISE	f LOOP ISE s	(0) {1, 2, 10} Given $cccccc.ffffii$ in the source $s$ , ISE increments $s$ by $ii$ , skipping next program step if $cccccc \geq ffff$ then (cf. the backside of your WP 43S). If $s$ has no fractional part then $ffff = 0$ and $ii = 0$ . If $ii = 0$ , $cccccc$ will be incremented by 1. ISE does not load <b>L</b> even for target address <b>X</b> . Note that neither $fff$ nor $ii$ can be negative, but $cccccc$ can.
ISG	f LOOP ISG s	(0) {1, 2, 10} Works like ISE but skips if $cccccc > fff$ .
ISZ	f LOOP ISZ s	(0) {1, 10} Increments $s$ by 1, skipping next program step if $s = 0$ thereafter. ISZ does not load <b>L</b> even for target address <b>X</b> . Cf. HP-16C.
I <sub>xyz</sub>	g X.FN f I <sub>xyz</sub>	(3) {1, 2} Returns the <i>regularized Beta function</i> . <span style="float: right;">See p. 257 for more.</span>
IΓ <sub>p</sub>	g X.FN f IΓ <sub>p</sub>	(2) {1, 2}
IΓ <sub>q</sub>	etc.	Returns the <i>regularized Gamma function</i> (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	f I/O	Menu. See p. 131.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$J_y(x)$	 $J_y(x)$	(2) {2}; {1} → {2} $J_y(x)$ returns the <i>Bessel function of first kind</i> and order $y$ . See p. 258 for details.
$J+$	 	(1) Increments or decrements the column index of the indexed matrix. If 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.
$J-$	 	
$J/G$	  $J/G$	(0) {2, 6} Sets the date the Gregorian calendar was introduced in the region you are interested in. See <i>Dates</i> in Section 2 of the OM.
$J \rightarrow Btu$	  etc.	(1) {2}; {1} → {2}
$J \rightarrow cal$		Convert energies. See pp. 148ff.
$J \rightarrow D$	 	(1) {1} → {6} Takes $x$ as a <i>Julian day number</i> <sup>13</sup> and converts it to a common <i>date</i> according to J/G (see above) and the date format selected.
$J \rightarrow Wh$	 	(1) {2}; {1} → {2} Converts energies. See pp. 148ff.
$k$	 	(-1) {} → {2} <i>Boltzmann constant</i> in <i>joule per kelvin</i> .
KEY		See KEYG and KEYX below.

<sup>13</sup> Translator's note: *Julian day number* translates to «jour Julien» in French and to “Julianisches Datum” in German. See the corresponding articles in Wikipedia for more information about these numbers.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
KEYG	<b>g [P.FN] P.FN2</b> <b>KEYG key#, labl</b>	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  <b>KEY key# GTO labl</b> or <b>KEY key# XEQ labl</b> ,
KEYX	<b>g [P.FN] P.FN2</b> <b>KEYX key#, labl</b>	respectively. Key numbers go from 1 to 18 with 1 corresponding to <b>F1</b> , 9 to <b>f F3</b> , and 14 to <b>g F2</b> , for example.
KEY?	<b>g [TEST] g KEY? r</b>	(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 <b>r</b> . Key codes reflect the rows and columns on the keyboard (see the OM, Sect. 3; cf. GETKEY on HP-42S).
kg→ct	<b>f [U→] m: f kg → carat</b>	(1) {2}; {1} → {2} Convert masses. See pp. 148ff.
kg→cwt	<b>f [U→] m: kg → cwt</b> etc.	
kg→lb.		
kg→oz		
kg→scw	<b>f [U→] m: f kg → sh.cwt</b>	
kg→sto	<b>f [U→] m: f kg → stone</b>	
kg→s.t	<b>f [U→] m: ▲ kg → short ton</b>	
kg→ton	<b>f [U→] m: ▲ kg→ton</b>	
kg→trz	<b>f [U→] m: f kg → tr.oz</b>	
K <sub>J</sub>	<b>f [CONST] K<sub>J</sub></b> <b>(-1) {} → {2}</b>	Josephson constant in <i>hertz per volt</i> .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
KTYP?	<b>g INFO</b> <b>KTYP? r</b>	<p>(-1) { } → {1}</p> <p>Assumes a key code in the address specified (see KEY?), checks it, and returns its key type:</p> <ul style="list-style-type: none"> <li>• 0 ... 9 if it corresponds to a digit 0 ... 9,</li> <li>• 10 if it corresponds to ., E, or +/-,</li> <li>• 11 if it corresponds to f or g,</li> <li>• 13 if it corresponds to a softkey, and</li> <li>• 12 if it corresponds to any other key.</li> </ul> <p>May help in user interaction with routines (see the OM, Section 3)..</p>
LASTx	<b>RCL L</b>	(-1) See Sect. 1 of the OM. Actually, this command will be recorded as RCL L in routines.
lbf→N	<b>f U→ F&amp;p: lbf→N</b>	(1) {2}; {1} → {2} Converts forces. See pp. 148ff.
LBL	<b>g LBL labl</b>	(0) Identifies programs and routines for execution and branching. Read more about labels and specifying them in Section 3 of the OM.
LBL?	<b>g TEST</b> <b>g LBL?</b> <b>labl</b>	(0) Tests for existence of the label specified, anywhere in program memory. See LBL for more.
lb.→kg	<b>f U→</b> <b>m: lb.→kg</b>	(1) {2}; {1} → {2} Converts masses. See pp. 148ff.
LCM	<b>g INTS</b> <b>f LCM</b>	(2) {1; 10} Returns the Least Common Multiple of x and y. <sup>14</sup> This will always be positive. Cf. GCD.

<sup>14</sup>  $\text{LCM}(x, y) = \left| \left( x / \text{GCD}(x, y) \right) \times y \right|.$

Item	Keystrokes	Remarks (see pp. 13ff for general information)
LEAP?	 	(0) {2, 6} 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 <sub>p</sub>	 	(1) {2} <i>Log-normal distribution</i> with $\mu = \ln \bar{x}_g$ specified in <b>I</b> and $\sigma = \ln \varepsilon$ in <b>J</b> . See $\bar{x}_g$ and $\varepsilon$ below and pp. 224ff for more.
LgNrm <sub>▲</sub>		
LgNrm <sub>▼</sub>		
LgNrm <sup>-1</sup>		LgNrm <sup>-1</sup> returns $x$ for a given probability $p$ in <b>X</b> , with $\mu$ in <b>I</b> and $\sigma$ in <b>J</b> .
LgNrm:	 	Submenu. See p. 133.
LinF	 	(0) Selects the linear fit model. Relevant for CORR, COV, L.R., $s_{XY}$ , $\hat{x}$ , and $\hat{y}$ . See pp. 224ff for more.
LJ	 	{10} Left justifies a bit pattern within its word size as in HP-16C: The stack will lift, placing the left-justified word in <b>Y</b> and the count of bit-shifts necessary to left justify the word in <b>X</b> .  <b>Example</b> for word size 8: 1 0110 <sub>2</sub> LJ returns $x = 3$ and $y = 1011\ 0000_2$
L <sub>m</sub>	 	(2) {2}; {1} → {2}
L <sub>max</sub>		<i>Laguerre polynomials</i> and <i>Laguerre's generalized polynomials</i> . See pp. 239f for more.
LN		(1) {2, 3, 8*, 9*}; {1, 10} → {2} Returns the natural logarithm of $x$ .

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. 13ff for general information)
$\text{LN}\beta$		(2) {2, 3}; {1} → {2} Returns the natural logarithm of <i>Euler's Beta function</i> (see p. 93).
$\text{LN}\Gamma$		(1) {2, 3}; {1} → {2}
		Returns the natural logarithm of $\Gamma(x)$ (see p. 93). Allows also for calculating really great factorials (see an example in the OM).
$\text{LN}(1+x)$		(1) {2, 8*} For $x \approx 0$ , this returns a more accurate result for the fractional part than $\ln(x)$ does.
LOAD		Restores the entire backup from <i>FM</i> and returns <b>Backup restored</b> . Thus, $\text{LOAD} = \text{LOADP} + \text{LOADR} + \text{LOADSS} + \text{LOAD}\Sigma$ . <sup>15</sup>
LOADP		(0) Loads the entire program memory from backup and appends it to the programs already in <i>RAM</i> (if there is sufficient space – else an error will be thrown). <sup>15</sup>
LOADR		(0) Recovers the numbered general purpose <i>registers</i> from backup. Lettered <i>registers</i> will not be recalled. <sup>15</sup>
LOADSS		(0) Recovers the system state from backup. <sup>15</sup>
LOADΣ		(0) Recovers the statistical summation <i>registers</i> from backup. Throws an error if there are none. <sup>15</sup>
LocR		(0) Allocates <i>n local registers</i> ( $\leq 100$ ) and 16 <i>local flags</i> for the <i>current routine</i> . See the OM, Sect. 3.

<sup>15</sup> See *SAVE* on p. 73 and *App. A* on pp. 143f for more.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
LocR?	g [INFO] f LocR?	(-1) {} → {1} Returns the number of <i>local registers</i> currently allocated.
LOG <sub>10</sub>	g lg	(1) {1, 2, 3, 8*, 9*, 10} ({1} → {2}) Returns the logarithm of $x$ for base 10.
LOG <sub>2</sub>	g EXP lb x	(1) {1, 2, 3, 8*, 9*, 10} ({1} → {2}) Returns the logarithm of $x$ for base 2.
LogF	f STAT ▾ LogF	(0) Selects the logarithmic curve fit model. Relevant for CORR, COV, L.R., s <sub>XY</sub> , $\hat{x}$ , and $\hat{y}$ . See pp. 224ff for more.
Logis <sub>p</sub>	g PROB f Logis: Logis etc.	(1) {2} <i>Logistic distribution</i> with $\mu$ given in <b>I</b> and $s$ in <b>J</b> . See pp. 224ff for details.
Logis <sub>Δ</sub>		
Logis <sub>▲</sub>		
Logis <sup>-1</sup>		
Logis:	g PROB f Logis:	Submenu. See p. 133.
LOG <sub>xy</sub>	g EXP log <sub>xy</sub>	(2) {1, 2, 3, 8*, 9*, 10}; ({1} → {2}) Returns the logarithm of $y$ for the base $x$ .
LOOP	f LOOP	Menu. See p. 132.
l <sub>PL</sub>	f CONST l <sub>PL</sub>	(-1) {} → {2} <i>Planck length</i> in <i>meter</i> .
ly→m	f U→ x: ly→m	(1) {2}; {1} → {2} Converts distances. See pp. 148ff.
L.INTS	f CATALOG VARS L.INTS	Submenu of <i>long integer variables</i> defined at execution time. See pp. 124ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
L.R.	<b>f STAT ▲ L.R.</b>	<p>(-2) or (-3) { } → {2}</p> <p>Pushes the parameters <math>a_2</math> (in Z), <math>a_1</math> (in Y), and <math>a_0</math> (in X) of the fit curve through the data points accumulated in the statistical summation registers on the stack, according to the curve fit model selected (see LINF, ORTHOF, EXPF, POWERF, LOGF, HYPF, ROOTF, PARABF, CAUCHF, GAUSSF).</p> <p>For a straight line, <math>a_0</math> is its y-intercept and <math>a_1</math> is its slope. See pp. 224ff for more.</p>
$m^2 \rightarrow ac$	<b>f [U→] f A: <math>m^2 \rightarrow acre</math></b>	<p>(1) {2}; {1} → {2}</p> <p>Convert areas. See pp. 146ff.</p>
$m^2 \rightarrow ac_{us}$	<b>f [U→] f A: <math>m^2 \rightarrow acre_{us}</math></b>	
$m^2 \rightarrow ha$	<b>f [U→] f A: <math>m^2 \rightarrow ha</math></b>	
$m^3 \rightarrow bbl$	<b>f [U→] f V: <math>m^3 \rightarrow barrel</math></b>	
$m^3 \rightarrow fz_{uk}$	<b>f [U→] f V: <math>m^3 \rightarrow fz_{uk}</math></b>	
$m^3 \rightarrow fz_{us}$	etc.	
$m^3 \rightarrow gl_{uk}$	<b>f [U→] f V: <math>m^3 \rightarrow gl_{uk}</math></b>	
$m^3 \rightarrow gl_{us}$	etc.	
MANT	<b>f PARTS MANT</b>	<p>(1) {2}; {1} → {2}</p> <p>Returns the mantissa <math>m</math> of the number <math>x = m \cdot 10^h</math> displayed. Cf. EXPT.</p>
MASKL	<b>f BITS MASKL <i>n</i></b>	<p>(-1) { } → {10}</p> <p>Work like MASKL and MASKR on HP-16C, but with the mask length (or its address) following the command instead of being taken from X. Thus, the mask is pushed on the stack. MASKL 0 and MASKR 0 return 0.</p>
MASKR	<b>f BITS MASKR <i>n</i></b>	

Item	Keystrokes	Remarks (see pp. 13ff for general information)
		<b>Example:</b> For WSIZE 8, MASKL 3 returns a mask word 1110 0000 <sub>2</sub> . Use it e.g. for extracting the top three bits of an arbitrary byte via AND.
MATRS	f [CATALOG] VARS MATRS	Submenu of matrix variables defined at execution time. See pp. 124f.
MATR?	g [TEST] ▲ MATR?	(0) Checks if <i>x</i> is a <i>real</i> or <i>complex</i> matrix.
MATX	f [MATX]	Menu. See p. 132.
Mat_X	f [MATX] SIM EQ Mat X	(-1) Returns the solution vector of a system of linear equations (see the OM, Sect. 2).
max	g [X.FN] g max	(2) {1, 2, 4, 5, 6, 7, 10}  Returns the maximum of <i>x</i> and <i>y</i> .
m <sub>e</sub>	f [CONST] m <sub>e</sub>	(-1) {} → {2}  Electron mass in <i>kilogram</i> .
MEM?	g [INFO] MEM?	(-1) {} → {1}  Returns the number of free <i>bytes</i> in program memory, also taking into account the local registers 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 [CAT.] MENUS	Submenu of all menus defined at execution time. See pp. 124ff.
min	g [X.FN] g min	(2) {1, 2, 4, 5, 6, 7, 10}  Returns the minimum of <i>x</i> and <i>y</i> .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
MIRROR	f BITS f MIRROR	(1) {10} Reflects the bit pattern in $x$ (e.g. 0001 0111 <sub>2</sub> would become 1110 1000 <sub>2</sub> for word size 8).
mi. $\rightarrow$ m	f U $\rightarrow$ x: f mi. $\rightarrow$ m	(1) {2}; {1} $\rightarrow$ {2} Converts distances. See pp. 146ff.
M <sub>Moon</sub>	f CONST M <sub>Moon</sub>	(-1) {} $\rightarrow$ {2}
m <sub>n</sub>	etc.	Mass of the Moon in <i>kilogram</i> ; neutron mass in <i>kilogram</i> ; neutron to proton mass ratio.
m <sub>n</sub> /m <sub>p</sub>		
MOD	g MOD	(2) {1, 2, 10}
	g INTS f MOD	Returns $y \bmod x$ (modulo, see Section 2 of the OM for examples). Cf. RMD.
MODE	g MODE	Menu. See p. 132.
MONTH	g CLK f MONTH	(1) {2, 6} $\rightarrow$ {1} Assumes $x$ containing a <i>date</i> in the format selected (or a <i>real</i> number in corresponding format) and extracts the month.
m <sub>p</sub>	f CONST m <sub>p</sub>	(-1) {} $\rightarrow$ {2}
m <sub>PL</sub>	etc.	Proton mass and Planck mass in <i>kilogram</i> ; proton to electron mass ratio.
m <sub>p</sub> /m <sub>e</sub>		
MSG	g P.FN P.FN2 f MSG	(0) {1, 2} Throws the ( <i>temporary</i> ) error message specified by the integer part of $x$ . Cf. ERR. See App. C on pp. 181ff for the respective error codes.
m <sub>u</sub>	f CONST m <sub>u</sub>	(-1) {} $\rightarrow$ {2}
m <sub>u</sub> c <sup>2</sup>	f CONST m <sub>u</sub> c <sup>2</sup>	Atomic mass constant in <i>kilogram</i> and its energy equivalent in <i>joule</i> .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
MUL $\pi$	<b>g MODE MUL<math>\pi</math></b>	(0) Sets the ADM to <i>multiples of <math>\pi</math></i> . Indicated in the <i>status bar</i> .
MUL $\pi\rightarrow$	<b>g L<math>\leftrightarrow</math> f MUL<math>\pi\rightarrow</math></b>	(1) {1, 2} → {4} Converts angles as described on pp. 154f.
MVAR	<b>g P.FN f MVAR name</b>	(0) Defines a <i>menu variable</i> . Such variables are required for VARMNU. Works in PEM only.
MyMenu	<b>f CAT MENUS MyMenu</b>	User menu. See the OM, Section 6.
My $\alpha$	<b>f CAT CHARS My<math>\alpha</math></b>	User menu in AIM.
$m_\mu$	<b>f CONST <math>m_\mu</math></b>	(-1) {} → {2} Muon mass in <i>kilogram</i> .
M.DELR	<b>f DELR with M.EDIT displayed</b>	(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	<b>f MATX f DIM name</b>	(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?	<b>g INFO g DIM?</b>	(8, 9) → {1}
	<b>f MATX ▲ f DIM?</b>	Returns the dimensions of the matrix $x$ (rows to <b>Y</b> , columns to <b>X</b> ). Note the matrix is saved in <b>L</b> . Former $y$ goes into <b>Z</b> , former $z$ into <b>T</b> , etc.
M.DY	<b>g CLK ▲ M.DY</b>	(0) Selects the format mm/dd/yyyy for dates.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
M.EDI	f MATX EDIT	(2) {8, 9} Opens $x$ using the <i>Matrix Editor</i> (like MATRIX EDIT in <i>HP-42S</i> ). <sup>16</sup> See Section 2 of the OM.
M.EDIN	f MATX f EDITN name	(2) Opens a named matrix using the <i>Matrix Editor</i> (like MATRIX EDITN in <i>HP-42S</i> ). <sup>16</sup> See Section 2 of the OM.
M.EDIT		Submenu for matrix editing, called by M.EDI or M.EDIN. See p. 132.
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	(0) Asks for target row and column and moves to this matrix element.
M.GROW	f GROW	(0) Allows the indexed matrix to grow automatically (see J+ above and Section 2 of the OM; see also GROW in the <i>HP-42S Owner's Manual</i> , p. 213.). Cf. M.WRAP.
M.INSR	f INSR	(0) Inserts a new row of elements containing zero, left of the current cursor position in the matrix.

<sup>16</sup> In the real *HP-42S*, EDIT and EDITN don't actually disable *automatic stack lift*; they preserve the *stack lift* state – you can observe this if you do ENTER vs. a *stack-lift-enabling* operation (e.g.  $x\gtrless y$ ) just before invoking them. This behavior is not really useful, but it needs to be emulated anyway, since not doing so risks breaking *HP-42S* programs.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
M.LU	 	(1) WP 34S: Takes a <i>descriptor</i> of a square matrix in <b>X</b> . Transforms ( <b>X</b> ) into its LU decomposition in-situ. The value in <b>X</b> 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		(2) {1, 2} → {8} Creates a new matrix (like NEW in HP-42S). Its number of rows shall be supplied in <b>Y</b> and its number of columns in <b>X</b> . M.NEW returns a matrix $x$ with all its elements set to zero.
M.OLD	with M.EDIT displayed	(0) Recalls the old element content (like OLD in HP-42S). See the OM, Sect. 2.
M.PUT	 	(0) {8, 9} Puts the matrix $x$ as is into the indexed matrix (like PUTM in HP-42S). Cf. M.GET.
M.R>R	 	(0) {8, 9} Swaps row $x$ and row $y$ of the indexed matrix (like R<>R in HP-42S).
M.SIMQ		Submenu of <u>MATX</u> , called by SIM_EQ.
M.SQR?	 	(0) Returns true if $x$ is a square matrix.
M.WRAP	with M.EDIT displayed	(0) Controls the index pointers (see Section 2 of the OM). Cf. M.GROW.
m:		Submenu. See p. 146.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
m→au	f U→ x: m→au	(1) {2}; {1} → {2} Convert distances or heights. See pp. 146ff.
m→fm.	f U→ x: ▲ m → fathom	
m→ft.	f U→ x: f m→ft.	
m→ft <sub>us</sub>	f U→ x: ▲ m → survey foot <sub>us</sub>	
m→in.	f U→ x: g m→in.	
m→ly	f U→ x: m→ly	
m→mi.	f U→ x: f m→mi.	
m→nmi.	f U→ x: f m→nmi.	
m→pc	f U→ x: m→pc	
m→pt.	f U→ x: g m → point	
m→yd.	f U→ x: g m→yd.	
m <sub>⊕</sub>	f CONST m <sub>⊕</sub>	(-1) {} → {2}
m <sub>⊕</sub>	etc.	Masses of the Sun and Earth in <i>kilogram</i> .
N <sub>A</sub>	f CONST N <sub>A</sub>	(-1) {} → {2} Avogadro's number in <i>particles per mol</i> .
NAND	f BITS f NAND	(2) Works in analogy to AND. See p. 20.
NaN	f CONST NaN	(-1) Not a Number.
NaN?	g TEST ▲ f NaN?	(0) Returns true if x is Not a Number.
NBin <sub>p</sub>	g PROB	(1) {2}
NBin <sub>▲</sub>	g NBin: NBin	etc.
NBin <sub>▲</sub>		
NBin <sup>-1</sup>		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
NBin:	<b>g PROB g NBin:</b>	Submenu. See p. 133.
NEIGHB	<b>g INFO f NEIGHB</b>	<p>(2) {1}</p> <p>Returns ...</p> <ul style="list-style-type: none"> <li>• <math>x + 1</math> for <math>x &lt; y</math> ;</li> <li>• <math>x</math> for <math>x = y</math> ;</li> <li>• <math>x - 1</math> for <math>x &gt; y</math> .</li> </ul> <p>(2) {2}</p> <p>Returns the nearest machine-representable number to <math>x</math> in the direction towards <math>y</math> in the mode set. For</p> <ul style="list-style-type: none"> <li>• ... <math>x &lt; y</math> , it is the machine successor of <math>x</math> ;</li> <li>• ... <math>x = y</math> , it is <math>y</math> ;</li> <li>• ... <math>x &gt; y</math> , it is the machine predecessor of <math>x</math>.</li> </ul> <p>NEIGHB may be useful investigating numeric stability (see NEIGHBOR in the <i>HP-71 Math Pac</i>).</p>
NEXTP	<b>g X.FN g NEXTP</b>	<p>(1) {1, 10}; {2} → {1}</p> <p>Returns the next prime number greater than <math> IP(x) </math>. See also PRIME? on p. 64.</p>
nmi.→m	<b>f U→ x: f nmi.→m</b>	<p>(1) {2}; {1} → {2}</p> <p>Converts distances. See pp. 146ff.</p>
NOP	<b>g P.FN P.FN2 f NOP</b>	'Empty' program step (for historical reasons only).
NOR	<b>f BITS f NOR</b>	(2) Works in analogy to AND. See p. 20.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
Norml <sub>p</sub>	<b>g PROB</b> Norml: Norml etc.	(1) {2}  Normal distribution with an arbitrary mean $\mu$ given in I and a standard deviation $\sigma$ in J. See Sect. 2 of the OM for an application example and pp. 224ff for more.
Norml <sub>A</sub>		
Norml <sup>-1</sup>		Norml <sup>-1</sup> returns $x$ for a given probability $p$ in X, with $\mu$ in I and $\sigma$ in J.
Norml:	<b>g PROB</b> Norml:	Submenu. See p. 133.
NOT	<b>f BITS NOT</b>	(1) {10}  Inverts $x$ bit-wise as on HP-16C.
		(1) {1, 2} → {1}  Returns 1 for $x = 0$ , and 0 for $x \neq 0$ .
nΣ	<b>g Σ n</b>	(-1) {} → {1}  Recalls the number of accumulated data points.
N→lbf	<b>f U→</b> F&p: N→lbf	(1) {2}; {1} → {2}  Converts forces. See pp. 146ff.
ODD?	<b>g TEST f ODD?</b>	(0) Checks if $x$ is integer and odd.
OFF	<b>g OFF</b>	(0) Turns your WP 43S off. In PEM, inserts a step to turn it off under program control.
OR	<b>f BITS OR</b>	(2) Works in analogy to AND. See p. 20.
OrthoF	<b>f STAT</b> <b>g OrthoF</b>	(0) Selects the linear orthogonal fit model. Relevant for CORR, COV, L.R., $s_{XY}$ , $\hat{x}$ , and $\hat{y}$ . See pp. 233ff for more.
ORTHOG	<b>g X.FN Orthog</b>	Submenu. See p. 136.
oz→kg	<b>f U→</b> m: oz→kg	(1) {2}; {1} → {2}  Converts masses. See pp. 146ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
P <sub>0</sub>	f CONST P <sub>0</sub>	(-1) {} → {2} Standard atmospheric pressure in pascal.
ParabF	f STAT ▼ f ParabF	(0) Selects the parabolic fit model. Relevant for COV, CORR, L.R., S <sub>XY</sub> , $\hat{x}$ , and $\hat{y}$ . See pp. 233ff for more.
PARTS	f PARTS	Menu. See p. 132.
PAUSE	g P.FN PAUSE n	(0) Within a routine running, refreshes the display and pauses program execution for $n$ ticks (s. TICKS), with $0 \leq n \leq 99$ . The pause will terminate early when you press a key.
Pa→atm	f U→ F&p: ▼ Pa→atm	(1) {2}; {1} → {2} Convert pressures. See pp. 148ff.
Pa→bar	f U→ F&p: Pa→bar	
Pa→iHg	f U→ F&p: f Pa → in.Hg	
Pa→psi	f U→ F&p: Pa→psi	
Pa→tor	f U→ F&p: f Pa → torr	
pc→m	f U→ x: pc→m	(1) {2}; {1} → {2} Converts distances. See pp. 148ff.
PERM	g PROB P <sub>yx</sub>	(2) {1} Returns the number of possible <u>arrangements</u> (a.k.a. <i>permutations</i> ) of $x$ <i>items</i> taken out of a set of $y$ <i>items</i> . No <i>item</i> occurs more than once in an arrangement, and <u>different orders</u> of the same $x$ <i>items</i> are counted separately. Cf. COMB.  (2) {2, 3} See pp. 222f for the formula.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
PGMINT	f ADV f PGMINT <i>labl</i>	Specifies the address of the expression to be integrated or solved, respectively. See Section 4 of the OM.
PGMSLV	f ADV f PGMSLV <i>labl</i>	
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. 19 for more.
PLOT	f STAT g PLOT	(0) Plots the <i>n</i> data points given by the <i>nx2</i> matrix <i>x</i> . See p. B-19 for more.
P <sub>n</sub>	g X.FN Orthog P <sub>n</sub>	(1) {2}; {1} → {2} <i>Legendre polynomials.</i> See pp. 239f for more.
POINT	g P.FN P.FN2 g POINT	(1) {1, 2} Turns on a square point (3x3 px □) on the screen. The location of its center is given by the integer parts of the numbers in X and Y. See AGRAPH on p. 19 for more.
Poiss <sub>p</sub>	g PROB g Poiss: Poiss	(1) {2}; {1} → {2}
Poiss <sub>λ</sub>	etc.	Poisson distribution with the number of successes <i>g</i> in X and the Poisson parameter <i>λ</i> in I. See pp. 224ff for details.
Poiss <sub>λ</sub> <sup>-1</sup>		Poiss <sup>-1</sup> returns the maximum number of successes <i>m</i> for a given probability <i>p</i> in X and <i>λ</i> in I.
Poiss:	g PROB g Poiss:	Submenu. See p. 133.
PopLR	g P.FN g PopLR	(0) Pops the local registers allocated to the <i>current routine</i> (see Section 3 of the OM) <u>without returning to the calling routine</u> . See LOCR and RTN.
PowerF	f STAT ▾ PowerF	(0) Selects the power curve fit model. Relevant for CORR, COV, L.R., s <sub>XY</sub> , ŷ, and ŷ̂ (see pp. 224ff for more).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
PRCL	<b>g P.FN f PRCL</b>	(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 <i>OM</i> ). 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?	<b>g TEST f PRIME?</b>	(0) {1, 2, 10}  Checks if $ IP(x) $ is a prime. Returns <b>true</b> for prime and <b>false</b> for composite. For $x > 3.3 \times 10^{24}$ , <b>true</b> means ‘probably prime’ with infinitesimal probability for being composite (see p. 175 for more).
PRINT	<b>g PRINT</b>	Menus. See p. 133.
PROB	<b>g PROB</b>	
PROG		<b>Submenu</b> of global labels defined when calling XEQ etc. See Section 3 of the <i>OM</i> .
PROGS	<b>f CAT. PROGS</b>	<b>Submenu</b> of global labels defined at execution time. See pp. 124f.
pr→dB	<b>f U→ ▲ power ratio → dB</b>	(1) {2}; {1} → {2}
psi→Pa	<b>f U→ F&amp;p: psi→Pa</b>	Convert ratios or pressures. See pp. 146ff.
PSTO	<b>g P.FN f PSTO</b>	(0) Copies the <i>current program</i> (see the <i>OM</i> ) from <i>RAM</i> and appends it to the <i>FM</i> library. 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 <b>CATALOG PROGS</b> and called by <b>XEQ</b> .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
pt. $\rightarrow$ m	f [U $\rightarrow$ ] x: g point $\rightarrow$ m	(1) {2}; {1} $\rightarrow$ {2} Converts print heights. See pp. 146ff.
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. <b>R/S</b> is required to resume program execution then. May help in user interaction with routines (see the OM, Section 3).
P.FN	g [P.FN]	Menu. See p. 134.
P.FN2	g [P.FN] P.FN2	Submenu. See p. 134.
P:	f [U $\rightarrow$ ] P:	Submenu. See p. 146.
R	f [CONST] R	(-1) {} $\rightarrow$ {2} Molar gas constant in joule per mol and kelvin.
RAD	g [MODE] RAD	(0) Sets the ADM to radians. Indicated in the status bar.
RAD $\rightarrow$	g [L $\rightarrow$ ] f RAD $\rightarrow$	(1) {1, 2} $\rightarrow$ {4} Converts angles as described on pp. 154f.
RAM	f [CAT.] PROGS RAM	Submenu of global labels defined at execution time. See pp. 124f.
RANGE	g [DISP] ▲ f RANGE	(0) {1, 2} Limits the range of displayable real numbers to $\pm 10^{\pm n}$ with $n = \text{IP}(x)$ . Startup default is 6145. RANGE allows for saving screen space for reasonable data. $99 \leq n \leq 6145$ . For greater input, $n$ will be set to 6145; for less it will be set to 99.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
RANGE?	<b>g DISP</b> ▲ <b>f RANGE?</b>  <b>g INFO</b> ▲ <b>RANGE?</b>	(-1) {} → {1}  Returns the current range setting, cf. RANGE.
RANI#	<b>g PROB</b>  ▲ <b>RANI#</b>	(-1) {} → {1}  Returns an integer random number $n$ with $\text{IP}[\min(x, y)] \leq n \leq \text{IP}[\max(x, y)]$ . After executing RANI #, the stack looks like $[n, x, y, \dots]$ , so you can drop or roll down $n$ and call RANI # again to get another random integer with the same parameters. Use RANI# e.g. for throwing dices.
RAN#	<b>g PROB</b>  ▲ <b>RAN#</b>	(-1) {} → {2}  Returns a random number between 0 and 1 like RAN does in HP-42S. See also SEED.
RBR	<b>f RBR</b>	Calls the register browser – see the OM, Sect. 5. You may call RBR also in PEM but it is not programmable.  Within RBR, <i>real</i> and <i>complex</i> numbers are generally displayed in their format chosen. Since it uses the small font all the way, more digits can be shown here in ALL 0 than in a numeric row. For <i>reals</i> and ALL 0, RBR display will turn over to SCI or ENG at 33 digits in worst case. Extended display precision may be observed for <i>complex</i> numbers as well.
RCL	<b>RCL</b> <i>r</i>	(-1) Recalls the content of a register or variable.
RCLCFG	<b>RCL</b> <b>f Config</b> <i>r</i>	(0) Recalls a configuration stored by STO CFG (see Sect. 2 and 6 of the OM).
RCLEL	<b>f MATX</b> <b>g RCLEL</b>	(-1) {} → {2, 3}
	<b>RCL</b> <b>g ...EL</b>	Recalls a copy of the current element $a_{ij}$ of the indexed matrix. Cf. STO EL.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
RCLIJ	 	(-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	 Stack	Recalls 4 or 8 values from a set of registers starting at address <i>r</i> , and pushes them on the stack. This is the converse command of STOS.
RCL+		(1) Recalls a content of a register or variable, executes the operation specified, and puts the result on the stack like a monadic function. <sup>17</sup>
RCL-		
RCL×		
RCL /		
RCL↑	 	(1) {1, 2, 4, 5, 6, 10} Replaces <i>x</i> with the maximum of <i>r</i> and <i>x</i> . <sup>17</sup>
RCL↓	 	(1) {1, 2, 4, 5, 6, 10} Replaces <i>x</i> with the minimum of <i>r</i> and <i>x</i> . <sup>17</sup>
RDP		(1) {2, 3, 5, 8*, 9*} Rounds <i>x</i> to <i>n</i> decimal places ( $0 \leq n \leq 99$ , think of FIX format), taking the RM setting into account. See RM and compare RSD.
Re	 	(1) {3} → {2}; {9} → {8} Returns the real part of <i>x</i> . Cf. IM.
r <sub>e</sub>		(-1) {} → {2} Classical electron radius in meter.

<sup>17</sup> 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. 13ff for general information)
REALS	f [CAT.] VARS REALS	Submenu of <i>real</i> variables defined at execution time. See pp. 124f.
REAL?	g [TEST] ▲ REAL?	(0) Checks if $x$ is a <i>real</i> number or matrix.
RECV	f [I/O] f RECV	(0) Prepares your WP 43S for receiving data via serial I/O. See SEND and Sect. 3 in the OM for more.
REGIST	f [CAT.] REGIST	Submenu of register names defined. See pp. 124f.
RESET	g [CLR] g RESET	Executes CLALL and resets everything to <i>startup default</i> , i.e. 2COMPL, ALL 0, DEG, DENMAX 0, DSTACK 4, GAP 3, J/G 1752-01-01, LinF, LocR 0, RM 0, TDISP -1, WSIZE 64, and Y.MD. RANGE is set to 6145.  RESET sets ALLSCI, ASLIFT, DECIM., DENANY, MULTx, PROPF, RECTN, and TDM24. It clears all other system flags (see pp. 112ff).  See said commands and flags for more.
RE→CX	CC  (does not work in PEM)	(2) {2} → {3}  Composes a <i>complex</i> number out of two <i>reals</i> or integers $x$ and $y$ , setting C and taking either... <ul style="list-style-type: none"><li>• (for L) the <i>real</i> part from Y and <i>imaginary</i> part from X, or</li><li>• (for ⊙) <i>magnitude</i> from Y and <i>phase</i> from X.</li></ul>
	g [CPX] f RE→CX	(2) {8} → {9}  Works in analogy for two <i>real</i> matrices $x$ and $y$ .
Re↔Im	g [CPX] Re↔Im	(1) {3, 9*}  Swaps <i>real</i> and <i>imaginary</i> part of <i>complex</i> objects.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
RJ	 	(10) Right justifies a bit pattern within its word size, in analogy to LJ (see there). The stack will lift, placing the right-justified word in Y and the count of bit-shifts necessary to justify the word in X. <b>Example:</b> 10 1100 <sub>2</sub> RJ results in y = 1011 <sub>2</sub> and x = 2.
R <sub>K</sub>		(-1) {} → {2} Von Klitzing constant in ohm.
RL	 	(1) {10} Works like <i>n</i> consecutive RLs on HP-16C, similar to RL <i>n</i> there ( $0 \leq n \leq 63$ ). RL 0 executes as NOP, but loads L. See the OM, Sect. 2, for more.
RLC	 	(1) {10} Works like <i>n</i> consecutive RLCs on HP-16C, similar to RLC <i>n</i> there ( $0 \leq n \leq 64$ ). RLC 0 executes as NOP, but loads L. See the OM, Sect. 2, for more.
RM	 	(0) Sets floating point rounding mode. This rounding mode is used only for RSD or when converting from the extended precision internal format (typically 39 digits) to packed reals. It will not alter the display nor change the behavior of ROUND. The following 7 modes are supported: 0: round half even: $\frac{1}{2}\uparrow$ 0.5 rounds to next even number (default, used in science). 1: round half up: $\frac{1}{2}\uparrow$ 0.5 rounds up ('businessman's rounding' <sup>18</sup> ).

<sup>18</sup> Translator's notes for French and German readers: Cela correspond à l'arrondi commercial. / Das entspricht kaufmännischer Rundung.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
		<p>2: round half down: <math>\frac{1}{2}\downarrow</math> 0.5 rounds down.      3: round up: <math>\leftarrow 0 \rightarrow</math> rounds away from 0.      4: round down: <math>\rightarrow 0 \leftarrow</math> rounds towards 0.      5: ceiling: <math>\lceil x \rceil</math> rounds towards <math>+\infty</math>.      6: floor: <math>\lfloor x \rfloor</math> rounds towards <math>-\infty</math>.</p> <p>The abbreviations printed on grey are used in STATUS for indicating the respective rounding modes (see Section 5 of the OM).</p>
RMD		(2) {1, 2, 10}  Returns the remainder of a division. Equals RMD on HP-16C but works for <i>reals</i> as well. See the OM, Sect. 2, for examples. Cf. MOD.
R <sub>Moon</sub>		(-1) {} → {2}  Mean radius of the Moon in <i>meter</i> .
RM?		(-1) {} → {1}  Returns the floating point rounding mode set. See RM for more.
RNORM	 	(1) {8, 9}  Calculates the row norm of the matrix $x$ , 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.
RootF	 	(0) Selects the root fit model. Relevant for CORR, COV, L.R., $s_{XY}$ , $\hat{x}$ , and $\hat{y}$ . See pp. 233ff for more.
ROUND		(1) {2, 3, 4, 5, 6, 8*, 9*}  Rounds $x$ using the current display format like RND on HP-42S.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
ROUNDI	<b>g</b> <b>DISP</b> <b>ROUNDI</b>	(1) {8*}; {2} → {1} Rounds $x$ to next integer. $\frac{1}{2}$ rounds to 1. Cf. IP.
RR	<b>f</b> <b>BITS</b> <b>RR</b> <b>n</b>	(1) {10}  Works like $n$ consecutive RRs on HP-16C, similar to RRn there ( $0 \leq n \leq 63$ ). RR 0 executes as NOP, but loads L. See the OM, Sect. 2, for more.
RRC	<b>f</b> <b>BITS</b> <b>RRC</b> <b>n</b>	(1) {10}  Works like $n$ consecutive RRCs on HP-16C, similar to RRCn there ( $0 \leq n \leq 64$ ). RRC 0 executes as NOP, but loads L. See the OM, Sect. 2, for more.
RSD	<b>g</b> <b>DISP</b> <b>f</b> <b>RSD</b> <b>n</b>	(1) {2, 3, 4, 8*, 9*} Rounds $x$ to $n$ significant digits ( $1 \leq n \leq 34$ ), taking the RM setting into account. Think of SCI. Cf. RM and RDP.
RSUM	<b>f</b> <b>MATX</b> <b>f</b> <b>RSUM</b>	(1) {8, 9} Calculates the row sum of the matrix $x$ , returning an $m \times 1$ matrix filled with the row sums of the $m \times n$ input matrix.
RTN	<b>g</b> <b>RTN</b>	(0) In PEM, RTN is the logically last command in a routine (see Section 3 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 $\overline{\uparrow}$ is lit.  If pressed in run mode with no routine executing, <b>RTN</b> resets the program pointer to the start of current program (see the OM, Sec. 3). If the program is in FM, the pointer is set to step 0000 in RAM, and $\overline{\uparrow}$ is lit.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
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	 	(0) {2} Interprets $x$ in the form $sss.nn$ . Clears $nn$ registers starting with address $sss$ .  <b>Example:</b> For $x = 34.567$ , R-CLR will clear R34 through R89.  <b>ATTENTION:</b> For $nn = 0$ , clearing will cover the maximum available: <ul style="list-style-type: none"><li>• For <math>sss \in [0; 99]</math>, it will stop at R99.</li><li>• For <math>sss \in [100; 111]</math>, it will stop at K.</li><li>• For <math>sss \geq 112</math>, it will stop at the highest currently allocated local register.</li></ul>
R-COPY	 	(0) {2} Interprets $x$ in the form $sss.nnddd$ . Takes $nn$ registers starting with address $sss$ and copies their contents to $ddd$ etc.  <b>Example:</b> For $x = 7.0304567$ , r07, r08, r09 will be copied into R45, R46, R47, respectively.  For $x < 0$ , R-COPY will take $nn$ registers from FM instead, starting with register number $ sss $ . Destination will be in RAM always.  <b>ATTENTION:</b> For $nn = 0$ , copying will cover the maximum available as explained with R-CLR. Then $x$ must be negative.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
R-SORT	 	(0) {2} Interprets $x$ in the form $sss.nn$ . Sorts the contents of $nn$ registers starting with address $sss$ . <b>Example:</b> 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 $r51 = 1.2$ . <b>ATTENTION:</b> For $nn = 0$ , sorting will cover the maximum available as explained with R-CLR.
R-SWAP	 	(0) {2} Works like R-COPY but swaps the contents of source and destination registers.
R→D		(1) {1, 2} → {4} Converts angles as described on pp. 154f.
R↑		Rotates the stack contents one level up or down, respectively. See Section 1 of the OM for details.
R↓		
R <sub>∞</sub>	R <sub>∞</sub> etc.	(-1) {} → {2}
R <sub>⊖</sub>		Rydberg constant (see p. 143);
R <sub>⊕</sub>		mean radii of the Sun and Earth in meter.
s	s	(-2) {} → {2} Takes the statistical sums accumulated, calculates the sample standard deviations $s_y$ and $s_x$ and pushes them on the stack. See Section 2 of the OM for the output format and pp. 224ff for the formula.
Sa	Sa	(-1) {} → {2} Semi-major axis in meter of the Earth model WGS84. <sup>19</sup>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
SAVE	f SAVE	(0) Saves user program space, registers and system state to FM, and returns Saved. Recall your backup using the different flavors of LOAD.
SB	f BITS g SB n	(1) {10} Sets the specified bit in $x$ .
Sb	f CONST Sb	(-1) {} → {2} Semi-minor axis in meter of WGS84. <sup>19</sup>
SCI	g DISP SCI n	(0) Sets scientific display format (see Section 2 of the OM).
scw→kg	f U m: f short cwt → kg	(1) {2}; {1} → {2} Converts masses. See pp. 146ff.
SDIGS?	g INFO f SDIGS?	(-1) {} → {1} Returns the number of significant digits set by SETSIG. Also returned by STATUS.
SDL	g P.FN P.FN2 f SDL n etc.	(1) {2} Shifts digits left (right) by $n$ decimal positions, equivalent to multiplying (dividing) $x$ by $10^n$ . Cf. SL and SR for binary integers.
Se <sup>2</sup>	f CONST Se <sup>2</sup>	(-1) {} → {2} First eccentricity squared of the Earth model WGS84. <sup>19</sup>
SEED	g PROB ▲ SEED	(0) {2} Stores a seed for random number generation. If $x = 0$ , the seed is taken from the real-time clock.

<sup>19</sup> 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](http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html)

Item	Keystrokes	Remarks (see pp. 13ff for general information)
SEND	<b>f</b> I/O <b>f</b> 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	<b>g</b> DISP ▲ CHINA	(0) Sets regional format preferences. <sup>20</sup>
SETDAT	<b>g</b> CLK ▲ SETDAT	(0) Sets the date for the real-time clock (the emulator takes this information from the PC clock).
SETEUR	<b>g</b> DISP ▲ EUROPE	
SETIND	<b>g</b> DISP ▲ INDIA	(0) Set regional format preferences. <sup>20</sup>
SETJPN	<b>g</b> DISP ▲ JAPAN	
SETSIG	<b>g</b> MODE <b>f</b> SETSIG	(0) {1} Sets the number of significant digits (0 ... 34) for rounding after each operation. SETSIG 0 sets maximum precision.
SETTIM	<b>g</b> CLK ▲ SETTIM	(0) Sets the time for the real-time clock (the simulator takes this information from the PC clock).
SETUK	<b>g</b> DISP ▲ UK	
SETUSA	etc.	(0) Set regional format preferences. <sup>20</sup>
Se'²	<b>f</b> CONST Se'²	(-1) {} → {2} Second eccentricity squared of the Earth model WGS84 (see footnote 19 on p. 74).
SF	<b>g</b> FLAGS SF <i>n</i>	(0) Sets the flag specified.
	<b>g</b> MODE SF <i>n</i>	

<sup>20</sup> See Section 2 of the OM about localization of numeric output.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
Sf <sup>-1</sup>	f CONST Sf <sup>-1</sup>	(-1) {} → {2} Flattening parameter of the Earth model WGS84 (see footnote 19 on p. 74).
SHOW	f SHOW	(0) {1, 2, 3, 4, 7} Shows all digits or characters stored in X until next keystroke in top numeric row, using small font. For <i>complex</i> numbers, either part will take one display row as soon as one row cannot take both parts. For <i>long integers</i> , ≤296 digits are displayed using ≤7 rows; for any greater integer, just its most significant 294 digits will be shown, trailed by ellipses.
SIGN	f PARTS sign	(1) {8}, {1, 2, 10} → {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( $x$ ).
		(1) {3} Returns the unit vector of the <i>complex</i> number $x$ (cf. UNITV). Maintained for backward compatibility only.
SIGNMT	f BITS ▼ SIGNMT g INTS ▲ SIGNMT	(0) Sets sign-and-mantissa mode for operations on <i>short integers</i> . See the OM, Section 2. Indicated in the <i>status bar</i> .
SIM_EQ	f MATX SIM EQ n	(0) Solves a system of $n$ linear equations $(MATA) \cdot \vec{MATX} = \vec{MATB}$ . If these matrices are not defined before, they will be created automatically at execution time. See Sect. 2 of the OM for more.
sin	TRI sin	(1) [2, 3, 8*, 9*]; {1, 4} → {2} Returns the sine of the angle in X.
sinc	g X.FN ▲ sinc	(1) [2, 3, 8*, 9*] Returns $\frac{\sin(x)}{x}$ for $x \neq 0$ and 1 for $x = 0$ . Note input shall be supplied in <i>radians</i> .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
sinh	<b>g EXP g sinh</b>	(1) {2, 3, 8*, 9*}
	<b>[TRI] g sinh</b>	Returns the hyperbolic sine of $x$ .
SKIP	<b>g P.FN P.FN2</b> <b>g SKIP n</b>	(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.  <b>ATTENTION:</b> 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	<b>f BITS</b> <b>▲ SL n</b>	(1) (10)  Works like $n$ ( $\leq 63$ ) consecutive SLs on HP-16C. SL 0 executes as NOP, but loads L. See Sect. 2 of the OM for more.
SLVQ	<b>f ADV</b> <b>SLVQ</b>	{1, 2, 3} → {1 or 2 or 3} Solves the quadratic equation $ax^2 + bx + c = 0$ with its parameters on the input stack [ $c, b, a, \dots$ ], and tests the result. The following holds for <i>real</i> parameters: <ul style="list-style-type: none"><li>• If <math>r := b^2 - 4ac \geq 0</math>, SLVQ returns <math>-\frac{b \pm \sqrt{r}}{2a}</math> in Y and X. In a routine, the step after SLVQ will be executed.</li><li>• Else, SLVQ returns the first <i>complex</i> root in X and the second in Y (the <i>complex conjugate</i> of the first). In a routine, the step after SLVQ will be skipped.</li></ul> In either case and also for <i>complex</i> parameters, SLVQ returns r in Z. Higher stack registers are kept unchanged. L will contain equation parameter c.

Item	Keystrokes	Remarks (see pp. 13ff for general information)												
$s_m$		(-2) {} → {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).												
S MODE?		(-1) {} → {1} Returns the <i>integer sign mode</i> set for <i>short integers</i> : <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 &amp; 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.												
$s_{mw}$		(-1) {} → {2} Returns the <i>standard error</i> for weighted data, i.e. the <i>standard deviation</i> of the mean $\bar{x}_w$ .												
SNAP		(0) Stores a snapshot of the screen in a BMP file on the calculator's USB flash disk in the /SCREENS directory. The file name comprises date and time of storage.												
SOLVE		{2, 3} Solves the equation $f(var) = 0$ , with $f$ calculated by the equation specified (in PEM by PGMSLV). Two initial estimates of the root must be supplied in <b>X</b> and <b>Y</b> when calling SOLVE. It returns $var_{root}$ in <b>X</b> , the second last <b>var</b> -value tested in <b>Y</b> , then $f(var_{root})$ in <b>Z</b> , and 0 in <b>T</b> . Additionally, SOLVE acts as binary test in programs, so the next program step will be skipped if SOLVE fails to find a root. See Section 4 of the OM for more. <b>ATTENTION:</b> SOLVE fills all stack registers with $x$ before calling the routine specified.												

Item	Keystrokes	Remarks (see pp. 13ff for general information)
Solver	<b>g EQN Solver</b>	<b>Submenu</b> for solving a given equation. See the OM, Sect. 4, for more.
SPEC?	<b>g TEST ▲ f SPEC?</b>	(0) True if $x$ is ‘special’ (i.e. $\pm\infty$ or NaN).
SR	<b>f BITS</b> ▲ SR <b>n</b>	(1) {10} 0 → [ ] → C  Works like $n$ ( $\leq 63$ ) consecutive SRs on HP-16C. SR 0 executes as NOP, but loads L. See Section 2 of the OM for more.
SSIZE?	<b>g INFO SSIZE?</b>	(-1) {} → {1}  Returns the number of <i>stack registers</i> currently allocated, 4 or 8.
STAT	<b>f STAT</b>	<b>Menu</b> . See p. 134.
STATUS	<b>f STATUS</b>	<b>Flag browser</b> . See Section 5 of the OM.
	<b>g FLAGS STATUS</b>	
STK	<b>g STK</b>	<b>Menu</b> . See p. 134.
STO	<b>STO r</b>	(0) Stores $x$ into destination.
STOCFG	<b>STO f Config r</b>	(0) Stores the current <i>configuration</i> for later use as described in Section 2 of the OM. RCLCFG recalls such data.
STOEL	<b>f MATX g STOEL</b>	(1) {1, 2, 3}
	<b>STO g ...EL</b>	Stores a copy of $x$ into the indexed matrix at the current element, $a_{ij}$ . Cf. RCLEL.
STOIJ	<b>f MATX ▲ STOIJ</b>	(1) {1}
	<b>STO g ...IJ</b>	Sets the index pointers to $IP(x)$ (= column number) and $IP(y)$ (= row number). Cf. RCLIJ.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
STOP	[R/S]	(0) Stops program execution. May be inserted in programs to wait for input, for example.
STOS	[STO] f Stack r	(0) Stores the entire stack in a set of 4 or 8 registers, starting at the destination address specified. See RCLS.
STO+	[STO] + r	
STO-	[STO] - r	
STOx	[STO] x r	
STO/	[STO] / r	
sto→kg	f U→ m: f stone → kg	(1) {2}; {1} → {2} Converts masses. See pp. 146ff.
STO↑	[STO] f Max r	
	[STO] ▲ r	(0) {1, 2, 4, 5, 6, 10}
STO↓	[STO] f Min r	
	[STO] ▼ r	Stores the maximum (or minimum) of r and x in the address specified. <sup>21</sup>
STR??	g TEST g STR??	(0) True if x is an alphanumeric string (like STR? in HP-42S).
STRING	f CATALOG VARS STRING	Submenu of alpha string variables defined at execution time. See pp. 124f.
SUM	f STAT SUM	(-2) {} → {2} Recalls the linear sums $\Sigma y$ and $\Sigma x$ . Useful in basic 2D vector algebra. Output is labeled in analogy to s.

<sup>21</sup> 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. 13ff for general information)
$s_w$	<b>f</b> <b>STAT</b> <b>f</b> <b>s<sub>w</sub></b>	(-1) {} → {2} Calculates the <i>standard deviation</i> for weighted data (where the weight $y$ of each data point $x$ was entered via <b>Σ+</b> ). See pp. 224ff for the formula.
$s_{xy}$	<b>f</b> <b>STAT</b> <b>▲</b> <b>s<sub>xy</sub></b>	(-1) {} → {2} Calculates the <i>sample covariance</i> for the two data sets entered via <b>Σ+</b> , depending on the curve fit model selected. See pp. 224ff for the formula and COV for the <i>population covariance</i> .
SYSTEM	<b>g</b> <b>MODE</b> <b>g</b> <b>SYSTEM</b>	(0) Returns to DMCP. Works on the calculator only (not on the simulator). See App F.
SYS.FL	<b>f</b> <b>CATALOG</b> <b>SYS.FL</b>	Submenu of system flags defined. See p. 124.
$s(a)$	<b>f</b> <b>STAT</b> ▲ <b>f</b> <b>s(a)</b>	(-2) {} → {2} Pushes the standard errors $s(a_1)$ (in <b>Y</b> ), and $s(a_0)$ (in <b>X</b> ) for the parameters of the line fitted through the data points accumulated in the statistical summation registers on the stack. Works for LINF only. See pp. 239f for more.
S.INTS	<b>f</b> <b>CAT.</b> <b>VARS</b> <b>S.INTS</b>	Submenu of short integer variables defined at execution time. See pp. 124f.
$s.t \rightarrow kg$	<b>f</b> <b>U→</b> <b>m:</b> ▲ <b>short ton</b> → <b>t</b>	(1) {2}; {1} → {2}
$s \rightarrow year$	<b>f</b> <b>U→</b> <b>f</b> <b>s → year</b>	Convert masses and times. See pp. 146ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$T_0$	<b>f CONST</b> $T_0$	(-1) {} → {2} Standard temperature ( $0^\circ\text{C}$ ) in <i>kelvin</i> .
$\tan$	<b>TRI tan</b>	(1) {2, 3, 8*, 9*}; {1, 4} → {2} Returns the tangent of the angle in <b>X</b> . Returns “ <i>Not a Number</i> ” for $x = \pm 90^\circ$ or equivalents if SPCRES is set.
$\tanh$	<b>g EXP g tanh</b>	(1) {2, 3, 8*, 9*} Returns the hyperbolic tangent of $x$ .
	<b>TRI g tanh</b>	
TDISP	<b>g CLK</b> <b>▲ TDISP n</b>	(0) Sets time display format. TDISP <b>0</b> and <b>1</b> allow for displaying just <i>hours</i> and <i>minutes</i> , TDISP <b>2</b> for <i>seconds</i> , too, and $n \geq 3$ also for $n-2$ digits showing decimal fractions of <i>seconds</i> . TDISP <b>-1</b> allows for displaying all digits.
TEST	<b>g TEST</b>	<b>Menu</b> . See p. 134.
$t_p(x)$	<b>g PROB</b> <b>t: t<sub>p</sub>(x)</b> etc.	(1) {2} <i>Student's t distribution</i> . The degrees of freedom are stored in <b>J</b> . $t_e(x)$ equals $Q(t)$ on HP-21S. See Section 2 of the OM for an application example and pp. 224ff for more mathematical details.
$t_{\Delta}(x)$		
$t_{\Delta}(x)$		
$t^{-1}(p)$		
TICKS	<b>g P.FN TICKS</b>	(-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.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
TIME	<b>g CLK TIME</b>	(-1) {} → {5} Recalls the time from the real-time clock at execution (see Sect. 2 of the OM for the output format).
TIMER	<b>f TIMER</b>	Starts the timer application based on the real-time clock and following the timer of HP-55. See the OM, Sect. 5 for a detailed description.
TIMES	<b>f CATALOG VARS f TIMES</b>	Submenu of time variables defined at execution time. See pp. 124f.
$T_n$	<b>g X.FN Orthog T<sub>n</sub></b>	(2) {2}; {1} → {2} <i>Chebyshev polynomials of first kind.</i> See pp. 239f for details.
TONE	<b>f I/O f TONE n</b>	(0) Sounds a tone according to $n$ (= 1 ... 9).
ton→kg	<b>f U→ m: ▲ ton→kg</b>	(1) {2}; {1} → {2} Converts masses. See pp. 146ff.
TOP?	<b>g TEST g TOP?</b>	(0) Returns ... <ul style="list-style-type: none"> <li>• <b>false</b> if called with the program pointer being in a subroutine;</li> <li>• <b>true</b> if called in the top routine (i.e. if the program-running flag is set and the SRS pointer is clear).</li> </ul>
tor→Pa	<b>f U→ F&amp;p: f torr → Pa</b>	(1) {2}; {1} → {2} Converts pressures. See pp. 146ff.
$T_p$	<b>f CONST T<sub>p</sub></b>	(-1) {} → {2}
$t_{PL}$	<b>f CONST t<sub>PL</sub></b>	<i>Planck temperature in kelvin;</i> <i>Planck time in seconds.</i>
TRANS	<b>f CAT. FCNS TRANS</b>	Works like [M] <sup>T</sup> on p. 102. Maintained for backward compatibility only.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
TRI	[TRI]	<b>Menu.</b> See p. 134.
trz→kg	[f] [U↔] m: [f] tr.oz → kg	(1) {2}; {1} → {2} Converts masses. See pp. 146ff.
TVM	[g] [FIN] TVM	<b>Submenu</b> for the application <i>Time Value of Money</i> . See <i>Section 5</i> of the OM.
t:	[g] [PROB] t:	<b>Submenu.</b> See p. 133.
t⇄	[g] [STK] t⇄ r	Swaps $t$ and $r$ , in analogy to $x\leftrightarrow$
ULP?	[g] [INFO] [f] ULP?	(1) {1, 2}  Returns 1 times the smallest power of ten which can be added to $x$ or subtracted from $x$ to actually change the (internal) value of $x$ in your WP 43S in the mode set. Thus, 1 is returned for integers. Indicated in STATUS.
U <sub>n</sub>	[g] [X.FN] Orthog U <sub>n</sub>	(2) {2}; {1} → {2} <i>Chebyshev polynomials of second kind.</i> See pp. 239f for details.
UNITY	[f] [MATX] [f] UNITY	(1) {8, 9}  Returns the unit vector for the matrix $x$ (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 <i>magnitude</i> will become 1.
	[g] [CPX] UNITY	(1) {3}  Returns a <i>complex</i> number with <i>magnitude</i> $ r  = 1$ in direction of $x$ .
UNSIGN	[f] [BITS] ▼ UNSIGN	(0) Sets unsigned mode for mode for operations on <i>short integers</i> . Indicated in the <i>status bar</i> . Cf. UNSGN on HP-16C. See <i>Section 2</i> of the OM.
	[g] [INTS] ▲ ...	

Item	Keystrokes	Remarks (see pp. 13ff for general information)
U→	f U→	Menu. See p. 134.
VAR		Submenu of variables defined at execution time when calling STO, RCL, etc.
VARMNU	g P.FN f VARMNU <i>label</i>	Creates a variable menu using the 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. 124f.
VERS?	g INFO g VERS?	(0) Shows your firmware version and build number (see Section 2 of the OM).
VIEW	g VIEW <i>r</i>	(0) Shows <i>r</i> until the next key is pressed.  <b>Example:</b> If a variable called <b>Test12</b> contains <b>-123.45</b> , VIEW @ Test12 ENTER↑ will display  Test12 = -123.45
v <sub>m</sub>	f CONST v <sub>m</sub>	(-1) { } → {2}  Molar volume of an ideal gas at standard conditions in <i>cubic meter per mol</i> .
v:	f U→ f v:	Submenu. See p. 146.
v <sub>4</sub>	g ↵	(2) {8} → {4}  Returns the angle between two 2D or 3D vectors $\vartheta = \arccos\left(\frac{\vec{v}_1 \cdot \vec{v}_2}{ \vec{v}_1  \vec{v}_2 }\right)$

Item	Keystrokes	Remarks (see pp. 13ff for general information)
WDAY	<b>g CLK WDAY</b>	(1) {2, 6} → {1} Assumes $x$ containing a <i>date</i> in the format selected (or a <i>real</i> number in corresponding format) and returns the name of the respective day and a corresponding integer (Monday = 1). <sup>22</sup>
Weibl <sub>p</sub>	<b>g PROB</b>	(1) {2}
Weibl <sub>▲</sub>	<b>f Weibl</b> Weibl etc.	Weibull distribution with its shape parameter $b$ in I and its characteristic lifetime $T$ in J. See pp. 224ff for details.
Weibl <sub>▲</sub>		
Weibl <sup>-1</sup>		Weibl <sup>-1</sup> returns the survival time $t_s$ for a given probability $p$ in X, with $b$ in I and $T$ in J.
Weibl:	<b>g PROB f Weibl:</b>	Submenu. See p. 133.
WHO?	<b>g INFO g WHO?</b>	(0) Displays credits to the brave men who made this project work.
Wh→J	<b>f U→ E: Wh→J</b>	(1) {2}; {1} → {2} Converts energies. See pp. 146ff.
W <sub>m</sub>	<b>g X.FN</b> ▲ W <sub>m</sub>	(1) {2, 3}; {1} → {2}
W <sub>p</sub>	etc.	W <sub>p</sub> returns the principal branch of Lambert's W for given $x \geq -1/e$ . W <sub>m</sub> returns its negative branch (working for $x \in \mathbb{R}$ only). W <sup>-1</sup> returns $x$ for a given W <sub>p</sub> ( $\geq -1$ ). See pp. 256ff for more.
W <sup>-1</sup>		

<sup>22</sup> 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. 13ff for general information)
WSIZE	<b>f</b> [BITS] ▼ WSIZE <b>n</b>	(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 <b>X</b> .  Reducing word size truncates the values in the stack as allocated and in <b>L</b> . All other memory content stays as is (see <i>App. B</i> on pp. 167ff).
	<b>g</b> [INTS] ▲ WSIZE <b>n</b>	Increasing the word size will add empty bits to each stack register. WSIZE 0 sets the word size to maximum, i.e. 64 bits.  WSIZE is indicated in the <i>status bar</i> .
WSIZE?	<b>g</b> [INFO] <b>g</b> WSIZE?	(-1) {} → {1}  Recalls the word size set. Indicated in the <i>status bar</i> .
W→hp <sub>E</sub>	<b>f</b> [U→] P: W→hp <sub>E</sub> etc.	
W→hp <sub>M</sub>		(1) {2}; {1} → {2}
W→hp <sub>UK</sub>		Convert powers. See pp. 146ff.
$\hat{x}$	<b>f</b> [STAT] <b>▲</b> $\hat{x}$	(1) {2}; {1} → {2}  Returns a forecast $\hat{x}$ for a given $y$ (in <b>X</b> ) according to the curve fit model chosen. See L.R. for more.
$\bar{x}$	<b>f</b> [STAT] <b>  x</b>	(-2) {} → {2}  Calculates the <i>arithmetic means</i> of the $y$ - and $x$ -data accumulated and pushes them on the stack. See also $s$ , $s_m$ , and $\sigma$ .
$x^2$	<b>x<sup>2</sup></b>	(1) {1, 2, 3, 8*, 9*, 10}
$x^3$	<b>g</b> [EXP] <b>x<sup>3</sup></b>	Return the square or cube of $x$ , respectively.
XEQ	<b>XEQ</b> <b>label</b>	(0) Executes the function or routine with the label specified. – In <i>PEM</i> , inserts a call to the subroutine with the label specified.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\bar{x}_G$		(-2) {} → {2} Calculates the <i>geometric means</i> of the y- and x-data accumulated and pushes them on the stack. See pp. 233ff for the formula. Output format will be similar to the one of $\bar{x}$ . See also $\varepsilon$ , $\varepsilon_m$ , and $\varepsilon_p$ .
$\bar{x}_H$		(-2) {} → {2} Calculates the <i>harmonic means</i> of the y- and x-data accumulated and pushes them on the stack.
xIN	with <b>type</b> = NILADIC, MONADIC, DYADIC, TRIADIC, or ..._COMPLEX defines how many stack levels are used for parameter input to the function under consideration. Furthermore it does some initialization work (e.g. set SSIZE8).  xIN is the recommended way to start an XROM routine. Thereafter, SSIZE8 is clear. Note xIN cannot nest and XROM routines using xIN cannot call user code.	
XNOR		(2) Work in analogy to AND. See p. 20.
XOR		
xOUT	with <b>way</b>	Cleans and reverts the settings of xIN, taking care of a proper return including the correct setting of <i>I</i> and the stack. Typically, <b>way</b> = xOUT_NORMAL. Generally, xOUT shall be the last command of an XROM routine.
$\bar{x}_{RMS}$		(-2) {} → {2} Calculates the <i>quadratic means</i> of the y- and x-data accumulated and pushes them on the stack. Quadratic means are often used in electrical engineering for alternating currents.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\bar{x}_w$	f STAT f $\bar{x}_w$	(-1) {} → {2} Returns the <i>arithmetic mean</i> for weighted data (where the weight $y$ of each data point $x$ was entered via $\Sigma+$ ). See pp. 233ff for the formula. See also $s_w$ and $s_{mw}$ .
$\sqrt[x]{y}$	g EXP $\sqrt[x]{y}$	(2) Returns the $x^{\text{th}}$ root of $y$ . Roots of negative integers or <i>reals</i> will return <i>complex numbers</i> if CPXRES is set.
X.FN	g X.FN	Menu. See p. 136.
$x!$	f $x!$	(1) {1, 10} Returns the <i>factorial</i> $n!$ . Note this is only defined for positive integers. $20!$ is the biggest factorial $< 2^{64}$ . $450!$ is maximum allowed for <i>long integers</i> .  (1) {2, 3} Returns $\Gamma(x + 1)$ . $2\ 123.549\ 956\ 662\ 463\ 236\ 31$ is the maximum $x$ for <i>reals</i> .
$x:$	f U→ $x:$	Submenu. See p. 146.
$x \rightarrow \text{DATE}$	g CLK $x \rightarrow \text{DATE}$	(1) {2} → {6} Interprets the <i>real/number</i> $x$ as a date coded in the date format selected (Y.MD, D.MY, or M.DY) and converts it to a proper <i>date</i> .
$x \rightarrow \alpha$	g a.FN $x \rightarrow \alpha$	(1) {1, 2, 10} → {7} Interprets the integer part of $x$ as a character code and converts it to the respective character $x$ , similar to XTOA in the HP-42S.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$x \leftrightarrow r$	<b>g</b> [STK] $x \leftrightarrow r$	Swaps $x$ and $r$ , analogous to $x \leftarrow y$ . Will be listed like $x \leftrightarrow J$ , $x \leftrightarrow .12$ , $x \leftrightarrow \rightarrow 12$ , etc.
$x \leftrightarrow y$	$x \leftrightarrow y$	Swaps the contents of stack registers <b>X</b> and <b>Y</b> .
$x = ?$	<b>g</b> [TEST] $x = ? r$ etc.	(0)
$x \neq ?$		Compare $x$ with $r$ . See $x < ?$ for more.
$x = +0?$	<b>g</b> [TEST] $\Delta$ $x = +0?$ etc.	(0) {2, 3, 10} These tests are for comparing <i>short integers</i> in modes 1COMPL and SIGNMT, and for <i>real</i> or <i>complex</i> numbers if SPCRES is set. Then e.g. $0./(-7)$ will display $-0$ .
$x = -0?$		
$x \approx ?$	<b>g</b> [TEST] $\Delta$ $x \approx ? r$	(0) {2, 3, 4, 5, 8, 9} Will be true if the <u>rounded</u> values of $x$ and $r$ are equal (see ROUND). See $x < ?$ for more.
$x < ?$	<b>g</b> [TEST] $x < ? r$ etc.	(0) {1, 2, 4, 5, 6, 7, 10} Compare $x$ with $r$ . Strings are compared according to their sorting order.
$x \leq ?$		
$x \geq ?$		
$x > ?$		
$\hat{y}$	<b>f</b> [STAT] $\Delta$ $\hat{y}$	(1) {2}; {1} $\rightarrow$ {2} Returns a forecast $y$ (in <b>X</b> ) for a given $x$ according to the curve fit model chosen. See L.R. for more.
$yd.\rightarrow m$	<b>f</b> [U $\rightarrow$ ] $x:$ <b>g</b> $yd.\rightarrow m$	(1) {2}; {1} $\rightarrow$ {2} Converts distances. See pp. 146ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
YEAR	<b>g CLK f YEAR</b>	(1) {2, 6} → {1} Assumes $x$ containing a <i>date</i> in the format selected (or a <i>real</i> number in corresponding format) and extracts the year.
year→s	<b>f U→ f year→s</b>	(1) {2}; {1} → {2} Converts times. See pp. 146ff.
$y^x$	<b><math>y^x</math></b>	(2) {1, 2, 3, 10} Returns the $x^{\text{th}}$ power of $y$ . It allows for raising any positive <i>real</i> number to an arbitrary <i>real</i> power, as well as any negative <i>real</i> number to an arbitrary integer power, all returning <i>reals</i> . Exceeding these boundaries may produce <i>complex</i> results if CPXRES is set.
Y.MD	<b>g CLK ▲ Y.MD</b>	(0) Sets the format yyyy-mm-dd for <i>dates</i> .
$y \leftrightarrow r$	<b>g STK <math>y \leftrightarrow r</math></b>	Swaps $y$ and $r$ , in analogy to $x \leftrightarrow$ .
$Z_0$	<b>f CONST <math>Z_0</math></b>	(-1) {} → {2} Characteristic impedance of vacuum in <i>ohm</i> .
$z \leftrightarrow r$	<b>g STK <math>z \leftrightarrow r</math></b>	Swaps $z$ and $r$ , in analogy to $x \leftrightarrow$ .
$\alpha$	<b>f CONST <math>\alpha</math></b>	(-1) {} → {2} Fine-structure constant.
$\alpha\text{INTL}$	<b>f CAT. CHARS <math>\alpha\text{INTL}</math></b>	Submenu. See pp. 124ff.
	<b>g +</b>	Menu in AIM (see p. 136).
$\alpha\text{LENG?}$	<b>g INFO g <math>\alpha\text{LENG? } r</math></b>	(-1) {} → {1}
	<b>g <math>\alpha\text{FN}</math> f <math>\alpha\text{LENG? } r</math></b>	Returns the number of characters found in $r$ , similar to ALENG in HP-42S. <sup>23</sup>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
αMATH	f [CAT.] CHARS αMATH	Submenu. See pp. 124ff.
	g [–]	Menu in AIM. See p. 137.
αPOS?	g [INFO] g αPOS? r	(-1) {} → {1} Looks in r for the <b>target</b> given in X. If a match is found, αPOS returns the position number where the <b>target</b> was found (counting the left-most character as position 0). If a match is not found, αPOS returns -1. <sup>23</sup>
	g [a.FN] f αPOS? r	The <b>target</b> may be an individual character code or an <i>alpha string</i> . αPOS saves a copy of the <b>target</b> in L. It works similar to POSA in HP-42S.
αRL	g [a.FN] αRL r	(0) Rotates r by x characters like AROT in HP-42S, but with $x \geq 0$ . αRL 0 executes as NOP, but loads L. <sup>23</sup>
αRR	g [a.FN] αRR r	(0) Works like αRL but rotates to the right.
αSL	g [a.FN] αSL r	(0) Shifts the x leftmost characters out of r, like ASHF in HP-42S. This allows for deleting the first x characters in the string. αSL 0 executes as NOP, but loads L. <sup>23</sup>
αSR	g [a.FN] αSR r	(0) Works like αSL but for the x rightmost characters out of r, deleting the last x characters in the string. <sup>23</sup>
α.FN	g [a.FN]	Menu. See p. 137.
A...Ω	f [CAT.] CHARS A...Ω	Submenu of Greek letters, see pp. 124ff.

<sup>23</sup> This command will throw an error if there is no string or an empty string in r at execution time.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\alpha$ -	<b>f CAT.</b> CHARs $\alpha$ -	Submenu. See pp. 124ff.
	<b>g</b>	Menu in AIM. See p. 137.
$\alpha \rightarrow x$	<b>g</b> <b>a.FN</b> $\alpha \rightarrow x$ <b>r</b>	(-1) {} → {10 <sub>16</sub> } Pushes the character code of the leftmost character in <b>r</b> on the stack and removes this character from the string, similar to ATOX in HP-42S. <sup>23</sup>
$\beta(x,y)$	<b>g</b> <b>X.FN</b> $\beta(x,y)$	(2) {2, 3}; {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 $\beta$ here to avoid ambiguity. See $\Gamma(x)$ below.
$\gamma$	<b>f CONST</b> $\gamma$	(-1) {} → {2} Newtonian constant of gravitation (also called G by other authors) in $m^3/kg\ s^2$ ;
$\gamma_{EM}$	<b>f CONST</b> $\gamma_{EM}$	Euler-Mascheroni constant (for mathematics);
$\gamma_p$	<b>f CONST</b> $\gamma_p$	proton gyromagnetic ratio (see p. 144).
$\Gamma_{xy}$	<b>g</b> <b>X.FN</b> $\Gamma_{xy}$	(2) {2}; {1} → {2}
$\gamma_{xy}$	<b>g</b> <b>X.FN</b> <b>f</b> $\gamma_{xy}$	Returns the lower / upper incomplete Gamma function. See pp. 256ff for more.
$\Gamma(x)$	<b>g PROB</b> $\Gamma(x)$	(1) {2, 3}; {1} → {2} Returns $\Gamma(x)$ . Note <b>x!</b> calls $\Gamma(x + 1)$ . See also LNF.
$\delta x$	<b>f CAT.</b> PROGS $\delta x$	Predefined global label for $f'(x)$ and $f''(x)$ – see Section 4 of the OM.
$\Delta\nu_{Cs}$	<b>f CONST</b> $\Delta\nu_{Cs}$	(-1) {} → {2} Hyperfine transition frequency of <sup>133</sup> Cs in hertz.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\Delta\%$	<b>f</b> <b>Δ%</b>	(1) {2}; {1} → {2} Returns $100 \frac{x-y}{y}$ leaving $y$ unchanged, like %CH in HP-42S. Use it also for calculating mark-ups or margins as explained in the OM, Sect. 2.
$\varepsilon$	<b>f</b> <b>STAT</b> <b>g</b> <b>ε</b>	(-2) {} → {2} Calculates the <i>scattering factors</i> $\varepsilon_y$ and $\varepsilon_x$ for <i>log-normally</i> distributed sample data and pushes them on the stack. $\varepsilon$ 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. 224ff for more information.
$\varepsilon_0$	<b>f</b> <b>CONST</b> <b>ε<sub>0</sub></b>	(-1) {} → {2} Electric constant or vacuum permittivity in <i>ampere-second per volt-meter</i> .
$\varepsilon_m$	<b>f</b> <b>STAT</b> <b>g</b> <b>ε<sub>m</sub></b>	(-2) {} → {2} Works like $\varepsilon$ above 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> ).
$\varepsilon_p$	<b>f</b> <b>STAT</b> <b>g</b> <b>ε<sub>p</sub></b>	(-2) {} → {2} Works like $\varepsilon$ but returns the <i>scattering factors</i> of the two populations.
$\zeta(x)$	<b>g</b> <b>X.FN</b> <b>▲</b> <b>f</b> <b>ζ(x)</b>	(1) {2, 3} Returns <i>Riemann's Zeta</i> . See p. 259 for more.
$\lambda_c$	<b>f</b> <b>CONST</b> <b>λ<sub>c</sub></b>	(-1) {} → {2}
$\lambda_{cn}$	etc.	<i>Compton wavelength of the electron, neutron, and proton in meter.</i>
$\lambda_{cp}$		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\mu_0$	f CONST $\mu_0$ etc.	(-1) {} → {2} Magnetic constant or vacuum permeability in <i>volt-second per ampere-meter</i> ; Bohr magneton and electron magnetic moment in <i>joule per tesla</i> .
$\mu_B$		
$\mu_e$		
$\mu_e/\mu_B$		
$\mu_n$		Ratio of electron magnetic moment to <i>Bohr magneton</i> ;
$\mu_p$		neutron and proton magnetic moments, nuclear magneton, and
$\mu_u$		Muon magnetic moment in <i>joule per tesla</i> .
$\mu_\mu$		
$\pi$	g π	(-1) {} → {2} Recalls $\pi$ .
$\Pi_n$	f ADV $\Pi_n$ <i>label</i>	Computes a product using the routine specified. See Section 4 of the OM for more. <b>ATTENTION:</b> $\Pi_n$ fills all <i>stack registers</i> with $x$ before calling the routine specified.
$\Sigma$	g Σ	Menu. See p. 138.
$\sigma$	f STAT σ	(-2) {} → {2} Works like s but returns the <i>standard deviations</i> of the two <i>populations</i> instead. See pp. 224ff.
$\sigma_B$	f CONST $\sigma_B$	(-1) {} → {2} Stefan-Boltzmann constant (see p. 146).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\Sigma^1/x$	<b>g</b> <b>[Σ]</b> <b>▲</b> <b>Σ<sup>1</sup>/x</b> etc.	<p>(-1) {} → {2}</p> <p>Recall the corresponding statistical sums, necessary for means and regressions beyond the linear model. Calling these 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. 167ff.).</p>
$\Sigma^1/x^2$		
$\Sigma^1/y$		
$\Sigma^1/y^2$		
$\Sigma \ln^2 x$	<b>g</b> <b>[Σ]</b> <b>f</b> <b>Σ ln<sup>2</sup> x</b> etc.	<p><b>ATTENTION:</b> Depending on your input data, logarithmic or inverted sums may become infinite or even non-numeric. If this happens no error will be thrown, regardless of the status of SPCRES.</p> <p>For space reasons, two sums are abbreviated:</p> <p><b>Σlnxy</b> denotes <math>\Sigma \ln(x)\ln(y)</math> and</p> <p><b>Σlny/x</b> denotes <math>\Sigma \frac{\ln(y)}{x}</math>.</p>
$\Sigma \ln^2 y$		
$\Sigma \ln x$		
$\Sigma \ln xy$		
$\Sigma \ln y$		
$\Sigma \ln y/x$		
$\Sigma_n$	<b>f</b> <b>ADV</b> <b>Σ<sub>n</sub> labl</b>	<p>Computes a sum using the routine specified. See Section 4 of the OM for more.</p> <p><b>ATTENTION:</b> Σ fills all <i>stack registers</i> with <math>x</math> before calling the routine specified.</p>
$\sigma_w$	<b>f</b> <b>STAT</b> <b>f</b> <b>σ<sub>w</sub></b>	(-1) {} → {2}
		Works like $s_w$ but returns the <i>standard deviation</i> of the <i>population</i> instead. See pp. 224ff.
$\Sigma x$	<b>g</b> <b>[Σ]</b> <b>Σx</b> etc.	<p>(-1) {} → {2}</p> <p>Recall the corresponding statistical sums, necessary for statistical analyses and regressions (see <math>\Sigma^1/x</math> above for more).</p>
$\Sigma x^2$		
$\Sigma x^2 y$		
$\Sigma x^2/y$		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\Sigma x^3$	 etc.	
$\Sigma x^4$		
$\Sigma xlny$		
$\Sigma xy$		
$\Sigma x/y$		$(-1) \{ \} \rightarrow \{2\}$ Recall the corresponding statistical sums, necessary for statistical analyses and regressions (see $\Sigma^1/x$ above for more).
$\Sigma y$	 etc.	
$\Sigma y^2$		
$\Sigma ylnx$		
$\Sigma+^{24}$		$\{8\} \rightarrow \{2\}$ If $X$ contains an $n \times 2$ matrix then $\Sigma+$ adds $n$ 2D data points to the statistical sums. Then the display will show the last data point added and the matrix will be in $L$ .
		$\{1, 2\}$ Adds one 2D data point to the statistical sums.
$\Sigma-^{24}$		$\{1, 2\}$ Subtracts one 2D data point from the statistical sums.
$\Phi$		$(-1) \{ \} \rightarrow \{2\}$
$\Phi_0$		Golden ratio; magnetic flux quantum in <i>volt-second</i> .

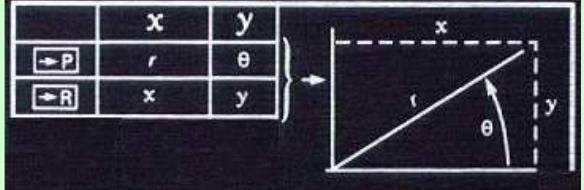
<sup>24</sup>  $\Sigma+$  and  $\Sigma-$  return *temporary information* as shown in Section 2 of the OM and disable *automatic stack lift*. Both commands may also be used for 2D vector adding and subtracting (see SUM and the corresponding example in Section 2 of the OM).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\chi^2_p(x)$	 $\chi^2(x)$ etc.	(1) {2}  <i>Chi-square distribution</i> (with its degrees of freedom given in I). $\chi^2_e(x)$ equals $Q(\chi^2)$ on HP-21S. See Section 2 of the OM for an application example and pp. 224ff for more.
$\chi^2:$	 $\chi^2:$	<b>Submenu.</b> See p. 133.
$\omega$	 $\omega$	(-1) {} → {2}  Angular velocity of the Earth in <i>radians per second</i> according to WGS84 (see footnote 19 on p. 74).
$(-1)^x$	 	(1) {1, 2, 3, 8*, 9*, 10}  If $x$ is non-integer, returns $\cos(\pi x)$ .
$[M]^T$	 $[M]^T$	(1) {8, 9}  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.
$[M]^{-1}$	 $[M]^{-1}$	(0) {8, 9}  Takes the square matrix in $X$ and inverts it in-situ (like INVRT on HP-42S).
$+$		(2) Returns $y + x$ for compatible objects. <sup>25</sup>
$+/-$	 (for closed input)	(1) ‘Unary minus’, returns $x \times (-1)$ . <sup>25</sup>

<sup>25</sup> See the tables in the OM, Section 2 for details and compatibility.

Item	Keystrokes	Remarks (see pp. 13ff for general information)									
$\pm\infty?$	 	(0) {2} $\rightarrow$ {1} Tests $x$ for infinity. Returns <table style="margin-left: 20px; border-collapse: collapse;"> <tr><td>true</td><td style="text-align: right;">1</td><td>for <math>x = +\infty</math>,</td></tr> <tr><td>true</td><td style="text-align: right;">-1</td><td>for <math>x = -\infty</math>, and</td></tr> <tr><td>false</td><td style="text-align: right;">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.									
-		(2) Returns $y - x$ for compatible numeric objects. <sup>25</sup>									
$-\infty$		(-1) {} $\rightarrow$ {2} Minus infinity. See p. 146.									
$\times$		(2) Returns $y \times x$ for compatible numeric objects. <sup>25</sup>									
$\times\text{MOD}$	 	(3) {1, 2, 10} Returns $(z \times y) \bmod x$ for $x > 1, y > 0, z > 0$ . <sup>26</sup>									
/		(2) Like $\times$ , but returns $y \times x^{-1}$ . Returns $y \div x$ if both $y$ and $x$ are of data type 1 or 10; cf. IDIV.									
$\wedge\text{MOD}$	 	(3) {1, 2, 10} Returns $(z^y) \bmod x$ for $x > 1, y > 0, z > 0$ . <sup>26</sup>									
$\rightarrow$		Reserved symbol for indirect addressing.									
$\rightarrow\text{DATE}$	 	(3) {1, 2} $\rightarrow$ {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 date in $x$ . Thus inverts DATE $\rightarrow$ .									

<sup>26</sup> See MOD.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
→DEG		
→D.MS		(1) {1, 2, 4} → {4} Convert angles as described on pp. 154f.
→GRAD		
→HR	 FCNS →HR	(1) {5} → {2} Operates on <i>times</i> like →REAL below. Maintained for backward compatibility only.
→H.MS	 (for closed input)	(1) {1, 2, 5} → {5} Converts $x$ to a sexagesimal <i>time</i> – cf. p. 119.
→INT	 (for closed input)	(1) {1, 2, 10} → {10} Converts the integer part of $x$ to a <i>short integer</i> of the base specified. Conversion to decimal may be abbreviated by , to hexadecimal by . Cf. p. 119.
→MULπ		(1) {1, 2, 4} → {4} Converts angles as described on pp. 154f.
→POL		(2); (1) → {2} 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 ( $r, \theta$ ). Reverted by →REC, see there.   For switching the display format of <i>complex</i> numbers, see RECTN .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
→RAD		(1) {1, 2, 4} → {4} Converts angles as described on pp. 154f
→REAL	 (for closed input)	(1) {1, 2, 4, 5, 6, 10} → {2} Converts $x$ to a <i>real</i> number. Any object (e.g. a <i>time</i> ) tagged sexagesimal will be converted in a decimal number. For <i>dates</i> , the date format chosen is taken into account. Numbers shown as fractions will be displayed as decimal numbers (cf. FRACT and PROPF.R). To return the <i>real</i> part of a <i>complex</i> number, take RE. To cut a <i>complex</i> number into its parts, use CC or CX→RE.
→REC		{2}; {1, 4} → {2} Assumes X and Y containing 2D polar coordinates of a point or components of a vector $(r, \theta)$ . Converts them to the respective Cartesian coordinates or components $(x, y)$ . Inverted by →POL. – For switching the format of <i>complex</i> numbers, see RECTN.
⤵		Shuffles the contents of the <i>stack registers</i> X, Y, Z, and T at execution time.  <b>Examples:</b> ⤵xyxz works like ENTER↑ (but does <u>not</u> disable <i>automatic stack lift!</i> ), ⤵yxzt works like x⤵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.  <b>ATTENTION:</b> This is a very powerful command although it does not look it. Note it will affect the <u>bottom four stack registers only</u> ; there is no connection to A ... D, <u>regardless of stack size</u> . Playing with ⤵, you may lose some <i>stack</i> contents and make a mess of the <i>stack</i> easily.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
M	f MATX  M	(1) {8} → {2}; {9} → {3} Requires a square matrix in X and returns its determinant. The original matrix is stored in L.
x	f  x  or f PARTS f  x	(1) {1, 2, 4, 10} Returns the absolute (unsigned) value of x.  (1) {8*} Returns a matrix with the absolute values of all input matrix elements. Cf. ENORM.
	f  x  or g CPX f  x	(1) {3} → {2} Returns the <i>magnitude</i> $\sqrt{\operatorname{Re}(x)^2 + \operatorname{Im}(x)^2}$ in X.  (1) {9*} → {8} Returns a <i>real</i> matrix with the <i>magnitudes</i> of all input matrix elements. Cf. ENORM.
	g X.FN ▲ f	(2) {2, 3}; {1} → {2} Returns $\left(\frac{1}{x} + \frac{1}{y}\right)^{-1}$ ; useful in electrical engineering especially. Returns 0. for $x \times y = 0$ .
%	g FIN %	(1) {2}; {1} → {2} Returns $\frac{xy}{100}$ , leaving y unchanged.
%MRR	g FIN %MRR	(3) {2}; {1} → {2} Returns the mean rate of return in % 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.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
%T	<b>g FIN %T</b>	(1) {2}; {1} → {2} Returns $\frac{100x}{y}$ , interpreted as % of total. Leaves y unchanged.
%Σ	<b>g FIN %Σ</b>	(1) {2}; {1} → {2} Returns $\frac{100x}{\sum x}$ .
%+MG	<b>g FIN %+MG</b>	(2) {2}; {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.
$\sqrt{x}$	<b>g <math>\sqrt{x}</math></b>	(1) {1, 2, 3, 8*, 9*, 10}; ({1} → {2}, {1, 2} → {3}) Returns the square root of x. Square roots of non-square long integers will return reals. Square roots of negative long integers or reals will return complex numbers if CPXRES is set.
	<b>g EXP <math>\sqrt{x}</math></b>	
$\infty$	<b>f CONST <math>\infty</math></b>	(-1) {} → {2} Infinity. See p. 146.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\int$	<b>f ADV</b> <b>∫fdx</b> <b>∫ var</b> (listed in programs as <b>∫fd</b> trailed by the integration variable)	[2] 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 ↓Lim and ↑Lim, accuracy by ACC. $\int$ returns the (approximated) integral in X and an upper limit of its uncertainty in Y. <sup>27</sup> <b>ATTENTION:</b> $\int$ fills all stack registers with x before calling the routine specified in PGMINT.
	<b>g EQN</b> <b>∫f</b> <b>∫</b>	Integrates the current equation. <sup>27</sup>
$\int f$	<b>g EQN</b> <b>∫f</b>	
$\int f dx$	<b>f ADV</b> <b>∫fdx</b>	Submenus. See pp. 130f.
$\arg$	<b>g ↗</b> or <b>g CPX</b> <b>f ↗</b> or <b>f PARTS</b> <b>f ↗</b>	(1) {3} → {4} Returns the phase or argument $\arg(x) = \arctan\left(\frac{\text{Im}(x)}{\text{Re}(x)}\right)$ . Cf.  x . (1) {9*} → {8} Returns a matrix with the phases of all input matrix elements. Cf.  x .
$\arg \rightarrow$	<b>g L→</b>	Menu of angular conversions. See p. 138.
<b>PRINT</b>	<b>g PRINT</b> <b>ADV</b>	(0) Prints the current contents of the print buffer and a linefeed. <b>ATTENTION:</b> The printer will actually print only when a linefeed is sent to it.

<sup>27</sup> See Section 4 of the OM for more.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
CHAR	<b>g</b> PRINT <b>f</b> CHAR <i>n</i>	(0) Sends a single character (with the code specified) to the printer. Character codes <i>n</i> > 127 can only be specified indirectly. MODE setting will be honored. See ADV.
DLAY	<b>g</b> PRINT <b>g</b> DLAY <i>n</i>	(0) Sets a delay of <i>n</i> ticks (see TICKS) to be used with each linefeed on the printer.
LCD	<b>g</b> PRINT LCD	(0) Sends the contents of the entire LCD to the printer, so you get a hardcopy of the screen.
MODE	<b>g</b> PRINT <b>g</b> MODE <i>n</i>	(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 line. 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	<b>g</b> PRINT PROG	(0) Prints the listing of the <i>current program</i> (see Sect. 3 of the OM), one row per step.
r	<b>g</b> PRINT r <i>r</i>	(0) Prints the <i>register</i> specified, right adjusted, without labeling the output (note r X takes a separate label x in this menu). If you want a heading label, compose the string in X first or use REGS. See ADV.

Item	Keystrokes	Remarks (see pp. 13ff 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 its label.  ATTENTION for $nn = 0$ : <ul style="list-style-type: none"><li>• For <math>sss \in [0; 99]</math>, printing will stop at <b>R99</b>.</li><li>• For <math>sss \in [100; 111]</math>, printing stops at <b>K</b>.</li></ul> For $sss \geq 112$ , printing stops at the highest allocated local register.
■STK		(0) Prints the entire stack contents. Each of the 4 or 8 registers prints in a separate row starting with its label.
■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 ■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 ■ADV.
■USER		(0) Prints all variable names and global program labels in alphabetic order. The variable names are printed first; if you are not interested in the program labels, press <b>R/S</b> to stop the listing.
■WIDTH		(-1) Returns the number of print columns that $x$ would take in the print mode set. See ■MODE.  Second use: in ■MODE 1 or 2, ■WIDTH returns the width of $x$ in px (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.

## **Names of Variables and System Flags Provided**

There is a *name* overlap between some constants and commands on one side and predefined variables and *system flags* on the other. Thus, the latter set is kept separate from the other *items*. As required for the *items* above, these *names* must be unique.

The current status of *items* printed on grey in the table below can be seen on the screen directly (e.g. in the *status bar*), for those printed on orange it is returned by STATUS. If the second column is green, the corresponding *item* may be written by the user, if it is yellow, the *item* is read-only for the user and is written by the system exclusively.

Some system flags can be accessed via lettered user flags as well (cf. Section 1 of the OM).

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
A		Reserved variable for <i>register A</i>
ACC	f ADV ∫fdx ACC	Reserved <i>real</i> variable for the accuracy of integration (see Sect. 4 of the OM).

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
ADM		Reserved integer variable for the ADM: 0: DEG, 1: D.MS, 2: RAD, 3: MUL $\pi$ , 4: GRAD.
ALLSCI	[A]	
ALPHA		
ALP.IN		
ASLIFT		
AUTOFF		
AUTXEQ		
B		Reserved variables for registers <b>B</b> and <b>C</b> .
C		
CARRY	[C]	
CPXj		System flags – see next chapter.
CPXRES	[I] (imaginary)	
D		Reserved variable for register <b>D</b> .
DECIM.		
DENANY		System flags – see next chapter.
DENFIX		
DENMAX		Reserved integer variable for the maximum denominator, filled by the command DENMAX.
DMY		
FRACT		System flags – see next chapter.

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
FV	<b>g [FIN] TVM FV</b>	Reserved real variable for the future value of your investment or loan in <i>TVM</i> . <sup>28</sup>
GRAMOD		Reserved integer variable determining how the AGRAPH image is displayed: <sup>29</sup> 0: It is merged (OR-ed) with the existing display. 1: It overwrites all pixels in that part of the display. 2: Duplicate “on” pixels get turned “off”. 3: It is XOR-ed with the existing display.
GROW		<i>System flag</i> – see next chapter.
I		Reserved variable for <i>register I</i> .
IGN1ER		
INTING		<i>System flags</i> – see next chapter.
i%/a	<b>g [FIN] TVM i%/a</b>	Reserved real variable for the annual interest rate of your investment or loan in <i>TVM</i> . <sup>28</sup>
J		
K		Reserved variables for <i>registers J, K, and L</i> .
L		
LEAD.0	<b>[L]</b>	
LOWBAT		<i>System flags</i> – see next chapter.

<sup>28</sup> See Section 5 of the OM.

<sup>29</sup> Working like flags 34 and 35 in HP-42S.

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
Mat_A	f [MATX] SIM EQ Mat A	
Mat_B	f [MATX] SIM EQ Mat B	
Mat_X	f [MATX] SIM EQ Mat X	Reserved variables for solving systems of linear equations (see Sect. 2 of the OM).
MDY		
MULTx		System flags – see next chapter.
NPER	g [FIN] TVM n <sub>PER</sub>	Reserved variable for the <u>total</u> number of <ul style="list-style-type: none"> <li>payment periods for your loan or</li> <li>compounding periods for your investment.</li> </ul>
NUM.IN		
OVERFL	B (big)	System flags – see next chapter.
PER/a	g [FIN] TVM f per/a	Reserved variable for the <u>annual</u> number of <ul style="list-style-type: none"> <li>payments for your loan or</li> <li>compounding periods of your investment.</li> </ul>
PMT	g [FIN] TVM PMT	Reserved variable for the payment per period for your investment or loan in <i>TVM</i> . <sup>28</sup>
PRINT		
PROPFRR		System flags – see next chapter.
PRTACT		
PV	g [FIN] TVM PV	Reserved variable for the present value of your investment or loan in <i>TVM</i> . <sup>28</sup>
QUIET		System flag – see next chapter.
REALDF		Reserved integer variable for real number display format: 0: ALL, 1: FIX, 2: SCI, 3: ENG.

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
RECTN	<input checked="" type="checkbox"/>	<i>System flag</i> – see next chapter.
REGS		Reserved variable for the 100×1 matrix of registers – if required.
RUNIO		
RUNTIM		<i>System flags</i> – see next chapter.
SLOW		
SOLVING		
SPCRES	<input type="checkbox"/> (danger)	<i>System flags</i> – see next chapter.
SSIZE8		
ST.A		
ST.B		
ST.C		
ST.D		
ST.T		
ST.X		
ST.Y		
ST.Z		
TDM24		
TRACE	<input type="checkbox"/>	<i>System flags</i> – see next chapter.
USER	<input checked="" type="checkbox"/> <b>USER</b>	<i>System flag</i> toggled by <b>USER</b> – see next chapter
VMDISP		
YMD		<i>System flags</i> – see next chapter.
αCAP		

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
$\text{\texttt{tLim}}$	<b>f ADV Jfdx</b> $\text{\texttt{tLim}}$	Reserved <i>real</i> variables for the upper and lower limit of integration (s. the OM, Sect. 4).
$\text{\texttt{vLim}}$	etc.	
<b>#DEC</b>		Reserved integer variable for the number of decimals in <i>real</i> number formatting (actually the parameter specified in last ALL, FIX, SCI, or ENG).

## System Flags

The *status bar*, especially its indicators (*SBI*), and the command STATUS return visible information about the system status of your WP 43S (cf. Sections 2 and 5 of the OM). Machine readable status information (e.g. for program control) is available via the 40 named *system flags* as listed below, sorted following the *status bar*.

Purpose	<i>SBI</i>	Flag <sup>30</sup>	Remarks
Time display	Time string	TDM24 (set)	Set for international 24h time display. Expands the date display in the <i>status bar</i> to four digits for the year.  Clear for 12h time display: 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.
Date display	Date string	YMD, DMY, MDY (YMD set)	Set for the respective date format chosen. The commands Y.MD, D.MY, and M.DY set the corresponding flag and clear the two others.

<sup>30</sup> The flags printed on green may be written by the user, those printed on yellow are written by the system exclusively. Those printed on light blue will be set as recalled after RCLCFG. Startup default status is given in parentheses for the flags set at startup – all others are clear at startup. Grey and orange colors are used as in previous chapter.

Purpose	SBI	Flag <sup>30</sup>	Remarks
Complex results	C / R	CPXRES	Set for allowing <i>complex</i> results also for <i>real</i> /input (e.g. $\sqrt{-1}$ ). Else an error will be thrown in such cases.
Complex letter	—	CPXj	Set for the letter <i>j</i> representing the <i>imaginary</i> number <i>i</i> , clear for <i>i</i> .
Rectangular notation	L / ⊙	RECTN (set)	Set for rectangular display of <i>complex</i> numbers, clear for polar.
Fraction display	—	FRACT	Set if fraction display is chosen, clear for decimal display ( <b>a b/c</b> sets FRACT, <b>.d</b> clears it). See next entries.
Fraction kind 1	—	PROPFRACTION	Set for <i>proper fractions</i> , clear for <i>improper fractions</i> ( <b>a b/c</b> toggles PROPFRACTION: coming from decimal display, it sets it). PROPFRACTION set allows only <i>proper fractions</i> in display (e.g. $1 \frac{2}{3}$ instead of $\frac{5}{3}$ ); any <i>reals</i> (with $ x  < 10^6$ ) will be displayed according to the settings of DENANY, DENFIX, and DENMAX as <i>proper fractions</i> .
Fraction kind 2	see OM	DENANY (set)	Set if <u>any</u> denominator up to DENMAX may appear (DENMAX is indicated in the <i>status bar</i> ). <sup>31</sup> For given DENMAX, this is the most precise way of displaying a decimal number as a fraction. See also next entry.
Fraction kind 3	see OM	DENFIX <sup>32</sup>	Set if the value set by DENMAX is the one and only denominator allowed. Clear if the denominator may be an integer factor of DENMAX. <sup>33</sup>

<sup>31</sup> E.g. for DENMAX = 5 and DENANY set, denominators 1, 2, 3, 4, and 5 are allowed.

<sup>32</sup> DENFIX is evaluated only if DENANY is clear.

<sup>33</sup> If DENMAX = 60 and DENFIX is clear, this will allow for denominators 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, and 60 (note 60 was a holy number in ancient Babylon).

Purpose	<i>SBI</i>	<i>Flag</i> <sup>30</sup>	Remarks
Carry	$c$ or $\bar{c}$	CARRY	Reflects the status of the carry bit.
Overflow	$o$ or $\bar{o}$	OVERFL	Reflects the status of the overflow bit.
Leading zeros	—	LEAD.0	Set for leading zeros turned on in <i>short integers</i> of bases 2, 4, 8, and 16. <sup>34</sup>
Alpha	A or $\alpha$	ALPHA	Set for A/M, else clear.
Upper case	A / $\alpha$	ACAP (set)	Set for capital letters, clear for lower case.
Running timer	⌚	RUNTIM	Set if the timer is running.
Running I/O	⬇️	RUNIO	Set if I/O is in progress.
Printing	🖨️	PRINT	Set if your WP 43S is sending data to the printer.
Tracing	—	TRACE	Set if the print line is in tracing mode. Else prints must be triggered explicitly.
User mode	👤	USER	Set if your WP 43S is in user mode.
Low battery	🔋	LOWBAT	Set if the battery voltage is low (see Section 2 of the OM).
Processor speed	—	SLOW	Kept clear for fresh batteries unless the user intervenes, set for battery voltage < 2.5V. Speed and power consumption will be reduced to ~50% with SLOW set.
Special results	—	SPCRES	Set for allowing special results of calculations (i.e. $\pm\infty$ and NaN).

<sup>34</sup> Works like flag 3 in HP-16C.

Purpose	SBI	Flag <sup>30</sup>	Remarks
Stack size	—	SSIZE8	Set for eight, clear for four <i>stack registers</i> . Note all <i>register</i> contents will remain unchanged if SSIZE8 is modified (as well as if <i>stack</i> size is changed by any other operation – e.g. by RCLCFG).
Beeper	—	QUIET	Set for disabled beeper.
Radix mark	—	DECIM. (set)	Set for a decimal point, clear for a comma.
Multiplication symbol	—	MULTx (set)	Set for multiplication symbol ×, clear for ·.
Overflow from ALL	—	ALLSCI (set)	Set if numbers exceeding the range displayable in ALL or FIX will be shown in SClientific, clear for ENGineer's format.
Matrix grow mode	—	GROW	Set (e.g. by M.GROW) if matrices may grow, else clear (e.g. by M.WRAP). See the command J+ above and <i>Section 2</i> of the OM; see also GROW in the <i>HP-42S Owner's Manual</i> , p. 213.
Automatic OFF	—	AUTOFF (set)	Set if automatic shutdown is enabled. Else your WP 34S will remain ON until you will turn it off manually or battery voltage will drop below the limit.
Automatic execution	—	AUTXEQ	Like flag 11 of HP-42S.
Printer activated	—	PRTACT	Like flag 21 (Print Enable) of HP-42S.
Data entry	—	NUM.IN, ALP.IN	Set for numeric or alphanumeric entry (like flags 22 and 23 of HP-42S).

Purpose	SBI	Flag <sup>30</sup>	Remarks
Enable automatic stack lift	—	ASLIFT (set)	Cleared by ENTER, CLX, $\Sigma+$ , and $\Sigma-$ , set by all other functions (see the OM, Section 1)
Error handling	—	IGN1ER	If set, your WP 43S ignores just 1 arbitrary error and clears IGN1ER then. The operation causing the error will not be executed. This works like flag 25 of HP-42S
Integrating	—	INTING	Set while the integrator is running.
Solving	—	SOLVING	Set while the solver is computing a root.
Variable menu	—	VMDISP	Set while the variable menu is displayed.

## Nonprogrammable Commands and Keys

The commands marked *violet* in the *IOI* cannot be programmed. The same applies to all 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**,  **$\alpha$** ,  **$\wedge$** ,  **$\equiv\Delta$** ,  **$\blacktriangleleft$** ,  **$\equiv\nabla$** , and  **$\blacktriangleright$**  in *startup default* condition 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 (on pp. 123ff) for more about this topic.

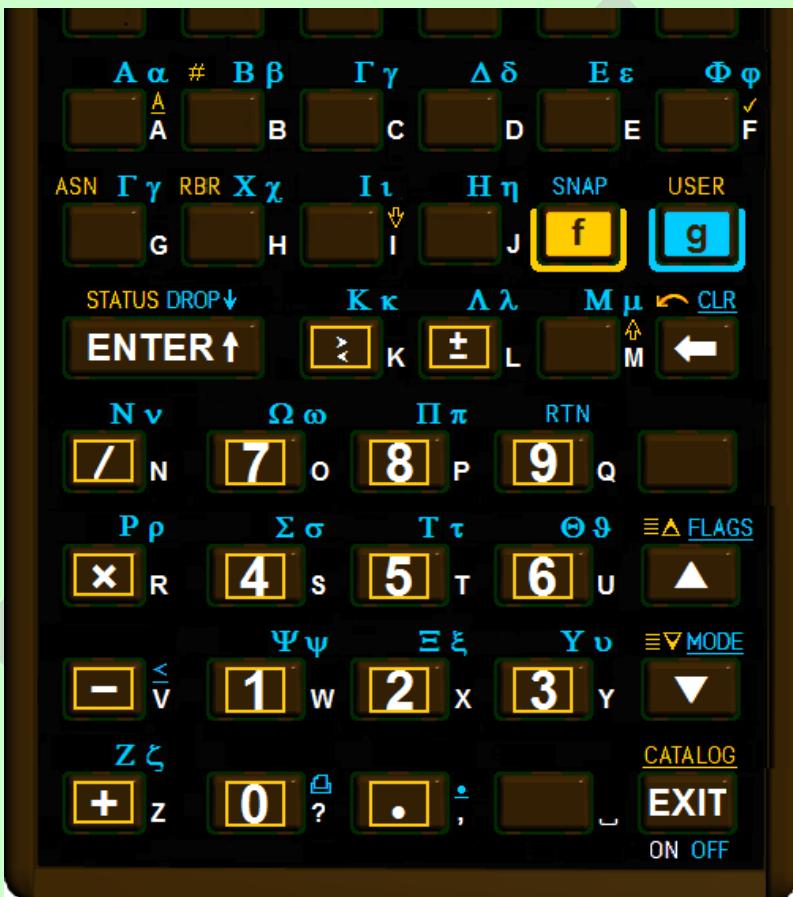
The *browsers* RBR and STATUS as well as the *application* TIMER use some keys for particular control purposes (e.g. **STO**, **RCL**,  **$\square$** , and numeric keys – see the OM, Section 5).

## 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. 119ff for input in **X** instead). Note that a “character” may be a letter, digit, punctuation mark, etc. The table below lists the respective keys beginning top left on the keyboard:

Keystrokes	Situation	Meaning
<b>[A ... D], [I ... L, T], [X ... Z]</b>	addressing	Enters the address of a <i>global general purpose register</i> or <i>user flag</i> .
<b>[A ... Z]</b> <b>[g A ... g O]</b> <b>[f O ... f 9]</b>	entering a label, a <i>system flag</i> or variable <i>name</i>	Appends the corresponding Latin or Greek letter or digit to the label, <i>system flag</i> or variable <i>name</i> pending. Use <b>▼</b> and <b>▲</b> to switch cases for letters. See the virtual keyboard on p. 118 (cf. the OM, Sect. 2).
<b>[ENTER↑]</b>	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>user flag</i> address, a <i>system flag</i> or variable <i>name</i> , or a label or alike, and executes the command. Cf. Section 1 of the OM.
<b>[←]</b>	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.
<b>[0 ... 9]</b>	addressing or specifying	Enters a numeric parameter, an address, or a local label. See Sections 1 and 3 of the OM for valid number ranges.
<b>[.]</b>	addressing	Header for <i>local registers</i> or <i>user flags</i> .

Keystrokes	Situation	Meaning
<b>EXIT</b>	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 <i>WP 43S</i> as it was before the current command was called.



Virtual keyboard in *alpha input mode (AIM)*. AIM is also active when a catalog is entered, so you can use all accessible characters for alphabetic searching (see pp. 127f). Note there is an ‘alpha helper’ printed on the back of your *WP 43S*.

## Alphanumeric Input in X and Closing It

The following table shows what will happen when particular keys are pressed with alphanumeric (incl. numeric) input in X being open still (turn to pp. 117f for command parameter input instead). The table lists the respective keys top left to bottom right on the keyboard:

Keystrokes in mode(s)		Meaning
[A] ... [Z]	A, $\alpha$	Appends the corresponding Latin or Greek letter to the <i>alpha string</i> $x$ . Use $\blacktriangledown$ and $\blacktriangleup$ to switch cases. See the picture on previous page and cf. <i>Section 2</i> of the OM.
[g] [A] ... [g] [O]		
[f] [#]	A, $\alpha$	Appends # to the <i>alpha string</i> $x$ .
[f] [#] base	$\neg$ (A, $\alpha$ )	Closes input of a <i>short integer</i>
[f] [d.ms]		sexagesimal angle
[f] [.d]		date
[f] [h.ms]		sexagesimal <i>time</i> in X. <sup>35</sup>
[f] [ $x^2$ ] ( ✓ )	A, $\alpha$	Appends a checkmark ✓ to the <i>alpha string</i> $x$ .
[CC]	$\neg$ (A, $\alpha$ )	Closes input of the first part (i.e. <i>real part</i> or <i>magnitude</i> ) of a <i>complex number</i> in X and waits for input of its second part (i.e. <i>imaginary part</i> or <i>phase</i> , see the <i>Key Response Table</i> and <i>Section 2</i> of the OM).
[f] [R↓] ( ⌄ )		A, $\alpha$
		Prefix making the next character a subscript, if applicable.

<sup>35</sup> See *Section 2* of the OM. At closure, input will be checked – illegal digits (e.g. 8 in octal input or C in decimal), bases, 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)	Meaning
<b>ENTER↑</b>	<p>arbitrary input pending</p> <p>If there was input expected but not entered, cancels entry.</p> <p>Else closes input (in <b>X</b>) and checks the following conditions top-down:</p> <ul style="list-style-type: none"> <li>• If this input is <u>alphanumeric</u> (i.e. if it contains at least one non-numeric character except <b>.</b>), takes it as an <i>alpha string</i>.</li> <li>• Else (i.e. if this input is purely numeric) if it contains one <b>CC</b>, takes it as a <i>complex number</i>.</li> <li>• Else if it contains two <b>,</b>, takes it as a <i>fraction</i>.</li> <li>• Else if it contains one <b>,</b> or one <b>E</b>, takes it as a <i>real number</i>.</li> <li>• Else (i.e. if it contains neither a <b>CC</b> nor a <b>,</b> nor an <b>E</b>), tests it for <b>#</b>: <ul style="list-style-type: none"> <li>◦ If it contains one <b>#</b> and a valid base trailing it then takes it as a <i>short integer</i>;</li> <li>◦ 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>.</li> </ul> </li> </ul> <p>Then checks the new input (according to the condition met) as outlined in footnote 35 and interprets it. Finally, unless an error had to be thrown, copies <i>x</i> into <b>Y</b>.</p>
<b>f x&gt;y</b>	<b>A, α</b> Appends <b>&gt;</b> to the <i>alpha string</i> <i>x</i> .
<b>+/</b>	$\neg(A, \alpha)$ Changes the sign of the number entered. Affects the exponent if pressed after <b>E</b> .
<b>f +/-</b>	<b>A, α</b> Appends <b>±</b> to the <i>alpha string</i> <i>x</i> .
<b>E</b>	$\neg(A, \alpha)$ Closes input of the mantissa and waits for input of the exponent (see <i>Section 1</i> of the OM).
<b>f E ( ↑ )</b>	<b>A, α</b> Prefix making the next character a superscript, if applicable.

Keystrokes	in mode(s)	Meaning
	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, $\alpha$	Appends  to the <i>alpha string</i> $x$ .
	A, $\alpha$	If MULT $x$ then appends , else  to the <i>string</i> $x$ .
	A, $\alpha$	Appends  to the <i>alpha string</i> $x$ .
	A, $\alpha$	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).
	$\neg$ (A, $\alpha$ )	Standard numeric input, appending the corresponding digit to $x$ . Note you can enter ... <ul style="list-style-type: none"> <li>• up to 16 digits plus a sign in the mantissa and up to three digits plus a sign in the exponent for a <i>real number</i> or any part of a <i>complex number</i>,</li> <li>• an arbitrary number of digits plus a sign for a <i>long integer</i>,</li> <li>• up to 64 bits for a <i>short integer</i>, or</li> <li>• up to 16 digits for the nominator and up to 4 digits for the denominator of a fraction.</li> </ul>
	A, $\alpha$	Appends  to the <i>alpha string</i> $x$ .
	A, $\alpha$	Appends the respective digit to the <i>alpha string</i> $x$ .
	A, $\alpha$	Appends  to the <i>alpha string</i> $x$ .
	$\alpha$	Turns to upper case for the following letters.
	A	Turns to lower case for the following letters.

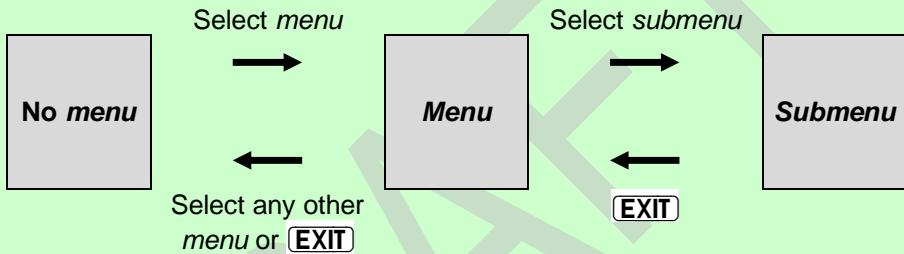
Keystrokes in mode(s)		Meaning
.	¬ (A, α)	Inserts a radix mark as selected.
		Separates <i>degrees</i> from <i>minutes</i> , <i>seconds</i> , and <i>hundredths of seconds</i> in angular input, so input format is dddd.dddd.mmmsshh <b>d.ms</b> for sexagesimal angles (cf. p. 119 and Section 2 of the OM).
		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.ffff.mmmssffff <b>h.ms</b> for sexagesimal times (cf. p. 119 and Section 2 of the OM).
Second .	¬ (A, α)	A 2 <sup>nd</sup> . in input indicates a fraction. See the OM, Sect. 2 for examples. The 2 <sup>nd</sup> . just separates the nominator and the denominator in input. Note you cannot enter E after you entered . twice – but you may delete the 2 <sup>nd</sup> dot while editing the input.
.	A, α	Appends , to the alpha string x.
f .	A, α	Appends . to the alpha string x.
R/S	!, program waits f. input	Closes input and starts its checks and interpretation like <b>ENTER↑</b> above. Resumes program execution.
	A, α	Appends a blank space to the alpha string x.
EXIT	arbitrary input pending	If there is an open menu, 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 p. 124 and the links printed there for these menus. Call FBR for browsing 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, variables, program labels).

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



**Catalogs** are a special kind of *menus* with their contents sorted alphabetically. Your *WP 43S* provides the following 17 *catalogs*:

- CONST,
- CATALOG'FCNS (the greatest *catalog* by far at startup),
- CATALOG'MENUS,
- CATALOG'DIGITS,
- CATALOG'SYS.FL,
- CATALOG'CHARS'aINTL,
- the nine *submenus* of CATALOG'VARS and
- the two *submenus* of CATALOG'PROGS .

Within *catalogs*, some special operations ease your path accessing the *items* stored therein (as shown on pp. 127f).

## One to Find and 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. 127f.

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

							Remarks
<b>CATALOG:</b>	FCNS	SYS.FL	CHARS	PROGS	VARS	MENUS	top branches
		REGIST	DIGITS				
FCNS:	$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$	$10^x$	1COMPL	$1/x$	$2^x$	contains some 700 functions provided
	2COMPL	$\sqrt[3]{\text{x}}$	ABS	ACOS	$\text{ac} \rightarrow \text{m}^2$	$\text{ac}_{\text{us}} \rightarrow \text{m}^2$	
	AGM	AGRAPH	ALL	AND	arccos	arcosh	
	...						
	...	#B					
SYS.FL:	ALLSCI	ALPHA	ALP.IN	...			system flags (cf. pp. 107ff)
	...						
CHARS:	$\alpha\text{INTL}$	$\alpha\ldots\Omega$		$\alpha\text{MATH}$	$\text{My}\alpha$	$\alpha\cdot$	character branches
$\alpha\text{INTL}:$	A	$\bar{A}$	$\breve{A}$	$\check{A}$	$\grave{A}$	...	international Latin letters, see p. 136
$\alpha\text{MATH}:$	<	$\leq$	=	...			mathematical operators and symbols, see p. 137
$\alpha\ldots\Omega:$	A	B	$\Gamma$	$\Delta$	...		Greek letters, see p. 137
$\alpha\bullet:$	!	;	...				punctuation marks, see p. 138
PROGS:	RAM					FLASH	global labels currently defined

	<input type="checkbox"/>	Remarks					
RAM:	...						both branches are empty at startup; they will be filled with your creations
FLASH:	...						
VARS:	L.INTS DATES	S.INTS TIMES	REALS ANGLES	CPXS 	STRING 	MATRS 	branches for various types of variables
ANGLES:	...						
CPXS:	...						
DATES:	...						
L.INTS:	ADM	DENMAX	GRAMOD	REALDF	#DEC	...	
MATRS:	Mat_A	Mat_B	Mat_X	REGS	...		
REALS:	ACC PV	FV ↑Lim	i%/a ↓Lim	nPER ...	PER/a	PMT	
STRING:	...						
S.INTS	...						
TIMES:	...						
MENUS:	ANGLES CLK DIGITS	A: CLR DISP	BITS CONST EQN	Binom: CPX EXP	Cauch: CPXS Expon:	CHARS DATES E:	menus and sub-menus currently defined (shown here at startup, but also this list will grow with your creations) – see above and below for fix menu contents
	...						
	...	→					
A:	...						(sub-) menus provided unless mentioned above already, see pp. 129ff for predefined contents – here your creations will be inserted as new entries
Binom:	...						
BITS:	...						
...							
...							
REGIST:	A K ST.T	B L ST.X	C ST.A ST.Y	D ST.B ST.Z	I ST.C	J ST.D	register names defined
DIGITS:	0 6 C	1 7 D	2 8 E	3 9 F	4 A i	5 B 	digits defined

Three branches of **CATALOG** are expandable (**MENUS** and the submenus of **PROGS** and **VARS**) since you may create *items* of these kinds (cf. the OM, Sect. 6); the other ones are fixed size (**FCNS**, **REGIST**, **SYS.FL**, **DIGITS**, and **CHARS**) since all functions, *register names*, system flags, digits, and characters on your *WP 43S* are predefined.

Calling CATALOG will display its top level branches. The seven labels shown are pointers to the sub-

							0.004	
	REGIST	DIGITS						

*menus* containing all the functions, *register names*, *system flags*, digits, characters, programs, variables, and *menus* defined at execution time.

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

	AGM	AGRAPH	ALL	AND	arccos	arcosh	U.UU7
	2COMPL	$\sqrt[3]{x}$	ABS	ACOS	$ac \rightarrow m^2$	$ac_{us} \rightarrow m^2$	
	$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$	$10^x$	1COMPL	$1/x$	$2^x$	

Select an *item* by pressing the corresponding *softkey* (headed by **f** or **g** if applicable); e.g.

- call any function via CATALOG'FCNS,
  - call any *menu* via CATALOG'MENUS,
  - test any system *flag* via CATALOG'SYS.FL, or
  - recall any real variable defined via CATALOG'VARS'REALS.

Within **CATALOG** branches, browsing by **▲** will advance by six *items* per keystroke (and **▼** will go back by six) only.<sup>36</sup> **EXIT** will just leave the open branch without doing anything.

<sup>36</sup> Navigating in catalogs, 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.

You will find all the over 700 functions available on your *WP 43S* stored in CATALOG'FCNS (most of them may be accessed easier through other *menus* though, see the chapter after next chapter). Remember that each and every command, constant, and predefined *menu* featured on your *WP 43S*, the keystrokes calling it, and the necessary particular explanations are listed for your reference in the *IOI* on pp. 13ff. Find the other predefined *items* provided (*system flags* and variable *names*) on pp. 107ff.

See the OM, Section 6, 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 also assign your favorite functions to almost any location on the keyboard. Actually, you can design your very own *WP 43S* user interface.

## Accessing Cataloged Items Rapidly

You can browse a *catalog* like any other *menu* just using **▲** and **▼** as explained in previous chapter. In CONST and major parts of CATALOG (**FCNS**, **SYS.FL**, **MENUS**, **REGIST**, **DIGITS**, **CHARS**  $\alpha$ **INTL**, and the submenus of **PROGS** and **VARS**), you may reach your target significantly faster taking advantage of the alphabetic access method demonstrated here. Assume we are looking for the function FS?S, for **example**:

1 User input

Return

**CATALOG** **FCNS**

Your *WP 43S* displays the first view  
in this catalog;<sup>37</sup> AIM is on.

AGM	AGRAPH	ALL	AND	arccos	arcosh
2COMPL	$\sqrt[3]{x}$	ABS	ACOS	$ac \rightarrow m^2$	$ac_{us} \rightarrow m^2$
$^{\circ}C \rightarrow ^{\circ}F$	$^{\circ}F \rightarrow ^{\circ}C$	$10^x$	1COMPL	$1/x$	$2^x$

<sup>37</sup> ... 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.

2 User input	First character of the <i>item</i> desired (e.g. <b>F</b> )																		
Return	<b>Your WP 43S displays a view starting with the first item starting with this character</b> <sup>38</sup> e.g.																		
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">FLOOR</td><td style="padding: 2px;">FP</td><td style="padding: 2px;">FP?</td><td style="padding: 2px;"><math>F_p(x)</math></td><td style="padding: 2px;">FS?</td><td style="padding: 2px;">FS?C</td></tr> <tr> <td style="padding: 2px;"><math>F_e(x)</math></td><td style="padding: 2px;">FF</td><td style="padding: 2px;">FIB</td><td style="padding: 2px;">FILL</td><td style="padding: 2px;">FIX</td><td style="padding: 2px;">FLASH?</td></tr> <tr> <td style="padding: 2px;">FB</td><td style="padding: 2px;">FBR</td><td style="padding: 2px;">FC?</td><td style="padding: 2px;">FC?C</td><td style="padding: 2px;">FC?F</td><td style="padding: 2px;">FC?S</td></tr> </table>	FLOOR	FP	FP?	$F_p(x)$	FS?	FS?C	$F_e(x)$	FF	FIB	FILL	FIX	FLASH?	FB	FBR	FC?	FC?C	FC?F	FC?S
FLOOR	FP	FP?	$F_p(x)$	FS?	FS?C														
$F_e(x)$	FF	FIB	FILL	FIX	FLASH?														
FB	FBR	FC?	FC?C	FC?F	FC?S														
3 User input	Second character of the <i>item</i> desired (e.g. <b>S</b> )																		
Return	<b>Your WP 43S displays a view starting with the first item starting with the string you specified</b> e.g.																		
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;"><math>fz_{us} \rightarrow m^3</math></td><td style="padding: 2px;"><math>f'(x)</math></td><td style="padding: 2px;"><math>f''(x)</math></td><td style="padding: 2px;">GAP</td><td style="padding: 2px;">GaussF</td><td style="padding: 2px;">GCD</td></tr> <tr> <td style="padding: 2px;"><math>fm. \rightarrow m</math></td><td style="padding: 2px;">fr <math>\rightarrow</math> dB</td><td style="padding: 2px;">ft. <math>\rightarrow</math> m</td><td style="padding: 2px;"><math>ft_{us} \rightarrow m</math></td><td style="padding: 2px;">ft. <math>\rightarrow</math> m</td><td style="padding: 2px;"><math>fz_{uk} \rightarrow m^3</math></td></tr> <tr> <td style="padding: 2px;">FS?</td><td style="padding: 2px;">FS?C</td><td style="padding: 2px;">FS?F</td><td style="padding: 2px;">FS?S</td><td style="padding: 2px;"><math>F(x)</math></td><td style="padding: 2px;"><math>F^{-1}(p)</math></td></tr> </table>	$fz_{us} \rightarrow m^3$	$f'(x)$	$f''(x)$	GAP	GaussF	GCD	$fm. \rightarrow m$	fr $\rightarrow$ dB	ft. $\rightarrow$ m	$ft_{us} \rightarrow m$	ft. $\rightarrow$ m	$fz_{uk} \rightarrow m^3$	FS?	FS?C	FS?F	FS?S	$F(x)$	$F^{-1}(p)$
$fz_{us} \rightarrow m^3$	$f'(x)$	$f''(x)$	GAP	GaussF	GCD														
$fm. \rightarrow m$	fr $\rightarrow$ dB	ft. $\rightarrow$ m	$ft_{us} \rightarrow m$	ft. $\rightarrow$ m	$fz_{uk} \rightarrow m^3$														
FS?	FS?C	FS?F	FS?S	$F(x)$	$F^{-1}(p)$														
4 User input	Press the corresponding softkey e.g. for <b>FS?S</b>																		

<sup>38</sup> 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 search for Greek letters via prefix **g**, e.g. **g** + **A** for  $\alpha$  (though watch the sorting order as printed at the beginning of the *IOI*). Also other characters can be specified in a search – please see the *virtual keyboard* printed on p. 110. Note the *items* in the *catalog* you search may be displayed at locations in the *menu section* deviating from the ones you see in simple browsing using **▼** or **▲** only.

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

If a character or string 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.

Return

Your *WP 43S* executes the command, tests the *system flag*, calls the program, recalls the constant or variable, or inserts the command, digit or character selected. *AIM* stays on until this catalog is exited – then your *WP 43S* returns to the mode as set before entering this catalog.

### Result

(in this example after specifying the *flag* number):

true

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 700);
- any constant provided can be recalled by **f CONST** + 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 **f A**) + 3 keystrokes maximum;
- any *system flag* provided can be tested by **f CATALOG SYS.FL** + 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. Individual *items* may appear in more than one *menu* and also on the keyboard.

Menu							Remarks
<b>ADV</b>	SOLVE	SLVQ	$f'(x)$	$\Pi_n$	$\Sigma_n$	$\int f dx$	advanced operations, see Sect. 4 of the OM
	PGMSLV		$f''(x)$			PGMINT	
	$\int f dx$		ACC	$\downarrow\downarrow$ Lim	$\uparrow\uparrow$ Lim	$\int$	
<b>BITS</b>	AND	OR	XOR	NOT	MASKL	MASKR	contains all the Boole's and bit operations (first two views) and settings (third view) of HP-16C and WP 34S
	NAND	NOR	XNOR	MIRROR		ASR	
	SB	BS?	#B	FB	BC?	CB	
	SL	RL	RLC	RRC	RR	SR	
	LJ					RJ	
	1COMPL	2COMPL	UNSIGN	SIGNMT		WSIZE	
<b>CLK</b>	DATE	$\rightarrow$ DATE	DATE $\rightarrow$	WDAY	TIME	$x \rightarrow$ DATE	date and time functions (first view) and settings (second view)
	J $\rightarrow$ D	D $\rightarrow$ J		DAY	MONTH	YEAR	
	SETTIM	TDISP	SETDAT	D.MY	Y.MD	M.DY	
						J/G	
<b>CLR</b>	CL $\Sigma$	CLP	CF	CLMENU	CLCVAR	CLX	almost as in HP-42S
	CLREGS	CLPall	CLFall		CLLCD	CLSTK	
	CLall					RESET	
<b>CONST</b>							<i>catalog</i> of constants, see pp. 138ff
<b>CPX</b>	dot	cross	UNITV	Re	conj	Re $\rightarrow$ Im	special complex functions
	CX $\rightarrow$ RE	RE $\rightarrow$ CX	sign	Im	x	$\neq$	
<b>DISP</b>	FIX	SCI	ENG	ALL	ROUNDI	ROUND	display rounding and shifts, formats and settings, mostly for <i>reals</i>
	SDL	SDR			RDP	RSD	
	CHINA	EUROPE	INDIA	JAPAN	UK	USA	
	GAP		RANGE	RANGE?		DSTACK	

Menu							Remarks
<a href="#">EQN</a>	NEW	EDIT	f''	f'	∫f	Solver	equations (see the OM, Sect. 4)
	DELETE						
<u>Solver</u>							show the names of all variables of the current equation and more
<u>f</u>							
<u>f'</u>							
<u>f''</u>							
<u>EQ.EDI</u>	←	( )	^	:	=	→	Equation Editor
<a href="#">EXP</a>	x <sup>3</sup>	√[y]	log <sub>x</sub> y	lb x	2 <sup>x</sup>	√x	exponential, logarithmic, and hyperbolic functions
	³√x			ln 1+x	e <sup>x-1</sup>		
	sinh	arsinh	cosh	arcosh	tanh	artanh	
<a href="#">FIN</a>	%	%MRR	%T	%Σ	%+MG	TVM	financial functions and settings (see the OM, Section 5)
<u>TVM</u>	n <sub>PER</sub>	i%/a	per/a	PV	PMT	FV	
	Begin					End	
<a href="#">FLAGS</a>	SF	FS?	FF	STATUS	FC?	CF	
	FS?S	FS?C	FS?F	FC?F	FC?S	FC?C	
						CLFall	
<a href="#">INFO</a>	SSIZE?	MEM?	RM?	SMODE?	WSIZE?	KTYP?	system information plus one non-binary test (±∞?)
	Loc?	FLASH?	ULP?	NEIGHB	SDIGS?	BATT?	
	WHO?	VERS?	DIM?	±∞?	αPOS?	αLENG?	
	RANGE?					BestF?	
<a href="#">INTS</a>	A	B	C	D	E	F	digits for short integers with bases > 10, integer operations
	IDIV	RMD	MOD	×MOD	FLOOR	LCM	
	DBL /	DBLR	DBL ×	^MOD	CEIL	GCD	
	1COMPL	2COMPL	UNSIGN	SIGNMT		WSIZE	
<a href="#">I/O</a>	BEEP	LOAD	LOADP	LOADR	LOADSS	LOADΣ	data exchange and signalling
	TONE				RECV	SEND	

Menu							Remarks
<u>LOOP</u>	DSE	DSZ	DSL	ISE	ISZ	ISG	
	DEC					INC	
<u>MATX</u>	NEW	[M] <sup>-1</sup>	M	[M] <sup>T</sup>	SIM EQ	EDIT	matrix operations (the items sorted almost as in HP-42S)
	dot	cross	UNITY	DIM	INDEX	EDITN	
	ENORM		STOEL	RCLEL	PUTM	GETM	
	I+	I-	STOIJ	RCLIJ	J-	J+	
	RSUM	RNORM	M.LU	DIM?		R <sup>Z</sup> R	
	EIGVAL					EIGVEC	
<u>M.EDIT</u>	←	↑	OLD	GOTO	↓	→	Matrix Editor as in HP-42S
	INSR		DELR		WRAP	GROW	
<u>M.SIMQ</u>	Mat A	Mat B				Mat X	solver for systems of linear equations
<u>MODE</u>	SF	DEG	RAD	GRAD	MULπ	CF	mode settings; SYSTEM is not available on the simulator
			RM		SETSIG	DENMA X	
	SYSTEM						
<u>MyMenu</u>							will show up out of AIM <sup>39</sup>
<u>Myα</u>							will show up in AIM <sup>39</sup>
<u>PARTS</u>	IP	FP	MANT	EXPT	sign	DECOMP	some overlaps with HP-42S CONVERT
					x	*	
					Re	Im	

<sup>39</sup> ... as long as no other menu is called (see Section 6 of the OM).

Menu							Remarks
<a href="#">PRINT</a>							the PRINT commands of the HP-42S
<a href="#">PROB</a>	Norml:	t:			F:	$\chi^2$ :	combinations, permutations, random number generators and 14 probability distributions. Selecting one (e.g. Norml) opens a submenu featuring 4 entries for its PDF (or PMF), CDF, error probability, and quantile function
	LgNrm:	Cauch:		Expon:	Logis:	Weibl:	
		NBin:	Geom:	Hyper:	Binom:	Poiss:	
	RAN#	SEED	RANI#		lnΓ	$\Gamma(x)$	
Binom:							
Cauch:							
Expon:							
F:	$F_p(x)$		$F_\Delta(x)$	$F_\Delta(x)$		$F^{-1}(p)$	
Geom:							
Hyper:							
LgNrm:							
Logis:							
NBin:							
Norml:							
Poiss:							
t:	$t_p(x)$		$t_\Delta(x)$	$t_\Delta(x)$		$t^{-1}(p)$	
Weibl:							
$\chi^2$ :	$\chi^2_p(x)$		$\chi^2_\Delta(x)$	$\chi^2_\Delta(x)$		$(\chi^2)^{-1}$	

Menu							Remarks
<u>P.FN</u>	INPUT	END	ERR	TICKS	PAUSE	P.FN2	additional programming functions (avoided a multi-view menu here).
	PSTO	PRCL	VARMNU	MVAR	CONST	PUTK	
	R-CLR	R-COPY	R-SORT	R-SWAP	LocR	PopLR	
<u>P.FN2</u>	MENU	KEYG	KEYX	CLMENU	EXITall	RTN+1	
	SDL	SDR	MSG	NOP			
	BACK	CASE	SKIP	AGRAPH	PIXEL	POINT	
<u>STAT</u>	$\Sigma+$	$\bar{x}$	S	G	$s_m$	SUM	for sample statistics.
	$\Sigma-$	$\bar{x}_w$	$s_w$	$g_w$	$s_{mw}$		
	CLΣ	$\bar{x}_g$	$\epsilon$	$\epsilon_p$	$\epsilon_m$	PLot	
	L.R.	r	$s_{xy}$	cov	$\hat{x}$	$\hat{y}$	for curve fitting and 2d sample statistics.
	s(a)	$\bar{x}_h$					
		$\bar{x}_{RMS}$				OrthoF	
	LinF	ExpF	LogF	PowerF		BestF	for choosing the fit model(s)
	GaussF	CauchF	ParabF	HypF	RootF		
<u>STK</u>	x $\geq$	y $\geq$	z $\geq$	t $\geq$	$\geq$	DROPy	stack related operations.
<u>TEST</u>	x<?	x≤?	x=?	x≠?	x≥?	x>?	binary tests.
	INT?	EVEN?	ODD?	PRIME?	LEAP?	FP?	
	ENTRY?	KEY?	LBL?	STRI?	CONVG?	TOP?	
	x=+0?	x=-0?	x≈?	MATR?	CPX?	REAL?	
	SPEC?	NaN?		M.SQR?			
<u>TRI</u>	sin	arcsin	cos	arccos	tan	arctan	trigonometric & hyperbolic functions (cf. EXP).
	sinh	arsinh	cosh	arcosh	tanh	artanh	

Menu							Remarks
<u>U→</u>	E: °C→°F	P: °F→°C	year→s s→year	F&p: power ratio → dB	m: dB → power ratio	x: field ratio → dB	unit conversions (see pp. 148ff).
A:	acre → m <sup>2</sup>	m <sup>2</sup> → acre	ha → m <sup>2</sup>	m <sup>2</sup> → ha	acre <sub>US</sub> → m <sup>2</sup>	m <sup>2</sup> → acre <sub>US</sub>	units of area
E:	cal → J	J → cal	Btu → J	J → Btu	Wh → J	J → Wh	units of energy
F&p:	lbf → N in.Hg → Pa	N → lbf in.Hg → Pa	bar→Pa torr → Pa	Pa→bar Pa → torr	psi→Pa atm→Pa	Pa→psi Pa→atm	units of force and pressure
m:	lb.→kg stone → kg	kg→lb. kg → stone	cwt→kg short cwt→kg	kg→cwt kg → sh.cwt	oz → kg tr.oz → kg	kg → oz kg → tr.oz	units of mass
	ton→kg	kg→ton kg → ton	short ton → kg	kg → short ton	carat → kg	kg → carat	
P:	hp <sub>E</sub> → W	W → hp <sub>E</sub>	hp <sub>UK</sub> →W	W→hp <sub>UK</sub>	hp <sub>M</sub> → W	W → hp <sub>M</sub>	units of power
V:	gl <sub>UK</sub> →m <sup>3</sup> floz <sub>UK</sub> → m <sup>3</sup>	m <sup>3</sup> →gl <sub>UK</sub> m <sup>3</sup> → floz <sub>UK</sub>	qt.→m <sup>3</sup>	m <sup>3</sup> → qt. .	gl <sub>US</sub> →m <sup>3</sup> floz <sub>US</sub> → m <sup>3</sup>	m <sup>3</sup> →gl <sub>US</sub> m <sup>3</sup> → floz <sub>US</sub>	units of volume
X:	au → m mi. → m in. → m	m → au m → mi. m → in.	ly → m nmi. → m	m → ly m→nmi. .	pc → m ft. → m yd. → m	m → pc m → ft. m → yd.	units of length
	fathom → m	m → fathom	point → m	m → point	survey foot <sub>US</sub> → m	m → survey foot <sub>US</sub>	

Menu							Remarks
<u>X.FN</u>	AGM	B <sub>n</sub>	B <sub>n</sub> *	erf	erfc	Orthog	advanced mathematical functions like Beta, Bessel, etc.
	FIB	g <sub>d</sub>	g <sub>d</sub> <sup>-1</sup>	I <sub>xyz</sub>	IΓ <sub>p</sub>	IΓ <sub>q</sub>	
	J <sub>y</sub> (x)	lnβ	lnΓ	max	min	NEXTP	
	sinc	W <sub>m</sub>	W <sub>p</sub>	W <sup>-1</sup>	β(x,y)	γ <sub>xy</sub>	
	Γ <sub>xy</sub>	ζ(x)	(-1) <sup>x</sup>				
Orthog	H <sub>n</sub>	L <sub>m</sub>	L <sub>mx</sub>	P <sub>n</sub>	T <sub>n</sub>	U <sub>n</sub>	orthogonal polynomials
	H <sub>np</sub>						
<u>αINTL</u>	A a	Ā ā	Á á	Ă ĕ	À à	Ä ä	[α] catalog of all Latin letters provided. <sup>40</sup> All letters but one in this menu will change when case is switched in AIM – note you will see the individual letters displayed in either case only at one time.
	Ã ã	Â â	Å å	Æ æ	Ą ą	Ɓ ɓ	
	C c	Ć ć	Č č	Ҫ ҫ	D d	Đ đ	
	Ď d'	E e	Ē ē	É é	Ě ě	È è	
	Ë ë	Ê ê	È è	Ë ë	Ӭ ӫ	F f	
	G g	Ӯ ӻ	H h	I i	Ӣ Ӣ	Ӣ Ӣ	
	Ӣ Ӣ	Ӣ Ӣ	Ӣ Ӣ	Ӣ Ӣ	Ӣ Ӣ	Ӣ Ӣ	
	J j	K k	L l	Ӆ ӑ	Ӆ ӑ	Ӆ ӑ	
	M m	N n	Ǹ ǹ	Ǹ ǹ	Ǹ ǹ	O o	
	Ӱ Ӱ	Ӱ Ӱ	Ӱ Ӱ	Ӱ Ӱ	Ӱ Ӱ	Ӱ Ӱ	
	Ӱ Ӱ	Ӱ Ӱ	Ӱ Ӱ	Ӱ Ӱ	Ӱ Ӱ	R r	
	Ӯ Ӯ	Ӯ Ӯ	S s	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	
	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	
	Ӯ Ӯ	Ӯ Ӯ	V v	Ӯ Ӯ	Ӯ Ӯ	X x	
	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Z z	
	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	Ӯ Ӯ	

<sup>40</sup> See [https://de.wikipedia.org/wiki/Liste\\_lateinischer\\_Alphabete#Erweiterungen](https://de.wikipedia.org/wiki/Liste_lateinischer_Alphabete#Erweiterungen).

Menu							Remarks
<a href="#">αMATH</a>	<	≤	=	≈	≥	>	[α] for comparison symbols, parentheses & brackets, as well as more mathematical and related symbols. You can reach every character by 3 keystrokes maximum.
	{	[	(	)	]	}	
	x / . <sup>41</sup>	÷ / :	∫	∞	∞	∞	
	¬	∧	∨	≠		&	
	✗	✗	✗	✗	✓	✗	
	✗	✗	✗	✗	✗	✗	
	:=	≈	≡	ᴱ	℃	ℝ	
	⊗	⊗	⊕				
	±	^	ᵀ	-¹	ℏ		
<a href="#">α.FN</a>	x→α	αRL	αRR	αSL	αSR	α→x	dedicated functions for <i>alpha</i> strings, plus font browser
					αLENG?	αPOS?	
	FBR						
<a href="#">Α...Ω</a>	Α α	Β β	Γ γ	Δ δ	Ε ε	Ζ ζ	[α] Greek letters. The keyboard grants direct access to 24 of them. <sup>42</sup> Note the two kinds of lowercase Σ. See αINTL for more.
	Η η	Θ θ	Ι ι	Κ κ	Λ λ	Μ μ	
	Ν ν	Ξ ξ	Ο ο	Π π	Ρ ρ	Σ σ	
	Ϛ	Ͳ τ	Ƴ ү	Փ Փ	Խ Ӯ	Ψ Ӯ	
	Ω ω	ά	έ	ή	ি	ő	
	Ϊ ő	ó	ú	ÿ ü	ö	á	

<sup>41</sup> With startup default settings, the multiplication dot is found here and the multiplication cross is called via in A/M. If MULT- is set, however, this dot is called via in A/M and the multiplication cross via [αMATH](#). The symbols : and ÷ will swap, too.

<sup>42</sup> The Greek alphabet (sic!) goes **alpha**, **beta**, **gamma**, **delta**, **ε-epsilon**, **zeta**, **ēta**, **theta**, **iota**, **kappa**, **lambda**, **my**, **ny**, **xi**, **ο-mikron**, **pi**, **rho**, **sigma**, **tau**, **y-psilon**, **phi**, **chi**, **psi**, **ō-mega**. About pronunciation, note that ancient Greek H, Θ, and Y are pronounced like Finnish ÄÄ, T, and Y; Finnish Y is spoken like French U or German Ü. Think of Nils Holgersson's goose Yksi (followed by Kaksi, Kolme, Neljä, Viisi, and Kuusi for

Menu							Remarks
$\alpha\bullet$	!	;	:	'	"	✓	[ $\alpha$ ] for punctuation marks, currency symbols, arrows, and further special characters.
	£	€	%	&	£	¥	
	←	↑	↓	↓	→	↖	
	«	»	»	⌚	•	*	
	⌚	⌚	⌚	⌚	⌚	*	
	„	”	...	—			
$\Sigma$	n	$\Sigma x$	$\Sigma x^2$	$\Sigma xy$	$\Sigma y^2$	$\Sigma y$	all the sums necessary for the statistics in STAT.
		$\Sigma \ln x$	$\Sigma \ln^2 x$	$\Sigma \ln xy$	$\Sigma \ln^2 y$	$\Sigma \ln y$	
	$\Sigma x^2 y$	$\Sigma x \ln y$		$\Sigma \ln y/x$		$\Sigma y \ln x$	
	$\Sigma x^2/y$	$\Sigma^1/x$	$\Sigma^1/x^2$	$\Sigma x/y$	$\Sigma^1/y^2$	$\Sigma^1/y$	
	$\Sigma x^3$	$\Sigma x^4$					
$\leftarrow\rightarrow$	$\rightarrow$ DEG	$\rightarrow$ RAD	$\rightarrow$ GRAD		$\rightarrow$ D.MS	$\rightarrow$ MUL $\pi$	angular conversions, cf. pp. 154f.
	DEG $\rightarrow$	RAD $\rightarrow$	GRAD $\rightarrow$		D.MS $\rightarrow$	MUL $\pi$ $\rightarrow$	
	D $\rightarrow$ R	R $\rightarrow$ D		D $\rightarrow$ D.MS	D.MS $\rightarrow$ D		

## Constants

Your WP 43S contains a catalog of 77 physical, astronomical, and mathematical constants:

G	$G_0$	$G_c$	$g_e$	$GM_{\oplus}$	$g_{\oplus}$	
$c_2$	e	$e_E$	F	$F_{\alpha}$	$F_{\delta}$	
a	$a_0$	$a_M$	$a_{\oplus}$	c	$c_1$	

Names of astronomical and mathematical constants are printed on colored background in the table starting overleaf. Values of physical

obvious reasons – these suffice: there is no goose named Seitsemän appearing in that novel).

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.<sup>43</sup> We use commas as radix marks for better visibility and multiplication dots for space reasons. Formulas are printed where applicable.

Name	Numeric value and <i>rel. SD</i>	Remarks
<b>a</b> (0) <sup>44</sup>	365,242.5 d <i>(per definition)</i>	<i>Gregorian year</i>
<b><math>a_0</math></b>	$5,291\ 772\ 109\ 03 \cdot 10^{-11}\ \text{m}$ <i>(<math>1,5 \cdot 10^{-10}</math>)</i>	<i>Bohr radius</i> $a_0 = \alpha / 4\pi R_\infty$

<sup>43</sup> For most of the physical constants, their precise numeric values (incl. their units) and their relative standard deviations (*SD*) are from CODATA 2018, copied in May 2019. These are the best values known in the scientific community, agreed on by the national standards institutes worldwide (e.g. by NIST and PTB). Note that the fundamental constants (printed **bold** in the table) of physics all feature less than 16 significant digits.

*Relative uncertainties* are included in the printed table here though not contained in CONST. 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> giving 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 we cannot know anything about a real-life object or process any better than we can measure it, these considerations are of fundamental importance. We recommend watching them – in your very own interest.

<sup>44</sup> The counting numbers in parentheses are to support determination of parameters for CONST – see the *IOI*.

Name	Numeric value and <i>rel. SD</i>	Remarks
$a_{\text{Moon}}$	$3,844 \cdot 10^8 \text{ m}$ $(1 \cdot 10^{-3})$	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> $\approx 499$ <i>light seconds</i> $\approx 8$ <i>light minutes</i> .
$c$	$2,997\,924\,58 \cdot 10^8 \text{ m/s}$ <b>(exact)</b>	<b>Speed of light in vacuum</b> $\approx 300\,000 \frac{\text{km}}{\text{s}} = 300 \frac{\text{km}}{\text{ms}} =$ $300 \frac{\text{m}}{\mu\text{s}} = 30 \frac{\text{cm}}{\text{ns}} = 0,3 \frac{\text{mm}}{\text{ps}}$ etc.
$c_1$ (5)	$3,741\,771\,85 \dots \cdot 10^{-16} \text{ W m}^2$ <b>(exact)</b>	First radiation constant $c_1 = 2\pi h c^2$
$c_2$	$0,014\,387\,768\,77 \dots \text{ m} \cdot \text{K}$ <b>(exact)</b>	Second radiation constant $c_2 = hc/k$
$e$	$1,602\,176\,634 \cdot 10^{-19} \text{ A s}$ <b>(exact)</b>	<b>Elementary charge</b> $e = \frac{2}{K_J R_K} = \Phi_0 G_0$
$e_E$	$2,718\,281\,828\,459\,045\,2\dots$	<i>Euler's e.</i>
$F$	$96\,485,332\,12 \dots \text{ A s/mol}$ <b>(exact)</b>	<i>Faraday constant</i> $F = e N_A$
$F_\alpha$ (10)	$2,502\,907\,875\,095\,892\,8\dots$	<i>Feigenbaum's</i> $\alpha$ and $\delta$
$F_\delta$	$4,669\,201\,609\,102\,990\,6\dots$	
$G$	$6,674\,30 \cdot 10^{-11} \text{ m}^3/\text{kg s}^2$ $(2,2 \cdot 10^{-5})$	Newtonian constant of gravitation; also known as $\gamma$ from other authors. See $\text{GM}_\oplus$ below for a more precise value.

Name	Numeric value and <i>rel. SD</i>	Remarks
$G_0$	$7,748\,091\,729\dots \cdot 10^{-5} / \Omega$ <i>(exact)</i>	Conductance quantum $G_0 = 2e^2/h = 2/R_K = eK_j$
$G_C$	$0,915\,965\,594\,177\,219\,0\dots$	<i>Catalan's constant</i>
$g_e$ (15)	$-2,002\,319\,304\,362\,56$ <i>(<math>1,7 \cdot 10^{-13}</math>)</i>	<i>Landé's electron g-factor</i>
$GM_{\oplus}$	$3,986\,004\,418 \cdot 10^{14} \text{ m}^3/\text{s}^2$ <i>(<math>2,0 \cdot 10^{-9}</math>)</i>	<i>Newtonian constant of gravitation times the Earth's mass with its atmosphere included (according to WGS84<sup>45</sup>)</i>
$g_{\oplus}$	$9,806\,65 \text{ m/s}^2$ ( <i>per def.</i> )	Standard earth acceleration
$h$	$6,626\,070\,15 \cdot 10^{-34} \text{ J s}$ <i>(exact)</i>	<b>Planck constant</b>
$\hbar$	$1,054\,571\,817\dots \cdot 10^{-34} \text{ J s}$ <i>(exact)</i>	Reduced <i>Planck constant</i> $\hbar = h/2\pi$
$k$ (20)	$1,380\,649 \cdot 10^{-23} \text{ J/K}$ <i>(exact)</i>	<b>Boltzmann constant</b> $k = R/N_A$
$K_J$	$4,835\,978\,48\dots \cdot 10^{14} \text{ Hz/V}$ <i>(exact)</i>	<i>Josephson constant</i> $K_j = 2e/h$
$l_{PL}$	$1,616\,255 \cdot 10^{-35} \text{ m}$ <i>(<math>1,1 \cdot 10^{-5}</math>)</i>	<i>Planck length</i> $l_{PL} = t_{PL}c$
$m_e$	$9,109\,383\,701\,5 \cdot 10^{-31} \text{ kg}$ <i>(<math>3,0 \cdot 10^{-10}</math>)</i>	Electron mass $\triangleq 511,00 \text{ keV}$
$M_{\text{Moon}}$	$7,349 \cdot 10^{22} \text{ kg}$ <i>(<math>5 \cdot 10^{-4}</math>)</i>	Mass of the Moon

<sup>45</sup> See [http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350\\_2.html](http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html)

Name	Numeric value and <i>rel. SD</i>	Remarks
$m_n$ (25)	$1,674\,927\,498\,04 \cdot 10^{-27}$ kg $(5,7 \cdot 10^{-10})$	Neutron mass $\triangleq 939,57$ MeV
$m_n/m_p$	1,001 378 419 31 $(4,9 \cdot 10^{-10})$	Neutron to proton mass ratio
$m_p$	$1,672\,621\,923\,69 \cdot 10^{-27}$ kg $(3,1 \cdot 10^{-10})$	Proton mass $\triangleq 938,27$ MeV
$m_{PL}$	$2,176\,435 \cdot 10^{-8}$ kg $(1,1 \cdot 10^{-5})$	Planck mass $m_{PL} = \sqrt{\hbar c/G} \approx 22$ µg
$m_p/m_e$	1 836,152 673 43 $(6,0 \cdot 10^{-11})$	Proton to electron mass ratio
$m_u$ (30)	$1,660\,539\,066\,60 \cdot 10^{-27}$ kg $(3,0 \cdot 10^{-10})$	Atomic mass constant $\approx 10^{-3}$ kg/ $N_A$
$m_u c^2$	$1,492\,418\,085\,60 \cdot 10^{-10}$ J $(3,0 \cdot 10^{-10})$	Energy equivalent of the atomic mass constant $\approx 931,49$ MeV
$m_\mu$	$1,883\,531\,627 \cdot 10^{-28}$ kg $(2,2 \cdot 10^{-8})$	Muon mass $\triangleq 105,66$ MeV
$M_\odot$	$1,989\,1 \cdot 10^{30}$ kg $(5 \cdot 10^{-5})$	Mass of the Sun
$M_\oplus$	$5,973\,6 \cdot 10^{24}$ kg $(5 \cdot 10^{-5})$	Mass of the Earth. See $GM_\oplus$ above for a more precise value.
$N_A$ (35)	$6,022\,140\,76 \cdot 10^{23}$ / mol <b>(exact)</b>	<b>Avogadro's number</b>
NaN	<i>Not a Number</i>	See p. 174 and the corresponding entry in Section 5 of the OM.

Name	Numeric value and <i>rel. SD</i>	Remarks
$p_0$	101 325 Pa ( <i>per def.</i> )	Standard atmospheric pressure
$R$	$8,314\,462\,618\dots \text{ J/mol K}$ <i>(exact)</i>	Molar gas constant
$r_e$	$2,817\,940\,326\,2 \cdot 10^{-15} \text{ m}$ <i>(4,5 · 10<sup>-10</sup>)</i>	Classical electron radius $r_e = \alpha^2 a_0$
$R_K$ (40)	$25\,812,807\,45\dots \Omega$ <i>(exact)</i>	Von Klitzing constant $R_K = h/e^2$
$R_{\text{Moon}}$	$1,737\,530 \cdot 10^6 \text{ m}$ <i>(5 · 10<sup>-7</sup>)</i>	Mean radius of the Moon
$R_\infty$	$10\,973\,731,568\,160 / \text{m}$ <i>(1,9 · 10<sup>-12</sup>)</i>	Rydberg constant $R_\infty = \frac{\alpha^2 m_e c}{2 h}$
$R_\odot$	$6,96 \cdot 10^8 \text{ m}$ <i>(5 · 10<sup>-3</sup>)</i>	Mean radius of the sun
$R_\oplus$	$6,371\,010 \cdot 10^6 \text{ m}$ <i>(5 · 10<sup>-7</sup>)</i>	Mean radius of the Earth
$S_a$ (45)	$6,378\,137\,0 \cdot 10^6 \text{ m}$ ( <i>p. def.</i> )	Semi-major axis
$S_b$	$6,356\,752\,314\,2 \cdot 10^6 \text{ m}$ <i>(1,6 · 10<sup>-11</sup>)</i>	Semi-minor axis
$S_e^2$	$6,694\,379\,990\,14 \cdot 10^{-3}$ <i>(1,5 · 10<sup>-12</sup>)</i>	First eccentricity squared
$S_e'^2$	$6,739\,496\,742\,28 \cdot 10^{-3}$ <i>(1,5 · 10<sup>-12</sup>)</i>	Second eccentricity squared
$S_f^{-1}$	298,257 223 563 ( <i>per def.</i> )	Flattening parameter

Name	Numeric value and <i>rel. SD</i>	Remarks
$T_0$ (50)	273,15 K ( <i>per definition</i> )	= 0°C, standard temperature
$T_p$	$1,416\,785 \cdot 10^{32}$ K $(1,1 \cdot 10^{-5})$	<i>Planck temperature</i> $T_p = \frac{c^2}{k} \sqrt{\frac{\hbar c}{G}} = \frac{M_p c^2}{k} = \frac{E_p}{k}$
$t_{PL}$	$5,391\,245 \cdot 10^{-44}$ s $(1,1 \cdot 10^{-5})$	<i>Planck time</i> $t_{PL} = l_{PL}/c$
$V_m$	$0,022\,413\,969\,5\dots$ m <sup>3</sup> /mol <i>(exact)</i>	Molar volume of an ideal gas at standard conditions $V_m = \frac{RT_0}{P_0} \approx 22,4 \text{ l/mol}$
$Z_0$	$376,730\,313\,668 \Omega$ $(1,5 \cdot 10^{-10})$	Characteristic impedance of vacuum
$\alpha$ (55)	$7,297\,352\,569\,3 \cdot 10^{-3}$ $(1,5 \cdot 10^{-10})$	Fine-structure constant $\alpha = \frac{e^2}{2\varepsilon_0 h c} \approx \frac{1}{137}$
$\gamma$	$6,674\,30 \cdot 10^{-11}$ m <sup>3</sup> /kg s <sup>2</sup> $(2,2 \cdot 10^{-5})$	<i>Newtonian</i> constant of gravitation; also known as G from other authors. See GM <sub>⊕</sub> below for a more precise value.
$\gamma_{EM}$	$0,577\,215\,664\,901\,532\,9\dots$	<i>Euler-Mascheroni</i> constant
$\gamma_p$	$2,675\,221\,874\,4 \cdot 10^8$ Hz/T $(4,2 \cdot 10^{-10})$	Proton gyromagnetic ratio $\gamma_p = 4\pi \mu_p / h$
$\Delta\nu_{Cs}$	<b>9 192 631 770 Hz</b> <i>(exact)</i>	<b>Hyperfine transition frequency of</b> <b><sup>133</sup>Cs</b>

Name	Numeric value and <i>rel. SD</i>	Remarks
$\epsilon_0$ (60)	$8,854\ 187\ 812\ 8 \cdot 10^{-12} \frac{\text{As}}{\text{Vm}}$ $(1,5 \cdot 10^{-10})$	Vacuum electric permittivity $\epsilon_0 = 1 / \mu_0 c^2$ (note the so-called Coulomb's constant is just $1 / 4\pi\epsilon_0$ )
$\lambda_c$	$2,426\ 310\ 238\ 67 \cdot 10^{-12} \text{ m}$ $(3,0 \cdot 10^{-10})$	Compton wavelengths of the electron
$\lambda_{cn}$	$1,319\ 590\ 905\ 81 \cdot 10^{-15} \text{ m}$ $(5,7 \cdot 10^{-10})$	$\lambda_c = h/m_e c$ , neutron $\lambda_{cn} = h/m_n c$ , and proton $\lambda_{cp} = h/m_p c$ , respectively
$\lambda_{cp}$	$1,321\ 409\ 855\ 39 \cdot 10^{-15} \text{ m}$ $(3,1 \cdot 10^{-10})$	
$\mu_0$	$1,256\ 637\ 062\ 12 \cdot 10^{-6} \frac{\text{Vs}}{\text{Am}}$ $(1,5 \cdot 10^{-10})$	Vacuum magnetic permeability
$\mu_B$ (65)	$9,274\ 010\ 078\ 3 \cdot 10^{-24} \frac{\text{J}}{\text{T}}$ $(3,0 \cdot 10^{-10})$	Bohr magneton $\mu_B = e\hbar / 2m_e$
$\mu_e$	$-9,284\ 764\ 704\ 3 \cdot 10^{-24} \frac{\text{J}}{\text{T}}$ $(3,0 \cdot 10^{-10})$	Electron magnetic moment
$\mu_e/\mu_B$	$-1,001\ 159\ 652\ 181\ 28$ $(1,7 \cdot 10^{-13})$	Ratio of electron magnetic moment to Bohr's magneton
$\mu_n$	$-9,662\ 365\ 1 \cdot 10^{-27} \frac{\text{J}}{\text{T}}$ $(2,4 \cdot 10^{-7})$	Neutron magnetic moment
$\mu_p$	$1,410\ 606\ 797\ 36 \cdot 10^{-26} \frac{\text{J}}{\text{T}}$ $(4,2 \cdot 10^{-10})$	Proton magnetic moment

Name	Numeric value and <i>rel. SD</i>	Remarks
$\mu_u$ (70)	$5,050\,783\,746\,1 \cdot 10^{-27} \text{ J/T}$ <i>(<math>3,1 \cdot 10^{-10}</math>)</i>	Nuclear magneton $\mu_u = e\hbar/2m_p$
$\mu_\mu$	$-4,490\,448\,30 \cdot 10^{-26} \text{ J/T}$ <i>(<math>2,2 \cdot 10^{-8}</math>)</i>	Muon magnetic moment
$\sigma_B$	$5,670\,374\,41 \dots 10^{-8} \frac{\text{W}}{\text{m}^2 \text{K}^4}$ <i>(exact)</i>	Stefan-Boltzmann constant $\sigma_B = \frac{2\pi^5 k^4}{15h^3 c^2}$
$\Phi$	$1,618\,033\,988\,749\,894\,8\dots$	Golden ratio $\Phi = \frac{1}{2}(1 + \sqrt{5})$
$\Phi_0$	$2,067\,833\,848 \dots 10^{-15} \text{ V s}$ <i>(exact)</i>	Magnetic flux quantum $\Phi_0 = \frac{h}{2e}$
$\omega$ (75)	$7,292\,115 \cdot 10^{-5} \text{ rad/s}$ <i>(<math>2 \cdot 10^{-8}</math>)</i>	Angular velocity of the Earth according to WGS84 (see footnote 45 on p. 141).
$-\infty$	$-\infty$	Note both these ‘constants’ are counted as numeric values in your WP 43S. They can be recalled and used with SPCRES set, else an error will be thrown.
$\infty$	$\infty$	

Some values found in nature are known with up to one or two digits precision only – thus, they are not stored in CONST but just printed below. They may be helpful for quick estimates nevertheless:

Radius of an atomic nucleus  $\sim 10^{-15} \text{ m}$

Radius of an atom <sup>46</sup>  $\sim 10^{-10} \text{ m}$

---

<sup>46</sup> 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. You can even touch many of these real-world objects. Think about it!

Radius of our home galaxy	$\sim 10^5 \text{ l.y.} \approx 10^{21} \text{ m}$
Radius of the observable universe <sup>47</sup>	$\approx 45 \times 10^9 \text{ l.y.} \approx 4.3 \times 10^{26} \text{ m}$
Number of stars in our home galaxy <sup>48</sup>	$\sim 2 \times 10^{11}$
Number of neuron connections in human brain	$\sim 10^{14}$
Number of stars in observable universe	$\sim 10^{23}$
Amount of atoms in the Sun <sup>49</sup>	$\sim 10^{57}$
Amount of atoms in observable universe	$\sim 10^{80}$
Baryonic mass in observable universe	$\sim 10^{53} \text{ kg}$

Note these quantities are all far within the range of *real* numbers allowed on your WP 43S (see App. B). Physical constants are seldom more precisely known than twelve digits (cf. the table above). Please take these facts into account when assessing very small differences as well as talking about very large numbers.

---

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

<sup>47</sup> For more information, please see [https://en.wikipedia.org/wiki/Observable\\_universe](https://en.wikipedia.org/wiki/Observable_universe) (or [https://de.w.../Beobachtbares\\_Universum](https://de.w.../Beobachtbares_Universum) or [https://fr.w.../Univers\\_observable](https://fr.w.../Univers_observable) or [https://es.wikipedia.org/wiki/Universo\\_observable](https://es.wikipedia.org/wiki/Universo_observable) etc. – the latter site includes a picture [https://es.wikipedia.org/wiki/Universo\\_observable#/media/Archivo:Universoobservable.PNG](https://es.wikipedia.org/wiki/Universo_observable#/media/Archivo:Universoobservable.PNG) explaining the difference between the radius printed above and the naïve assumption of  $13.8 \times 10^9 \text{ l.y.}$  for it; this picture is not found on the other sites).

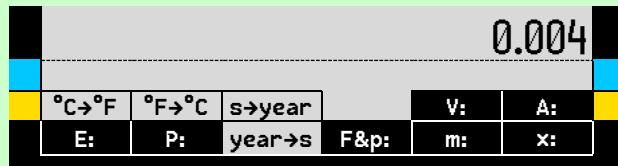
Note the radius of the observable universe divided by the radius of our home galaxy returns a value in the same ballpark as the radius of an atom divided by the radius of a nucleus.

<sup>48</sup> Assume our home galaxy fills a huge flat cylinder 1000 l.y. high, what will be the average distance between its stars? So what will happen when two such galaxies collide?

<sup>49</sup> You can take this number for the amount of atoms in our solar system as well – the stuff beyond the sun is neglectable here.

## Unit Conversions

Your WP 43S features 14 angular conversions provided in 4→ (cf. p. 138) and 88 unit conversions in U→. The structure of U→ follows various branches as explained 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 pp. 135f for more details of the structure.

Conversions contained in U→ either begin or end in one of the seven basic *SI* units: *kelvin*, *meter*, *kilogram*, *second*, *ampere*, *mol*, and *candela*. 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 and communications:

Quantity	Unit	Symbol and formula
Temperature	<i>degree Celsius</i>	$\vartheta [^\circ\text{C}] = T [\text{K}] - 273.15$
Force	<i>newton</i>	$1 \text{ N} = 1 \text{ kg m/s}^2$
Pressure	<i>pascal</i>	$1 \text{ Pa} = 1 \text{ N/m}^2 = 1 \frac{\text{kg}}{\text{m s}^2}$
Energy	<i>joule</i>	$1 \text{ J} = 1 \text{ N m} = 1 \text{ kg m}^2/\text{s}^2$
Power	<i>watt</i>	$1 \text{ W} = 1 \text{ V A} = 1 \frac{\text{J}}{\text{s}}$
Electric potential	<i>volt</i>	$1 \text{ V} = 1 \frac{\text{W}}{\text{A}}$
Charge	<i>coulomb</i>	$1 \text{ C} = 1 \text{ A s}$
Capacitance	<i>farad</i>	$1 \text{ F} = 1 \frac{\text{C}}{\text{V}} = 1 \frac{\text{A s}}{\text{V}}$
Conductance	<i>siemens</i>	$1 \text{ S} = 1 \frac{\text{A}}{\text{V}}$
Resistance	<i>ohm</i>	$1 \Omega = 1 \frac{\text{V}}{\text{A}}$
Magnetic flux	<i>weber</i>	$1 \text{ Wb} = 1 \text{ V s}$
Magnetic flux density	<i>tesla</i>	$1 \text{ T} = 1 \frac{\text{Wb}}{\text{m}^2} = 1 \frac{\text{V s}}{\text{m}^2}$
Inductance	<i>henry</i>	$1 \text{ H} = 1 \frac{\text{Wb}}{\text{A}} = 1 \frac{\text{V s}}{\text{A}}$
Frequency	<i>hertz</i>	$1 \text{ Hz} = 1 \frac{1}{\text{s}}$
Absorbed dose	<i>gray</i>	$1 \text{ Gy} = 1 \frac{\text{J}}{\text{kg}}$

For talking about inputs and results, knowing the symbols and names of the SI prefixes as listed below is beneficial, covering 36 orders of magnitude:

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

Prefix	Name	Factor
d	deci-	$10^{-1}$
c	centi-	$10^{-2}$
m	milli-	$10^{-3}$
$\mu$	micro-	$10^{-6}$
n	nano-	$10^{-9}$
p	pico-	$10^{-12}$
f	femto-	$10^{-15}$
a	atto-	$10^{-18}$

All conversions featured in **U→** and **↓→** are explained in alphabetical order below. Numeric values are either exact (printed on white background) or rounded to six significant digits for your orientation (your WP 43S uses more precise values where applicable). Commas are printed as radix marks below for better visibility.

Softkey	Calculation	Remarks	Branch
<b>°C → °F</b>	$x 1,8 + 32$	$\theta [{}^\circ\text{C}] = (T [\text{K}] + T_0)$ and $t [{}^\circ\text{F}] = (t [{}^\circ\text{R}] + T_0 \times 1,8)$	<b>U→</b>
<b>°F → °C</b>	$- 32 ) / 1,8$		
<b>acre → m<sup>2</sup></b>	$\times 4\,046,86$	These <i>acres</i> are based on the ' <i>international feet</i> ', see below	
<b>acre<sub>us</sub> → m<sup>2</sup></b>	$\times 4\,046,87$	These <i>acres</i> are based on the ' <i>U.S. survey feet</i> ', see below	<b>U→ f A:</b>
<b>atm → Pa</b>	$\times 1,013\,25 \times 10^5$	<i>Atmospheres</i>	<b>U→ F&amp;p:</b>
<b>au → m</b>	$\times 1,495\,98 \times 10^{11}$	<i>Astronomic units</i>	<b>U→ x:</b>
<b>barrel → m<sup>3</sup></b>	$\times 0,158\,987$	(U.S.) <i>barrels</i> of oil, abbr. bbl	<b>U→ f v:</b>
<b>bar → Pa</b>	$\times 10^5$	$1 \text{ mbar} = 1 \text{ hPa}$	<b>U→ F&amp;p:</b>
<b>Btu → J</b>	$\times 1\,055,06$	<i>British thermal units</i>	
<b>cal → J</b>	$\times 4,186\,8$	<i>Calories</i>	<b>U→ E:</b>

Softkey	Calculation	Remarks	Branch
<b>carat → kg</b>	$\times 0,000\ 2$		
<b>cwt → kg</b>	$\times 50,802\ 4$	1 ( <i>long</i> ) hundredweight := 112 lbs	<b>U→ m:</b>
<b>dB → field ratio</b>	$10^{R_{dB}/20}$	<i>Decibels</i>	
<b>dB → power ratio</b>	$10^{R_{dB}/10}$		<b>U→ ▼</b>
<b>fathom → m</b>	$\times 1,828\ 8$	1 fathom := 2 yards := 6 feet	<b>U→ x:</b>
<b>field ratio → dB</b>	$20 \lg(a_1/a_2)$	Also known as <i>amplitude ratio</i>	<b>U→ ▼</b>
<b>floz<sub>UK</sub> → m<sup>3</sup></b>	$\times 2,841\ 31 \times 10^{-5}$	<i>Fluid ounces</i>	<b>U→ f v:</b>
<b>floz<sub>US</sub> → m<sup>3</sup></b>	$\times 2,957\ 35 \times 10^{-5}$		<b>U→ f v:</b>
<b>ft. → m</b>	$\times 0,304\ 8$	These are the so-called ' <i>international feet</i> ' of 1959 1 foot := 12 inches	<b>U→ x:</b>
<b>gl<sub>UK</sub> → m<sup>3</sup></b>	$\times 4,546\ 09 \times 10^{-3}$	<i>Gallons</i>	<b>U→ f v:</b>
<b>gl<sub>US</sub> → m<sup>3</sup></b>	$\times 3,785\ 42 \times 10^{-3}$		<b>U→ f v:</b>
<b>ha → m<sup>2</sup></b>	$\times 10\ 000$	<i>Hectares</i>	<b>U→ f A:</b>
<b>hp<sub>E</sub> → W</b>	$\times 746$	<i>Electric horsepower</i>	
<b>hp<sub>M</sub> → W</b>	$\times 735,499$	So-called ' <i>metric</i> ' horsepower (equivalent to PS in German)	<b>U→ P:</b>
<b>hp<sub>UK</sub> → W</b>	$\times 745,700$	<i>British Imperial horsepower</i>	
<b>in. → m</b>	$\times 0,025\ 4$	1 inch := 1000 mil	<b>U→ x:</b>
<b>in.Hg → Pa</b>	$\times 3\ 386,39$	<i>Inches of mercury</i>	<b>U→ F&amp;p:</b>
<b>J → Btu</b>	/ 1 055,06	<i>Joules</i>	
<b>J → cal</b>	/ 4,186 8		<b>U→ E:</b>
<b>J → Wh</b>	/ 3 600		

Softkey	Calculation	Remarks	Branch
<b>kg → carat</b>	/ 0,000 2		
<b>kg → cwt</b>	/ 50,802 4		
<b>kg → oz</b>	/ 0,028 349 5		
<b>kg → lb.</b>	/ 0,453 592		
<b>kg → sh.cwt</b>	/ 45,359 2	1 t [(metric) ton] := 1000 kg	<b>U→ m:</b>
<b>kg → short ton</b>	/ 907,185		
<b>kg → stone</b>	/ 6,350 29		
<b>kg → ton</b>	/ 1 016,05		
<b>kg → tr.oz</b>	/ 0,031 103 5		
<b>lbf → N</b>	× 4,448 22	Pounds force	<b>U→ F&amp;p:</b>
<b>lb. → kg</b>	× 0,453 592	Pounds; 1 lb := 16 ounces	<b>U→ m:</b>
<b>ly → m</b>	× 9,460 73×10 <sup>15</sup>	Light years	<b>U→ x:</b>
<b>m<sup>2</sup> → acre</b>	/ 4 046,86	Square meters;	
<b>m<sup>2</sup> → acre<sub>US</sub></b>	/ 4 046,87	1 a [are] := 100 m <sup>2</sup> , 1 ha [hectare] := 10 000 m <sup>2</sup> , 1 km <sup>2</sup> = 100 ha = 1 000 000 m <sup>2</sup>	<b>U→ f A:</b>
<b>m<sup>2</sup> → ha</b>	/ 10 000		
<b>m<sup>3</sup> → barrel</b>	/ 0,158 987		
<b>m<sup>3</sup> → floz<sub>UK</sub></b>	/ 2,841 31×10 <sup>-5</sup>		
<b>m<sup>3</sup> → floz<sub>US</sub></b>	/ 2,957 35×10 <sup>-5</sup>	Cubic meters;	
<b>m<sup>3</sup> → gl<sub>UK</sub></b>	/ 4,546 09×10 <sup>-3</sup>	1 liter := 1 dm <sup>3</sup> = 10 <sup>-3</sup> m <sup>3</sup> , 1 ml [milliliter] = 1 cm <sup>3</sup>	<b>U→ f v:</b>
<b>m<sup>3</sup> → gl<sub>US</sub></b>	/ 3,785 42×10 <sup>-3</sup>		
<b>m<sup>3</sup> → qt.</b>	/ 1,1365×10 <sup>-3</sup>		

Softkey	Calculation	Remarks	Branch
mi. → m	$\times 1\,609,344$	1 mile := 1 760 yards	
m → au	$/ 1,495\,98 \times 10^{11}$		
m → fathom	$/ 1,828\,8$		
m → ft.	$/ 0,304\,8$		
m → in.	$/ 0,025\,4$		
m → ly	$/ 9,460\,73 \times 10^{15}$		
m → mi.	$/ 1\,609,344$	Meters	<b>[U→] x:</b>
m → nmi.	$/ 1\,852$		
m → pc	$/ 3,085\,68 \times 10^{16}$		
m → point	$/ 352,778 \times 10^{-6}$		
m → survey foot <sub>US</sub>	$/ 0,304\,801$		
m → yd.	$/ 0,914\,4$		
nmi. → m	$\times 1\,852$	Nautical miles	
N → lbf	$/ 4,448\,22$	Newton	<b>[U→] F&amp;p:</b>
oz → kg	$\times 0,028\,349\,5$	Ounces	<b>[U→] m:</b>
Pa → atm	$/ 1,013\,25 \times 10^5$		
Pa → bar	$/ 10^5$	Pascals; 1 hPa = 1 mbar	
Pa → in.Hg	$/ 3\,386,39$		<b>[U→] F&amp;p:</b>
Pa → psi	$/ 6\,894,76$		
Pa → torr	$/ 133,322$		
pc → m	$\times 3,085\,68 \times 10^{16}$	Parsecs	<b>[U→] x:</b>
point → m	$\times 352,778 \times 10^{-6}$	(Typographical) points	
power ratio → dB	$10 \lg(p_1/p_2)$		<b>[U→] ▼</b>

Softkey	Calculation	Remarks	Branch
psi → Pa	× 6 894,76	Pounds per square inch	[U→] F&p:
qt. → m <sup>3</sup>	× 1,1365×10 <sup>-3</sup>	1 (Imperial) quart := 40 fl.oz.UK	[U→] f V:
short cwt → kg	× 45,359 2	1 short hundredweight := 100 lbs	
short ton → kg	× 907,185	1 short ton := 2000 lbs	[U→] m:
stone → kg	× 6,350 29		
survey foot <sub>US</sub> → m	× 0,304 801	1 U.S. survey foot := $\frac{1200}{3937}$ m	[U→] x:
s → year	/ 31 556 952		[U→]
ton → kg	× 1 016,05	1 Imperial ton := 200 cwt	[U→] m:
torr → Pa	× 133,322	1 torr = 1 mm Hg	[U→] F&p:
tr.oz → kg	× 0,031 103 5	Troy ounces	[U→] m:
Wh → J	× 3 600	Watt-hours	[U→] E:
W → hp <sub>E</sub>	/ 746	Watts	
W → hp <sub>M</sub>	/ 735,499		[U→] P:
W → hp <sub>UK</sub>	/ 745,700		
yd. → m	× 0,914 4	1 yard := 3 feet	[U→] x:
year → s	× 31 556 952	= 365,242 5 × 24 × 60 <sup>2</sup>	[U→]

[L→] ...	Remarks
DEG→	Takes an integer or real <sup>50</sup> $x$ as an angular input in decimal or sexagesimal degrees, resp., and converts it to the current ADM.
D.MS→	
D.MS→D	Takes an integer or real <sup>50</sup> $x$ as an angular input in sexagesimal degrees (formatted dddd.dd.msshh) and converts it to an angle in decimal degrees (corresponding to the old command H.MS→H).

...	Remarks
<b>D→R</b>	Takes an integer or <i>real</i> <sup>50</sup> $x$ as an angular input in <i>decimal degrees</i> and converts it to ... <i>radians</i> . <sup>51</sup>
<b>D→D.MS</b>	... <i>sexagesimal degrees</i> (corresponding to the old command H→H.MS).
<b>GRAD→</b>	... <i>grad/gon</i> ...
<b>MULπ→</b>	Takes an integer or <i>real</i> <sup>50</sup> $x$ as an angular input in ... <i>... multiples of π ...</i> ... and converts it to the current <i>ADM</i> .
<b>RAD→</b>	... <i>radians</i> <sup>51</sup> ...
<b>R→D</b>	Takes an integer or <i>real</i> <sup>50</sup> $x$ as an angular input in <i>radians</i> and converts it to <i>decimal degrees</i> (equaling the old command R→D). <sup>51</sup>
<b>→DEG</b>	Takes an integer or <i>real</i> ( <i>data type 2</i> ) $x$ as an angular input in the current <i>ADM</i> and converts it to <i>decimal degrees</i> , <i>sexagesimal degrees</i> , <i>grad/gon</i> , <i>multiples of π</i> , or <i>radians</i> , respectively. <sup>51</sup>
<b>→D.MS</b>	If $x$ is a tagged <i>real</i> ( <i>data type 4</i> ), on the other hand, this information is used in conversion (e.g. if $x = 1.5\pi$ then →GRAD will return <b>300°</b> regardless of current <i>ADM</i> ).
<b>→GRAD</b>	
<b>→MULπ</b>	
<b>→RAD</b>	If $x$ is neither of <i>data type 1</i> , <i>2</i> , nor <i>4</i> , error 24 will be thrown ('illegal input data type for this operation').

Angular output is tagged always.

<sup>50</sup> If  $x$  is neither integer nor *real* (i.e. neither *data type 1* nor *2*), error 24 will be thrown. Conversions of integer values to *angles* in *sexagesimal degrees* do not make real sense but are allowed nevertheless.

<sup>51</sup> Note that *real* angles given in *radians* cannot represent full circles (or simple fractions of  $\pi$  like  $\pi/2$ ,  $\pi/3$ ,  $\pi/4$ ,  $\pi/5$ , etc.) exactly but with an accuracy of 34 digits 'only'. If you want to avoid most rounding errors caused by that, *multiples of π* may be a better choice here. Note large numeric inputs in trigonometric functions will be reduced to values between  $-\pi$  and  $+\pi$  before calculating (as mentioned in *Section 2* of the *OM*).

## **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 know how to call *items* appearing on the keyboard or in *menus* (including *catalogs*). In *Section 6* of the *OM*, you have learned about storing *items* in user *menus* and/or assigning them to specific locations on your *WP 43S*. There is one more way you can use for calling and executing operations: take **[XEQ]** followed by the *name* of the operation typed in *AIM*.

In the two chapters following thereafter, we will list all the functions requiring parameters and those changing *data types*.

### **Using XEQ for Executing Operations**

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

1. Press **[XEQ]**.
2. Press **[ $\alpha$ ]**. 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. Case may be important, subscript or superscript is not.
4. Press **[ENTER $\uparrow$ ]**. 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 (numeric or alphanumeric) parameter specifying what they shall do precisely (see the OM, Sect. 1). The following three lists summarize these operations:

Operations requiring one trailing parameter	Numeric parameter	Alpha par.
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 $\ddagger$ VIEW x $\ddagger$ x=? x≠? x≈? x<? x≤? x≥? x>? y $\ddagger$ z $\ddagger$ aLEN? aPOS? aRL aRR aSL a $\rightarrow$ 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	
BestF	Fit model code	
CF FC? FC?C FC?F FC?S FF FS? FS?C FS?F FS?S SF	User flag number	System flag name
CONST	Constant number	
DSTACK	Number of stack registers	
ERR MSG	Error number	
f'(x) f''(x) GTO LBL LBL? PGMINT PGMSLV XEQ $\Pi_n$ $\Sigma_n$		Program label
GTO.	Number of program step	Label

Operations requiring one trailing parameter	Numeric parameter	Alpha par.
INDEX MVAR M.DIM MEDIN SOLVE VARMNU ]		Variable name
LocR	Number of local registers	
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 for any command **XYZ** requiring one trailing parameter, you can enter

**XYZ → ST.X**

and it will fetch its parameter from **X** like a good old *RPN* command instead.

Operations requiring two trailing parameters	First parameter	Second parameter
ASSIGN	/item	Sequence of keystrokes
KEYG KEYX	Key number (1 ... 18)	Program label

Operation requiring four trailing parameters	1 <sup>st</sup> to 4 <sup>th</sup> parameter
»	Name of stack register

## Operations Changing Data Types

Most functions will return data of the same type they operate on. Some, however, will change the *data type (DT)* of the lowest *stack register(s)* regardless of 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
1	1/x $\sqrt[3]{x}$ AGM ALL cos ENG erf erfc e <sup>x</sup> FIX f' f'' gd gd <sup>-1</sup> J <sub>y</sub> (x) LN LN $\beta$ LN $\Gamma$ LOG <sub>10</sub> LOG <sub>2</sub> LOG <sub>xy</sub> MANT POISS... SCI sin tan W <sub>m</sub> W <sub>p</sub> W <sup>-1</sup> $\sqrt{y}$ $\beta(x,y)$ $\Gamma_{xy}$ $\gamma_{xy}$ $\Gamma(x)$ Δ% →REAL % %MRR %T %Σ %+MG $\sqrt{x}$    as well as all unit conversions and all orthogonal polynomials	2 <sup>52</sup>	X
	→POL →REC	2	X, Y
	SLVQ	2 or 3	X, Y, Z
	f'(x) f''(x) SOLVE	2	X, Y, Z, T
	all angular conversions	4	X
	→H.MS	5	X
	J→D →DATE	6	X
	x→α	7	X
	M.GET M.NEW	8 or 9	X
	→INT	10	X

<sup>52</sup> The functions printed on yellow background will return *long integers (data type 1)* wherever possible.

Input DT	Operation(s)	Output DT	Output registers
2	AND CEIL DATE→ DAY D→J EXPT FLOOR IDIV IDIVR IP MONTH NAND NEXTP NOR NOT OR ROUNDI 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
3	ABS CROSS IM RE  x  4	2	X
	CX→RE	2	X, Y
	SLVQ	2 or 3	X, Y, Z
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
7	α→x	1	X

Input <i>DT</i>	Operation(s)	Output <i>DT</i>	Output <i>registers</i>
8	M.DIM?	1	X, Y
	DET DOT ENORM  M	2	X
	$\Sigma+$	2	X, Y, statistic registers
	V <sub>4</sub>	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
	$e^x$ LN LOG <sub>x</sub> y →REAL	2	X
	x→α	7	X

## **APPENDIX A: HARDWARE**

Overall dimensions: wedge-shaped: 77 mm × 144 mm × 13 mm or 8 mm (see p. 163)

Mass with battery: ~100 g

LCD dimensions: about 58.8 mm × 35.3 mm visible area, 400 × 240 quadratic pixels monochrome

Processor: STMicroelectronics STM32L476 incl. RTC (see <https://www.st.com/en/evaluation-tools/32l476gdiscovery.html> 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. 167ff),  
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**

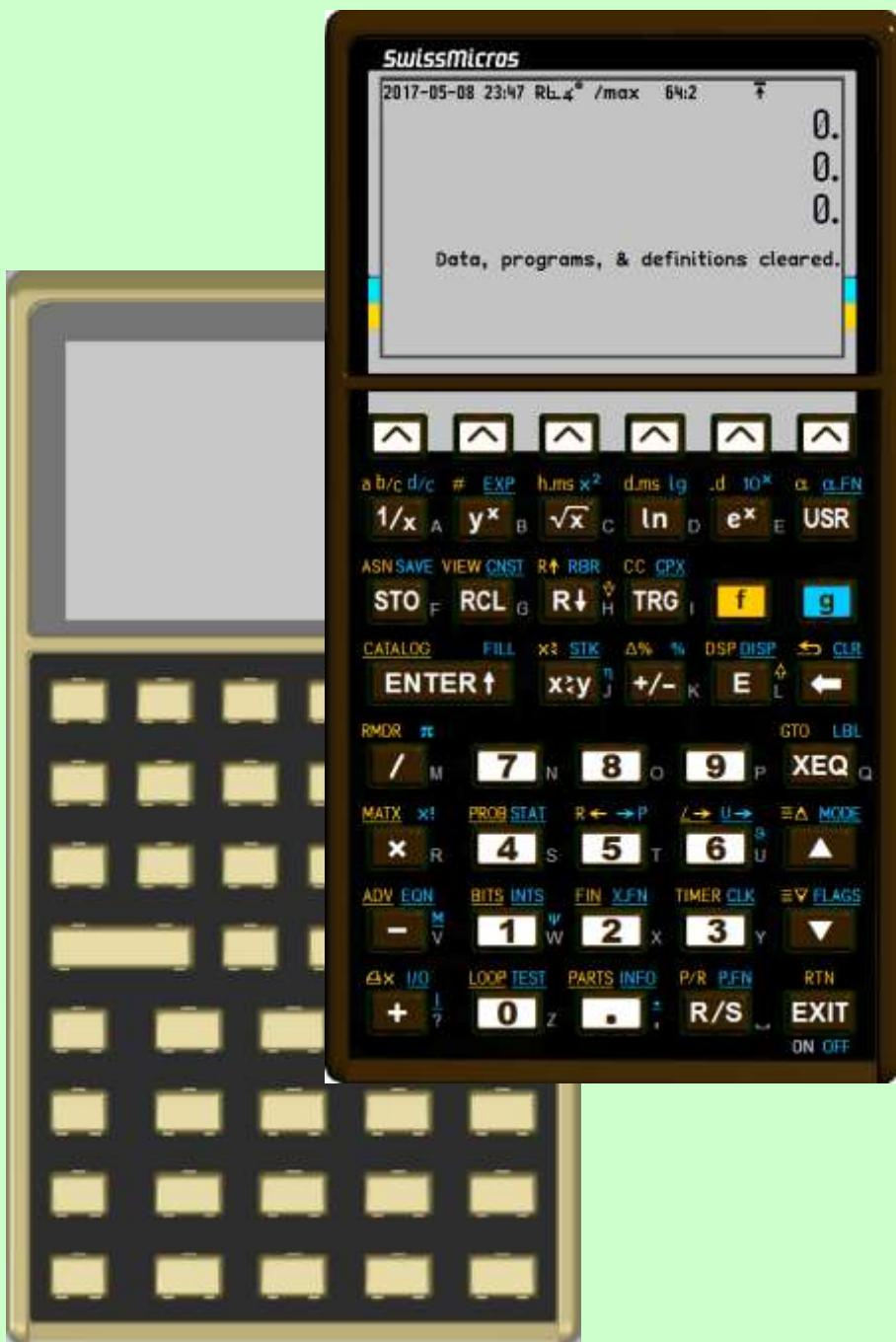
Keyboard overlays: Three short slots on either side of the keyboard are provided in the calculator housing for easy fixing overlay sheets with your personal layouts



printed on them. See App. F for more (on pp. 218f). Look up Section 6 of the OM for inspiration.

Seven pictures of the hardware are displayed here and on the three pages following. The default keyboard layout as delivered by SwissMicros for the DM42 (bottom right) and the front views on next page are printed approximately to scale. Find the printed circuit board (*PCB*) displayed thereafter.







See here the internals. 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 **A**, then unfasten the two Philips bolts **B** pictured left and below. **These operations are at your own risk.**



This picture shows the *PCB* of an early *DM42* of spring 2017.



Please use the following link to find a discussion (in February 2018) of the various hardware components used on the *DM42 PCB* as pictured on previous page: <https://www.hpmuseum.org/forum/thread-10143.html>.

## APPENDIX B: MEMORY MANAGEMENT

### Data Types

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

- 7-character 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  $\pm 2\ 147\ 483\ 648$  (i.e. 32 *bits*),
- *flag words* for storing 128 (i.e. 112 global plus 16 local) *user flags* and the same amount of *system flags*,<sup>53</sup>
- two *data types* for two kinds of *menus*,
- a *data type* of variable length for storing *configurations* (modes and user assignments, see STOCFG and RCLCFG),
- another one for *expressions* in EQN (see Section 4 of the OM),
- two more for program steps and routines (see Section 3 of the OM).

A 4-byte *header* is specified for each object of each *data type*:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pointer to the data															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pointer to the variable <i>name</i>				0	0	data information <sup>54</sup>					<i>data type</i> (as specified below) - 1				

<sup>53</sup> Up to 256 *flags* would be possible in total. We need far less *system flags* so far.

<sup>54</sup> E.g. base of a *short integer*, angular unit for an *angle*.

Data type number and meaning		Size [bytes]
1	$\mathbb{Z}$ Long integer <sup>55</sup>	$\geq 4 + 2 + 4 = 10\dots422$
2	$\mathbb{R}$ Real number <sup>56</sup> (real34)	$4 + 16 = 20$
3	$\mathbb{C}$ Complex number (complex34)	$4 + 2 \times 16 = 36$
4	Angle <sup>57</sup> (angle34)	$4 + 16 = 20$
5	Time <sup>58</sup>	$4 + 16 = 20$
6	Date <sup>59</sup>	$4 + 16 = 20$
7	Alpha string (each character requires 16 bits) <sup>60</sup>	$4 + 2 + n \times 2$
8	Real matrix (featuring $n$ rows and $m$ columns)	$4 + n \times m \times 16$
9	Complex matrix (featuring $n$ rows and $m$ columns)	$4 + n \times m \times 32$
10	Short integer <sup>61</sup>	$\leq 4 + 8 = 8 \text{ or } 12$
11 ... 13	n/a	n/a
14	Constant <sup>62</sup>	$4 + 0 = 4$

<sup>55</sup> This *data type* is for number theory kind of problems. 2 *bytes* are for the size (in *bits*) of the integer following. 4 *bytes* following allow for signed integers up to  $2^{31} \approx 2 \times 10^9$ , 8 *bytes* for  $2^{63} \approx 9 \times 10^{18}$ . Size is increased in steps of 4 *bytes* if required. Maximum integer size is 3328 *bits* (= 416 *bytes*) equivalent to 1001 decimal digits. Look up the display limits further below.

<sup>56</sup> Deviating from the WP 34S, standard *reals* on your WP 43S feature 128 *bits* and 34-digits precision. See the chapter after next.

<sup>57</sup> A tagged *angle* is stored as a *real* number, just with a specific header.

<sup>58</sup> A *time* or time interval 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. The format allows for expressing intervals of some 100 million years with *femtoseconds* precision.

<sup>59</sup> A *date* is stored as *real* number of *seconds* passed since -4713-01-01 12:00:00 (noon). This is date and time zero for *Julian Day Number* counting.

<sup>60</sup> 2 *bytes* are for the size (in *bytes*) of the string following, including the trailing zero. The size must be even.

<sup>61</sup> This *data type* is for computer science problems. Most probably, such a storage space will be either 4 + 4 or 4 + 8 *bytes* long.

<sup>62</sup> A pointer is sufficient here regardless of the precision of the constant itself.

Data type number and meaning		Size [bytes]
15	Extended precision <i>real</i> (39 digits, exclusively for internal use, not exposed to the user) <sup>63</sup>	$4 + 32 =$ 36
16	<i>Label</i> (each character requires 16 <i>bits</i> )	$4 + 7 \times 2 =$ 18
17	System integer (for internal use only)	$4 + 4 =$ 8
18	System and user <i>flags</i> (128 <i>flags</i> each)	$4 + 2 =$ 6
19	User-created <i>menu</i> (limited to 1 <i>view</i> )	$4 + 18 \times 7 \times 2 =$ 256
20	Predefined <i>menu</i> (featuring <i>n</i> <i>views</i> )	$4 + n \times 18 \times 14$
21	<i>Configuration</i> (as stored by STOCFG) <sup>64</sup>	$4 + M$
22	Program step, may be stored as <i>alpha string</i> (7) <sup>65</sup>	$4 + M$
23	Program, containing <i>n</i> program steps	$4 + M$
24	<i>Expression</i> (for members of <u>EQN</u> ), may be stored as <i>alpha string</i> (7)	$4 + M$
25	<i>Directory</i> (proposed by P. 2012-12)	$4 + M$

Data types 7 - 9 and 20ff 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.

---

<sup>63</sup> There are also even longer (i.e. more precise) *reals* used in internal computations.  
See further below.

<sup>64</sup> The size will vary according to the number of user assignments being part of the configuration (see Section 6 of the OM).

<sup>65</sup> The size will vary depending on parameters. Exact limits and methods are not decided yet.

## Statistical Summation Registers

Your *WP 43S* features a block of 23 special *registers* for storing statistical sums (like the 14 summation *registers* of the *WP 34S* and *WP 31S* before). These statistical *registers* neither overlap nor interfere with any general purpose *registers* unlike they did on *HP's* pocket calculators. Their contents can be recalled using their names (cf. pp. 61 and 96f).

And like on our *WP* 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 de-allocated and the memory is returned to the pool by  $\text{CL}\Sigma$ .

## Range of Real Numbers

Your *WP 43S* could calculate with *real* numbers of more than 12 000 orders of magnitude. Within a range of  $10^{-6143} \leq |x| < 10^{+6145}$ , it computes with 34 digits precision.<sup>66</sup> Results should be accurate within

---

<sup>66</sup> The *WP 43S* software is based on the *decNumber library* supporting arbitrary precision *BCD* numbers. As mentioned at some places in the *IOI*, internal computations are usually carried out with 39 digits. Actually this is the minimum; some modulo calculations for the trigonometric functions are performed with more than a thousand digits to avoid cancellation.

There is a quasi standard to find out about processors, firmware, and assess accuracy of calculators – compute  $\arcsin\{\arccos[\arctan(\tan\{\cos[\sin(9^\circ)]\})]\}$ . An ideal calculator with infinite internal precision would return exactly 9 without cheating. Real calculators (all computing with a finite number of digits) deviate for obvious reasons. Your *WP 43s* returns

- $8,999\,999\,999\,999\,999\,999\,999\,999\,999\,937\,535 = 9 - 6,246\,5 \cdot 10^{-29}$ .

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

Another – far simpler – test discussed in the internet works as follows: Enter 1,000 000 1 and then press  $x^2$  just 27 times. Your *WP 43s* will return

- 674 530,470 741 084 559 382 689 **184 727 772 2**.

This is the most precise result known of a pocket calculator so far (the *WP 34S* with DBLON and Free42 concur, computing with 34 digits as well). Nevertheless, only

$\pm 1 \times 10^{-33}$  (e.g. **n** **1/x** **2n** **x**) returns a plain 2 for all primes  $< 500$  – except **7** **1/x** **1** **4** **x** **2** **-** returns  $1 \times 10^{-33}$ , and for 67 and 83 the corresponding results are  $-1 \times 10^{-33}$ ). Note that also rounding errors due to calculating with a finite number of digits will accumulate as demonstrated by two examples in the footnote here.

Results  $|x| < 10^{-6176}$  are set to zero. For results  $|x| \geq 10^{6145}$ , error 4 or 5 will appear unless SPCRES is set (see App. C).

All these effects are caused by the **internal representation of reals**: Standard floating point numbers are stored on your WP 43S in sixteen bytes using an internal format coarsely following decimal128 packed coding,<sup>67</sup> though with some exceptions:

- 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 as pure integer in eleven groups of three digits. Each such group is packed into ten bits straight forward, meaning e.g.  $555_{10} = 10\ 0010\ 1100_2 = 22B_{16}$  or  $999_{10} = 11\ 1110\ 0111_2 = 3E7_{16}$ . So the 33 rightmost decimal digits of the *significand* take the least significant 110 bits. Trailing zeros are omitted, so the *significand* will be right adjusted.
- The most significant (128<sup>th</sup>) bit takes the sign of the mantissa.
- The remaining 17 bits are used for the exponent and the leftmost digit of the mantissa. Of those 17, the lowest 12 are reserved for the exponent ( $< 4096$ ). For the top 5 bits (below the sign bit) it

---

the first 25 digits of this output are correct! Calculating with unlimited precision returns

• 674 530,470 741 084 559 382 689 178 029 746 812 844 444 143 410 34

instead for the first 50 of the  $10^9$  digits of the complete result (note you get more than 1000 decimals after 8 presses of **x<sup>2</sup>** already).

Please take this information into account when assessing small deviations or many decimals returned by your WP 34S. It will return up to 34 digits but not all of them are guaranteed being true always.

<sup>67</sup> It comes close to what is called *quadruple (precision)* in this text about floating point arithmetic: <https://people.eecs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF>. Find out about decimal128 in [https://en.wikipedia.org/wiki/Decimal128\\_floating-point\\_format](https://en.wikipedia.org/wiki/Decimal128_floating-point_format).

becomes complicated – following the standard *IEEE 754*. If these 5 bits 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 represent the leftmost digit of the *significand*.  
If uu reads 00, 01, or 10 then these will represent the two most significant bits of the exponent. If it reads 11, there are bit patterns specified for encoding special numbers (see below).

Thus, the maximum absolute value of the stored exponent is  $10\ 1111\ 1111\ 1111_2 = 12\ 287_{10}$ . For reasons becoming obvious below, 6176 must be subtracted from the stored value to get the true exponent of the floating point number represented. Thus and since  $12287 - 6176 + 34 = 6145$  as well as  $-6176 + 34 = -6142$ , *data type 2* can support 34-digit numbers within  $10^{-6143} \leq |x| < 10^{+6145}$ .

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. *SE* stands for *stored exponent* in the following:

Floating point number	Hexadecimal value stored	Bottom bits in groups of 10	Top 18 bits in binary notation	SE
1.	22 08 00 00 00 00 00 00 00 00 00 00 00 00 00 01		0010 0010 0000 1000 00	6176
-1.	a2 08 00 00 00 00 00 00 00 00 00 00 00 00 00 01		1010 0010 0000 1000 00	6176
111.	22 08 00 00 00 00 00 00 00 00 00 00 00 00 00 6f	06f	0010 0010 0000 1000 00	6176

Floating point number	Hexadecimal value stored	Bottom bits in groups of 10	Top 18 bits in binary notation	SE
<b>111.111</b> $(111.111 \times 10^{-3})$	22 07 40 00 00 00 00 00 00 00 00 00 00 01 bc 6f	06f 06f	0010 0010 0000 0111 01	6173
<b>-123.000 123</b> $(-123.000 123 \times 10^{-6})$	a2 06 80 00 00 00 00 00 00 00 00 00 07 b0 00 7b	07b 000 07b	1010 0010 0000 0110 10	6170
<b>9.99x10<sup>99</sup></b> $(999 \times 10^{97})$	22 20 40 00 00 00 00 00 00 00 00 00 00 00 03 e7	3e7	0010 0010 0010 0000 01	6273
<b>1x10<sup>-99</sup></b>	21 ef 40 00 00 00 00 00 00 00 00 00 00 00 00 01		0010 0001 1110 1111 01	6077
<b>-1x10<sup>-6143</sup></b>	80 08 40 00 00 00 00 00 00 00 00 00 00 00 00 01		1000 0000 0000 1000 01	33

You will lose one digit precision if you divide  $10^{-6143}$  by 10 and one more for each such division following. At  $10^{-6176}$ , only one digit will be left, stored as hexadecimal 1.

Divide this by 1.999 999 999 999 999 999 999 999 999 999 and the result will remain  $10^{-6176}$  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 upper end of our numeric range now:

Floating point number	Hexadecimal value stored	Bottom bits in groups of 10	Top 18 bits in binary notation	SE
<b>9.999 999 999 999 999 999 999 999 999 999 999 x10<sup>6144</sup> (= 9 999 ... 999 999 x10<sup>6110</sup>)</b>	77 ff be 7f 9f e7 f9 fe 7f 9f e7 f9 fe 7f 9f e7	9 3e7 3e7 3e7 3e7 3e7 3e7 3e7 3e7 3e7 3e7 3e7	0111 0111 1111 1111 10	12 286

This *real* number (featuring 34 times the digit 9) is the maximum which can be keyed in directly. It will be displayed as  $\infty$  in startup default format – 9.999 999 999 999 999 999 999 999 999 999 999 999 999 999 999  $\times 10^{6144}$  will be only unveiled by SHOW (see p. 76). The greatest legal *significand* is  $9\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999 = 10^{34} - 1$ .

Additionally, your *WP 43S* features three ‘special reals’:

Floating point ‘number’	Hexadecimal value stored	Top 8 bits in binary notation
$\infty$	78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0111 1000
$-\infty$	F8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1111 1000
NaN	7C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0111 1100

Neither an exponent nor a mantissa is applicable here. **These three ‘special reals’ may be legal results of your *WP 43S* if SPCRES is set** – no error will be thrown then. NaN (i.e. ‘Not a Number’) covers poles as well as regions where a function result is not defined at all (see the corresponding entry in *Section 5* of the OM and examples in next chapter). Note that  **$\infty$  and  $-\infty$  may be also legal numeric inputs on your *WP 43S*.**

Remember not every 34-digit number displayed will be true to 34 digits – cf. footnote 66 on pp. 170f. And errors accumulate as explained in footnote 43 on p. 139.

As mentioned above, some calculations are executed in “*internal high precision*”. This means even more digits than 34: it may mean 39, 72, 75, or even more than 1000 digits in special cases.

 Rounding mode settings (see RM) may affect results of high precision calculations!

## Numeric Limitations

### Maximum numeric input for *data type* ...

- 1:  $\pm 10^{1001}$  (if your patience will suffice for the input) or  $\pm 2^{3326}$  (if you are less patient)
- 2:  $\pm 9.999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999 \times 10^{\text{RANGE}-1}$  (absolute maximum for **RANGE** is 6145)
- 3:  $\pm 9.999\dots \times 10^{\text{RANGE}-1}$  for either part
- 4:  $\pm 9.999\dots \times 10^{\text{RANGE}-1}$  in arbitrary *ADM*, so the actual absolute maximum is  $9.999\dots \times 10^{\text{RANGE}-1} \pi$
- 5: times xxx
- 6: dates xxx
- 8:  $\pm 9.999\dots \times 10^{\text{RANGE}-1}$  for each matrix element
- 9:  $\pm 9.999\dots \times 10^{\text{RANGE}-1}$  for either part of each matrix element
- 10: Absolute maximum is FF FF FF FF FF FF FF FF<sub>16</sub> in unsigned mode for *WSIZE 64*, equivalent to some  $1.8 \times 10^{19}$ . Signed modes and/or shorter word sizes mean lower limits.

### Internal limits:

- PRIME? works like the other binary tests described in the OM. Above 3 317 044 064 679 887 385 961 981 (i.e.  $3.3 \times 10^{24}$ ), however, NEXTP cannot find primes anymore but ‘probable primes’ only (cf. p. 64). If you want to apply NEXTP or PRIME? above this limit, consult appropriate reference literature.<sup>68</sup>
- Trigonometric functions actually operate between  $+\pi$  and  $-\pi$  (or their equivalents) exclusively. The necessary modulo  $2\pi$  reduction

---

<sup>68</sup> See [https://en.wikipedia.org/wiki/Miller%20-%20Rabin\\_primality\\_test](https://en.wikipedia.org/wiki/Miller%20-%20Rabin_primality_test) for a start (also available in other languages). Whatever is a prime will be confirmed by PRIME? – a composite may be falsely assessed as being prime with a probability of  $9 \times 10^{-16}$ . With some care, NEXTP can find ‘probable primes’ for long integers up to some  $6.3 \times 10^{1001}$  on your WP 43S, and PRIME? will check them and their neighbouring numbers but you cannot see most of their 1002 digits directly.

ensures that these functions return correct 34-digit results for the entire legal range of angles.

### Maximum numeric output for *data type* ...

- 1:  $\pm 10^{1001}$  with full 1001-digit precision (though you can see and read up to 296 digits only of such numbers (cf. SHOW, p. 76)).
- 2, 3, 8, and 9: The maxima are as specified for input above.  
Any (intermediate) result exceeding  $-10^{\text{RANGE}} < x < 10^{\text{RANGE}}$  will be displayed as  $-\infty$  or  $\infty$ , and any (intermediate) result within  $-10^{-\text{RANGE}} < x < 10^{-\text{RANGE}}$  will be displayed as '0.'.  
But any (intermediate) result within  $-10^{-6176} < x < 10^{-6176}$  will be assessed and displayed as 0. (cf. previous chapter). Any (intermediate) result exceeding  $-10^{6145} < x < 10^{6145}$  will be assessed as  $-\infty$  or  $\infty$ , respectively, and will then be treated according to the system settings at display time: if SPCRES is set, it will be shown as  $-\infty$  or  $\infty$ , else an overflow error will be thrown.
- 4: For angular conversions, the maxima are as specified for input above. The functions ARCSIN, ARCCOS, and ARCTAN return values between  $-\pi$  and  $\pi$  (or their equivalents) only.
- 5: xxx
- 6: xxx
- 10: The maxima are as specified for input above.

## Special Results

Within this chapter, SPCRES is presumed to be set. Thus, infinities and non-numeric results are legal – no error message will be thrown if such results happen to occur (cf. the end of previous chapter).<sup>69</sup>

<sup>69</sup> Results were crosschecked against WP 34S wherever possible. Additionally, Wolfram Alpha was used for checking results with finite arguments. Where deviations are observed we are confident your WP 43S returns the correct results.

The following monadic functions, if called with  $\mathbb{R}$  lit (i.e. CPXRES clear), return either  $\infty$ ,  $-\infty$ , or  $\text{NaN}$  under the conditions stated below:

Input $x$	Operation(s)	Output for $\mathbb{R}$ lit
-1.	$\text{artanh}$	
0 or 0.	$\text{In}$ , $\text{Ig}$ , $\text{lb } x$	$-\infty$
0.	$\frac{1}{x}$	
1.	$\text{artanh}$	$\infty$
0 or 0.	$\Gamma(x)$	
$\text{Re}(x) < 1$	$\text{arcosh}$	
$ \text{Re}(x)  > 1$	$\text{arccos}$ , $\text{arcsin}$ , $\text{artanh}$	$\text{NaN}$
$\pm 90^\circ$ or equivalents in other ADM	$\tan$	

And the following monadic functions operate also on infinities:

Input $x$	Operation(s)	Output for $\mathbb{R}$ lit
$-\infty$	$x^3$ , $\sqrt[3]{x}$	$-\infty$
	$\text{arctan}$	$-90^\circ$ or equivalents
	$\tanh$	-1.
	$\frac{1}{x}$ , $e^x$ , $10^x$ , $2^x$ , $\text{sinc}$	0.
	$x^2$ , $\text{arsinh}$	$\infty$
$-\infty$	$\text{arcosh}$	$\text{NaN}$
$-\infty \leq x < 0$	$\text{In}$ , $\text{Ig}$ , $\text{lb } x$	$\text{NaN}$
$\infty$	$\frac{1}{x}$ , $\text{sinc}$	0.
	$\tanh$	1.
	$\text{arctan}$	$90^\circ$ or equivalents

Input $x$	Operation(s)	Output for R lit
	$\ln$ , $e^x$ , $x^2$ , $\sqrt{x}$ , $\lg$ , $10^x$ , $\text{lb } x$ , $x^3$ , $\sqrt[3]{x}$ , $\sinh$ , $\cosh$ , $\text{arsinh}$ , $\text{arcosh}$	$\infty$
$-\infty$ or $\infty$	$\cos$ , $\sin$ , $\tan$ , $\text{artanh}$	NaN

For dyadic functions, we combined the respective tables:

Input $y$	$x$	Op.(s)	Output for R lit
$\infty$	$x \neq -\infty$	+	$\infty$ <sup>70</sup>
$-\infty$	$x \neq \infty$		$-\infty$ <sup>70</sup>
$-\infty$	$\infty$	+	NaN <sup>70</sup>
$\infty$	$x \neq \infty$		$\infty$ <sup>71</sup>
$-\infty$	$x \neq -\infty$	-	$-\infty$ <sup>71</sup>
$-\infty$	$-\infty$		NaN
$\infty$	$\infty$	$\times$	NaN
$\infty$	$x > 0$		$\infty$ <sup>70</sup>
$-\infty$	$x < 0$	$\times$	NaN
$\infty$	$x < 0$		$-\infty$ <sup>70</sup>
$-\infty$	$x > 0$	$\times$	NaN <sup>70</sup>
$0$ or $0.$	$-\infty$ or $\infty$		NaN
$0 < y \leq \infty$	$0.$	/	$\infty$
$-\infty \leq y < 0$			$-\infty$
$-\infty$ or $\infty$	$-\infty$ or $\infty$	/	NaN
$0$ or $0.$	$0.$	/, $y^x$	NaN
$-\infty$ or $\infty$	$0.$ or $0$	$y^x$	NaN

<sup>70</sup> Swapping  $x$  and  $y$  will return the same result here.

<sup>71</sup> Swapping  $x$  and  $y$  will return this result times -1.

Input	$y$	$x$	Op.(s)	Output for $\mathbb{R}$ lit
	$-\infty \leq y < 0$	$-\infty$ or $\infty$	$\boxed{x/y}$	$\text{NaN}$
	$-\infty < y < 0$	non-integer $x$	$\boxed{y^x}$ , $\boxed{x/y}$	$\text{NaN}$
	$-\infty$	odd $x > 0$		$-\infty$
	$-\infty$	even $x > 0$	$\boxed{y^x}$ $\boxed{x/y}$	$\infty$ $\text{NaN}$
$y \neq 0$	$-\infty$		$\boxed{y^x}$	$0.$
	$\infty$		$\boxed{y^x}$	$\infty$
0.		$-\infty \leq x < 0$	$\boxed{y^x}$	$\infty$
		$0 < x \leq \infty$	$\boxed{y^x}$	$0.$
		$-\infty < x < 0$	$\boxed{x/y}$	$\infty$
		$0 \leq x < \infty$	$\boxed{x/y}$	$0.$
$0 \leq y \leq \infty$	$-\infty$ or $\infty$		$\boxed{x/y}$	$1.$
	$\infty$		$\boxed{x/y}$	$0.$
$\infty$		$-\infty \leq x < 0$	$\boxed{y^x}$	$0.$
		$0 < x \leq \infty$	$\boxed{y^x}$	$\infty$
		$-\infty < x < 0$	$\boxed{x/y}$	$0.$
		$0 \leq x < \infty$	$\boxed{x/y}$	$\infty$
0.		$0 < x < \infty$	$\log_{x/y}$	$-\infty$

The functions printed on light yellow background in the three tables above will return  $\text{NaN}$  (or  $\text{NaN}+i\text{NaN}$ ) also with *complex* results allowed (i.e. CPXRES set). Others will change their output when  $\mathbb{C}$  is lit.

Some particular returns of elementary transient functions operating at the edge of the complex plane at  $\pm\infty$  or close to it (or returning a value at the edge) are listed here:

Input <sup>72</sup> Re( $x$ ) Im( $x$ )		r( $x$ )	$\varphi(x)$	Op.	Output for C lit
-∞	—	—			$\infty \not\in 90^\circ = 0 + i \times \infty$
-∞	0	∞	180°	$\sqrt{x}$	$\infty \not\in 360^\circ = \infty + i \times 0.$
-∞	0	∞	180°	$x^2$	$\infty \not\in 180^\circ = -\infty + i \times 0.$
0.	∞	∞	90°		
-∞	—	—			-∞
-∞		∞			$-\infty + i \times 0.$ <sup>73</sup>
-10 <sup>999</sup>	0	10 <sup>999</sup>	180°	$\sqrt[3]{x}$	$1 \times 10^{333} \not\in 60^\circ = 0.5 \times 10^{333} + i \times 0.866 025 404 \times 10^{333} = 5 \times 10^{332} (1 + i \times \sqrt{3})$
-∞	—	—		$x^3$	-∞
-∞	0	∞	180°		$-\infty + i \times 0.$
-∞	—	—			$\infty + i \times 3.141 592 65\dots = \infty + i\pi$
-∞	0	∞	180°		
-∞	∞	∞	135°		$\infty + i \times 2.356 194 49\dots = \infty + i^{3\pi/4}$
0.	∞	∞	90°		$\infty + i \times 1.570 796 32\dots = \infty + i^{\pi/2}$
∞	∞	∞	45°		$\infty + i \times 0.785 398 16\dots = \infty + i^{\pi/4}$
∞	—	—			∞
∞	0	∞	0°		$\infty + i \times 0.$
∞	-∞	∞	-45°		$\infty - i \times 0.785 398 16\dots = \infty - i^{\pi/4}$

<sup>72</sup> Complex infinities should be treated in polar notation (see an article by HP at <http://hparchive.com/Journals/HPJ-1984-07.pdf>, p. 27, left column for reasons).

<sup>73</sup> This result may be calculatory correct but is mathematically incorrect – compare next row. For mathematical reasons, similar cases may occur with other roots or powers operating near the edge of complex plane. Note complex numbers are stored in Cartesian coordinates.

Input <sup>72</sup>						Op.	Output for C lit
Re(x)	Im(x)	r(x)	$\varphi(x)$				
0.	- $\infty$	$\infty$	-90°				$\infty - i\pi/2$
- $\infty$	- $\infty$	$\infty$	-135°				$\infty - i^{3\pi/4}$
0.	0.	0.	0.			In	$-\infty + i \times 0.$
0.	—	—	—				$-\infty$
- $\infty$	0	$\infty$	180°			$e^x$	$0. + i \times 0.$
-10 <sup>999</sup>	10 <sup>999</sup>	10 <sup>999</sup>	135°				$0. + i \times 0.$
- $\infty$	$\infty$	$\infty$				$e^x$	NaN+i×NaN
0.	$\infty$	$\infty$	90°				NaN+i×NaN
$\infty$	$\infty$	$\infty$	45°			$e^x$	$\infty + i \times 0.$
$\infty$	0	$\infty$	0°				NaN+i×NaN
$\infty$	- $\infty$	$\infty$	-45°			$e^x$	NaN+i×NaN
0.	- $\infty$	$\infty$	-90°				$\infty - i\pi/2$
- $\infty$	- $\infty$	$\infty$	-135°			$e^x$	NaN+i×NaN
-10 <sup>999</sup>	-10 <sup>999</sup>	10 <sup>999</sup>					$0. + i \times 0.$

Computation of  $\text{lg}$  and  $\text{lb } x$  is derived from  $\text{In}$ . The same applies in analogy for  $e^x$ ,  $10^x$ , and  $2^x$ .

At the bottom line, we hope confusion is limited (and recommend keeping off  $\pm\infty$  in *complex plane*).

## Program Step Size

Program step size is assumed to be 4 *bytes* typically. But compare *data type 22* on p. 169.  $\text{xxx}$

## **APPENDIX C: MESSAGES AND ERROR CODES**

There are some commands generating *temporary information* (as specified in *Section 2* of the OM), e.g. CORR, DAY, ERR, L.R., MSG, s, VERS, WDAY,  $\bar{x}$ ,  $\hat{x}$ ,  $\hat{y}$ ,  $\Sigma+$ ,  $\Sigma-$ ,  $\sigma$ ,  $\rightarrow$ POL,  $\rightarrow$ REC, and the binary test commands.

Furthermore, there are a number of error messages issued by the operating system. Depending on conditions, the following messages will be displayed. They are listed below in alphabetical order (*EC* means *error code* here):

	EC	Explanations, countermeasures and examples
An argument exceeds the function domain	1	{1, 2, 3, 4, 10} 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 $0^0$ , $x/0$ , $0/0$ , $\Gamma(0)$ , $\tan(\pm 90^\circ)$ and equivalents, by $\text{artanh}(x)$ for $ \text{Re}(x)  \geq 1$ , by $\text{arcosh}(x)$ for $\text{Re}(x) < 1$ , etc. <sup>74</sup>
Bad time or date input	2	{2, 5, 6} Invalid date format or incorrect <i>date</i> 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	Predefined variables or menus cannot be deleted.
Distribution parameter out of valid range	16	{1, 2} A parameter specified in <b>I</b> , <b>J</b> , or <b>K</b> 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 <i>FM</i> to regain space.

<sup>74</sup> Note that e.g.  $\tan(90^\circ)$  and logs of 0 are legal operations on {1, 2, 3} if SPCRES is set. See the end of this appendix.

	<i>EC</i>	Explanations, countermeasures and examples
<b>Flash memory is write protected</b>	19	There was an attempt to edit or delete program steps in <i>FM</i> . See PRCL and PSTO to circumvent.
<b>Function to be coded for that data type</b>	30	Functions may not be coded yet during FW development.
<b>Illegal digit in integer input for this base</b>	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).
<b>Illegal input data type for this operation</b>	24	... called. Convert what is necessary. Cf. " <i>operation is undefined in this mode</i> ".
<b>Input data types do not match</b>	31	Attempt to operate on different data types (e.g. for a Boolean operation: real AND short integer).
<b>Input is too long</b>	10	{7} Keyboard input is too long for the buffer. Only alpha input is limited presently.
<b>Invalid or corrupted data</b>	18	Set when there is a checksum error either in <i>FM</i> or as part of a serial download. Also set if a <i>FM</i> segment is otherwise not usable.
<b>Item to be coded</b>	29	Functions may not be coded yet during FW development.
<b>I/O error</b>	17	See Section 3 of the OM.
<b>Matrix mismatch</b>	21	<p>{8, 9}</p> <ul style="list-style-type: none"> <li>• A matrix isn't square although it should be.</li> <li>• Matrix sizes aren't miscible.</li> </ul>
<b>No root found</b>	20	{2} The <i>Solver</i> did not converge.

	EC	Explanations, countermeasures and examples
No such function	7	Returned when calling a nonexistent function via <b>XEQ</b> <b>α</b> ... <b>ENTER↑</b> (check for typos!) or running a routine containing a nonprogrammable command.
No such label found	6	Attempt to address an undefined label.
No summation data present	28	Attempt to address an un-allocated summation register.
Operation is undefined in this mode	13	Caused e.g. by calling a <i>real-number</i> operation in AIM. Cf. “illegal input data type for this operation”.
Out of range	8	<p>{1, 2, 3, 10}</p> <ul style="list-style-type: none"> <li>A number exceeds the valid range. This can be caused by specifying decimals &gt; 16, <i>word size</i> &gt; 64, negative <i>flag</i> numbers, short integers <math>\geq 2^{64}</math>, <i>hours</i> or <i>degrees</i> &gt; 9 000, invalid <i>dates</i> or <i>times</i>, denominators <math>\geq 9\,999</math>, etc.</li> <li>A <i>register</i> or <i>flag</i> address exceeds the valid range of currently allocated <i>registers</i> or <i>flags</i>. May also happen in indirect addressing or when calling nonexistent local addresses.</li> <li>An R-operation (e.g. R-COPY) attempts accessing invalid <i>register</i> addresses.</li> </ul>
Output would exceed 196 characters	33	{7} Maximum length of alphanumeric strings.
Overflow at $+\infty$	4	<p>{1, 2, 3, 8, 9} unless SPCRES is set</p> <ul style="list-style-type: none"> <li>Division of a number <math>&gt; 0</math> by 0.</li> <li>Divergent sum or product or integral.</li> <li>Positive overflow (see p. 170).</li> </ul>

	<i>EC</i>	Explanations, countermeasures and examples
<b>Overflow at <math>-\infty</math></b>	5	{1, 2, 3, 8, 9} unless SPCRES is set <ul style="list-style-type: none"> <li>• Division of a number <math>&lt; 0</math> by 0.</li> <li>• Divergent sum or product or integral.</li> <li>• Negative overflow (see p. 170).</li> </ul>
<b>Please enter a NEW name</b>	26	Trying to define a new variable or user <i>menu</i> with a <i>name</i> already in use.
<b>RAM is full</b>	11	May be caused by attempts to write too large routines, allocate too many variables, and the like (see pp. 167ff for the space required by different <i>data types</i> ). May happen also in program execution due to dynamic allocations (see Sect. 3 of the OM).
<b>Singular matrix</b>	22	{8, 9} <ul style="list-style-type: none"> <li>• Attempt to use a LU decomposed matrix for solving a system of equations.</li> <li>• Attempt to invert a matrix which isn't of full rank.</li> </ul>
<b>Stack clash</b>	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 set and STOS 93 .
<b>This does not work with an empty string</b>	34	{7} Self-explanatory.
<b>This system flag is write protected</b>	32	Self-explanatory.
<b>Too few data points for this statistic</b>	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.

	<i>EC</i>	Explanations, countermeasures and examples
<b>Undefined op-code</b>	3	An instruction with an undefined operation code occurred. Should never happen – but who knows?
<b>Word size is too small</b>	14	{10} Input or <i>register</i> content is too great to be handled by the word size currently set.
	25	Left unused for <i>WP 34S</i> compatibility

If SPCRES 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 (cf. the corresponding entries in CONST on pp. 138ff and the tables on pp. 174ff). E.g., **0** **In** will return  $-\infty$  then.

Each error message will be displayed in **Z** numeric row and is *temporary information* (see *Section 2* of the OM). So **◀** or **EXIT** will erase it and allow continuation most easily. Any other key pressed will erase the message as well, but will also – if applicable – execute with the stack contents present.

DRY



## APPENDIX D: 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. 194, one for the *HP-21S* on p. 196, and another one for the *WP 34S* on p. 198. Functions newly introduced with *WP* calculators are compiled on pp. 202ff.

**Functional differences of homonymous commands are covered in the *IOI* (on pp. 13ff).**

### **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*.

<b><i>HP-42S</i></b>	<b><i>WP 43S</i></b>	<b>Remarks</b>
ACOSH	<b>arcosh</b>	In <u>EXP</u>
ADV	<b>��ADV</b>	In <u>PRINT</u>
AIP	Dispensable	You can merge text and numeric data easily as described in <i>Section 2</i> of the <i>OM</i> .
ALENG	<b>�ALENG</b>	In <u>�.FN</u>
ALL�	Dispensable	Your <i>WP 43S</i> runs in ALL� mode always. The summation <i>registers</i> do not overlap with general purpose <i>registers</i> .
<b>ALPHA</b>	<b>�</b>	See the description of A/M in <i>Sect. 2</i> of the <i>OM</i> .

HP-42S	WP 43S	Remarks
AOFF	CF ALPHA	
AON	SF ALPHA	
ARCL	Dispensable	Any register or variable can take an alpha string. Simply press <b>RCL</b> instead.
AROT	$\alpha RL$ or $\alpha RR$	In <u><math>\alpha.FN</math></u>
ASHF	$\alpha SL$	
ASINH	<b>arsinh</b>	In <u>EXP</u>
ASTO	Dispensable	Any register or variable can take an alpha string. Simply press <b>STO</b> instead.
ATANH	<b>artanh</b>	In <u>EXP</u>
ATOX	$\alpha \rightarrow x$	In <u><math>\alpha.FN</math></u>
AVIEW	Dispensable	Any register or variable can take an alpha string. Simply press <b>VIEW</b> instead.
<b>BASE</b>	<b>INTS</b> or <b>BITS</b>	
BASE+	Dispensable	Your WP 43S executes these arithmetic commands automatically for short integer inputs.
BASE-		
BASE $\times$		
BASE $\div$		
BASE $^{+/-}$		
BINM	Dispensable	Press <b># 2</b> for converting any closed integer number or integer part in $x$ to binary.
BIT?	<b>BS?</b>	In <u>BITS</u>
<b>BST</b>	 (  WP 43S R v0.15	

<b>HP-42S</b>	<b>WP 43S</b>	<b>Remarks</b>
CLV	See remark	Variables are cleared as specified in Section 6 of the OM.
<b>COMPLEX</b>	<b>CC</b>	You can also enter <i>complex</i> numbers directly using <b>CC</b> as explained in Section 2 of the OM.
<b>CONVERT</b>	<b>L<math>\leftrightarrow</math> &amp; PARTS</b>	
<b>CUSTOM</b>	n/a	You can create as many <i>menus</i> as memory will hold – not only one CUSTOM menu. See Section 6 of the OM.
DECM	Disposable	Any input featuring a <b>.</b> or an <b>E</b> is interpreted as a <i>real</i> (decimal) number.
DEL	n/a	Not featured. Too dangerous, in our opinion.
DELAY	<b>DLAY</b>	In <u>PRINT</u>
DELR	<b>M.DELR</b>	
DET	<b> M </b>	
DIM	<b>M.DIM</b>	
DIM?	<b>M.DIM?</b>	
EDIT	<b>M.EDI</b>	
EDITN	<b>M.EDIN</b>	
FCSTX	$\hat{x}$	
FCSTY	$\hat{y}$	In <u>STAT</u>
FNRM	<b>ENORM</b>	In <u>MATX</u> . Euclid is older than Frobenius.
GAMMA	$\Gamma(x)$	In <u>PROB</u>
GETKEY	<b>KEY?</b>	In <u>P.FN</u>
GETM	<b>M.GET</b>	
GROW	<b>M.GROW</b>	In <u>MATX</u>
HEXM	Disposable	Press <b># H</b> for converting any closed integer number or integer part in $x$ to hexadecimal.

<b>HP-42S</b>	<b>WP 43S</b>	<b>Remarks</b>
H.MS+	Dispensable	Your WP 43S executes the respective command automatically for sexagesimal times in $x$ and $y$ when $[+]$ or $[-]$ is pressed.
H.MS-		
INSR	<b>M.INSR</b>	In <u>MATX</u>
INTEG	$\int$	In <u>ADV</u>
INVRT	$[M]^{-1}$	In <u>MATX</u>
KEYASN	Dispensable	Not needed since no CUSTOM menu is featured (see CUSTOM).
<b>LASTx</b>	<b>RCL</b> <b>L</b>	
<b>LBL</b>		Press <b>LBL</b> .
LCLBL	Dispensable	Obsolete since no CUSTOM menu is featured (see CUSTOM). Nevertheless, your WP 43S provides local labels (see Section 3 of the OM).
LINΣ	Dispensable	Your WP 43S runs in ALLΣ mode always.
LIST	n/a	Use <b>PROG</b> instead.
LOG	<b>LOG<sub>10</sub></b>	Press <b>Ig</b> .
MAN	<b>CF</b> <b>T</b>	Manual print mode is <i>startup default</i> here.
MAT?	<b>MATR?</b>	In <u>TEST</u>
MEAN	$\bar{x}$	In <u>STAT</u>
<b>MOD</b>		Press <b>MOD</b> .
<b>MODES</b>	<b>MODE</b>	
N!	$x!$	
NEWMAT	<b>M.NEW</b>	In <u>MATX</u>
NORM	n/a	Not featured.
OCTM	Dispensable	Press <b>#</b> <b>8</b> for converting any closed integer number or integer part in $x$ to octal.
OLD	<b>M.OLD</b>	In <u>MATX</u>
ON	n/a	Programmable ON is not featured.
<b>PGM.FCN</b>	<b>P.FN</b>	<b>GTO</b> , <b>LBL</b> , <b>RTN</b> , <b>VIEW</b> are on the keyboard.

<b>HP-42S</b>	<b>WP 43S</b>	<b>Remarks</b>
PI	$\pi$	Press <b>T</b> .
POSA	$\alpha$ POS	In <u><math>\alpha</math>.FN</u>
PRA	<b>R</b> <b>K</b>	In <u>PRINT</u>
<b>PRGM</b>	<b>P/R</b>	
PRLCD	<b>LCD</b>	In <u>PRINT</u>
PROFF	<b>CF</b> <b>T</b>	
PROMPT	Disposable	Use <b>VIEW</b> , <b>STOP</b> instead.
PRON	<b>SF</b> <b>T</b>	
PRP	<b>PROG</b>	
PRSTK	<b>STK</b>	
PRUSR	<b>USER</b>	
PRV	<b>r</b>	
PRX	<b>r</b> <b>X</b>	Press <b>[x]</b> .
PR $\Sigma$	<b>Σ</b>	In <u>PRINT</u>
PUTM	<b>M.PUT</b>	In <u>MATX</u>
PWRF	<b>PowerF</b>	In <u>STAT</u>
RAN	<b>RAN#</b>	In <u>PROB</u>
RND	<b>ROUND</b>	In <u>PARTS</u>
RNRM	<b>RNORM</b>	In <u>MATX</u>
ROTXY	<b>RL</b> , <b>RLC</b> , <b>RR</b> , and <b>RRC</b>	In <u>BITS</u>
<b>RTN</b>		Press <b>RTN</b> .
SDEV	<b>s</b>	In <u>STAT</u>
SIZE	Disposable	There are 100 global general purpose <i>registers</i> always.
SLOPE	<b>L.R.</b>	In <u>STAT</u>
SOLVE	<b>SLV</b>	In <u>ADV</u>
SQRT	$\sqrt{x}$	

<b>HP-42S</b>	<b>WP 43S</b>	<b>Remarks</b>
<b>SST</b>	 (  )	Shortcut works if no <i>multi-view menu</i> is open.
STR?	<b>STRI?</b>	In <u>TEST</u>
<b>TOP.FCN</b>	Disposable	Obsolete – no top functions are overwritten.
TRACE	<b>SF</b> 	
TRANS	<b>[M]<sup>T</sup></b>	In <u>MATX</u>
UVEC	<b>UNITY</b>	In <u>MATX</u> and <u>CPX</u>
VARMENU	<b>VARMNU</b>	Truncated to 6 characters to fit the <i>menu</i> space.
<b>VIEW</b>		Press <b>VIEW</b> .
WMEAN	<b><math>\bar{x}_w</math></b>	In <u>STAT</u>
WRAP	<b>M.WRAP</b>	In <u>MATX</u>
XTOA	<b><math>x \rightarrow \alpha</math></b>	The conversion is done in <b>X</b> .
X<0?, X<Y?	<b>x&lt; ?</b>	In <u>TEST</u>
X≤0?, X≤Y?	<b>x≤ ?</b>	
X=0?, X=Y?	<b>x= ?</b>	
X≠0?, X≠Y?	<b>x≠ ?</b>	
X≥0?, X≥Y?	<b>x≥ ?</b>	
X>0?, X>Y?	<b>x&gt; ?</b>	
YINT	<b>L.R.</b>	In <u>STAT</u>
<b><math>y^x</math></b>		Press <b><math>y^x</math></b> .
<b>ΣREG</b>	Disposable	There are 100 global general purpose <i>registers</i> always. Statistical registers are separate.
<b>ΣREG?</b>		
→DEC	<b>→INT 10</b>	Press <b>#</b> 
<b>→HR</b>		Press <b>.d</b>
<b>→H.MS</b>		Press <b>h.ms</b> ... for closed input.
→OCT	<b>→INT 8</b>	Press <b>#</b> 
<b>→POL</b>		Press <b>→P</b> .
<b>→REC</b>		Press <b>R→</b> .

HP-42S	WP 43S	Remarks
%CH	Δ%	Press Δ%.
÷	/	Cf. ISO 80000-2: "The symbol ÷ should not be used."

## 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.

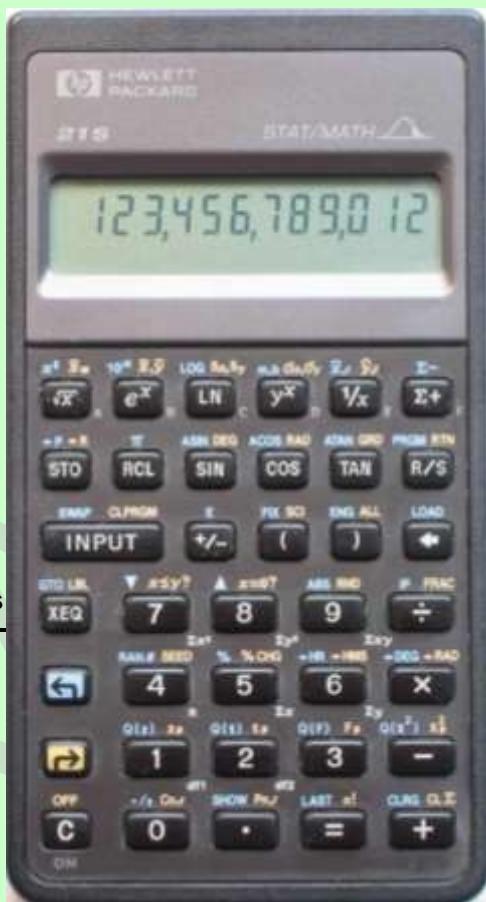
HP-16C	WP 43S	Remarks
RL , RLn	RL	
RR , RRn	RR	
RLC , RLCn	RLC	In BITS
RRC , RRCn	RRC	
÷	/	(see also ISO 80000-2: "The symbol ÷ should not be used.")
DBL÷	DBL/	In INTS
x <sup>≥(i)</sup> x <sup>≥I</sup>	Superfluous	Any register may be used for indirection.
SHOW HEX	n/a	
SHOW DEC		
SHOW OCT		
SHOW BIN		
B?	BS?	In BITS
GSB	XEQ	
HEX	# H	
DEC	# D	
OCT	# 8	
BIN	# 2	

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

## 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.



<b>HP-21S</b>	<b>WP 43S</b>	Remarks
<b>[<math>\bar{x}_w</math>]</b>	<b><math>\bar{x}_w</math></b>	
<b>[<math>\bar{x}, \bar{y}</math>]</b>	<b><math>\bar{x}</math></b>	
<b>[<math>S_x, S_y</math>]</b>	<b><math>s</math></b>	
<b>[m.b]</b>	<b>L.R.</b>	
<b>[<math>\sigma_x, \sigma_y</math>]</b>	<b><math>\sigma</math></b>	
<b>[<math>\hat{x}.r</math>]</b>	<b><math>r, \hat{x}</math></b>	
<b>[<math>\hat{y}.r</math>]</b>	<b><math>r, \hat{y}</math></b>	
<b>[PRGM]</b>	<b>P/R</b>	
<b>[SWAP]</b>	<b><math>x \leftrightarrow y</math></b>	
<b>[CLPRGM]</b>	<b>CLP</b>	In <u>CLR</u>
<b>[INPUT]</b>	Disposable	This functionality is contained in ENTER. Also your <i>WP 43S</i> features a command called INPUT but this works in programs.
<b>(</b> , <b>)</b>	Disposable	You can forget these keys in RPN.

<b>HP-21S</b>	<b>WP 43S</b>	<b>Remarks</b>
<b>LOAD</b>	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.
<b>ABS</b>	<b> x </b>	In <u>PARTS</u>
<b>RND</b>	<b>ROUND</b>	
<b>FRAC</b>	<b>FP</b>	
<b>÷</b>	<b>/</b>	Cf. ISO 80000-2: “The symbol $\div$ should not be used.”
<b>SEED</b>	<b>SEED</b>	In <u>PROB</u>
<b>%CHG</b>	<b>Δ %</b>	
<b>Q(z)</b>	<b>Norml<sub>e</sub></b>	In <u>submenus of PROB</u> .  Note your <i>WP 43S</i> features the normal distribution for arbitrary $\mu$ and $\sigma$ instead of the standardized one ( $\mu=0$ , $\sigma=1$ ). And 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.
<b>zp</b>	<b>Norml<sup>-1</sup></b>	
<b>Q(t)</b>	<b>t<sub>e</sub>(x)</b>	
<b>tp</b>	<b>t<sup>-1</sup>(p)</b>	
<b>Q(F)</b>	<b>F<sub>e</sub>(x)</b>	
<b>Fp</b>	<b>F<sup>-1</sup>(p)</b>	
<b>Q(x<sup>2</sup>)</b>	<b>χ<sup>2</sup>e(x)</b>	
<b>(x<sup>2</sup>)p</b>	<b>(χ<sup>2</sup>)<sup>-1</sup></b>	
<b>Cn.r</b>	<b>COMB</b>	In <u>PROB</u>
<b>Pn.r</b>	<b>PERM</b>	
<b>LAST</b>	<b>RCL L</b>	
<b>=</b>	<b>Dispensable</b>	You can forget this key in <i>RPN</i> .
<b>n!</b>	<b>x!</b>	
<b>CLRG</b>	<b>CLREGS</b>	In <u>CLR</u>

## Corresponding Operations on WP 34S

The WP 34S and 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. Thus, your WP 43S allows for softkeys – the WP 34S can only carry four hotkeys instead. Also dealing with matrices is greatly eased by the large high resolution dot matrix display of your WP 43S; thus some elementary matrix commands of the WP 34S are not required anymore on 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	Binom <sub>▲</sub>	
Binom <sub>u</sub>	Binom <sub>▲</sub>	
Binom	Binom <sub>▲</sub>	
Cauch <sub>u</sub>	Cauch <sub>▲</sub>	
CL <sub>a</sub>	0 [STO] K	Check the OM for the conditions when this register is used.
CONST	CNST	For keyboard space reasons.
CONV	U <sub>↔</sub>	
DBLOFF	Dispensable	Your WP 43S features 34-digit data types per default – it does neither need nor feature a double precision mode.
DBLON		
Expon	Expon <sub>▲</sub>	
Expon <sub>u</sub>	Expon <sub>▲</sub>	
F(x)	F <sub>▲</sub> (x)	
F <sub>u</sub> (x)	F <sub>▲</sub> (x)	
dRCL	Dispensable	Your WP 43S features various data types.

<b>WP 34S</b>	<b>WP 43S</b>	<b>Remarks</b>
gCLR, gDIM, gDIM?, gFLP, gPIX?, gPLOT, gSET	n/a	The <i>LCD</i> of your <i>WP 43S</i> features $240 \times 400$ px rows compared to $6 \times 43$ px of <i>HP-30b</i> – the graphic paradigm of <i>WP 34S</i> makes no sense on your <i>WP 43S</i> . On the other hand, it was not our objective designing a graphing calculator. Thus, we include just the basic graphic support of <i>HP-42S</i> (AGRAPH, CLLCD, PIXEL) plus POINT.
Geom	<b>Geom</b> 	
Geom <sub>u</sub>	<b>Geom</b> 	
GTO <sub>a</sub>	Dispensable	Use <b>GTO</b> with an appropriate parameter instead.
H.MS+, H.MS-	Dispensable	Your <i>WP 43S</i> features a dedicated <i>data type</i> for <i>times</i> , so <b>+</b> and <b>-</b> suffice for adding or subtracting sexagesimal times, respectively.
INTM?	Dispensable	Your <i>WP 43S</i> features dedicated <i>data types</i> for integers – it does neither need nor feature an integer mode.
iRCL	Dispensable	Your <i>WP 43S</i> features various <i>data types</i> .
I <sub>x</sub>	<b>I<sub>xyz</sub></b> 	This is a triadic function after all.
Lgnrm	<b>LgNrm</b> 	
Lgnrm <sub>u</sub>	<b>LgNrm</b> 	
L <sub>n</sub>	<b>L<sub>m</sub></b>	Renamed to avoid search conflict with LN.
L <sub>na</sub>	<b>L<sub>ma</sub></b>	Renamed in consequence to L <sub>m</sub> .
Logis	<b>Logis</b> 	
Logis <sub>u</sub>	<b>Logis</b> 	
MROW+ <sub>x</sub> , MROW <sub>x</sub>	Dispensable	Obsolete matrix commands.
MROW <sub>⤔</sub>	<b>M.R⤔R</b>	
M+ <sub>x</sub>	Dispensable	Obsolete matrix command.
M <sup>-1</sup>	<b>[M]<sup>-1</sup></b>	

<b>WP 34S</b>	<b>WP 43S</b>	<b>Remarks</b>
M-ALL, M-COL, M-DIAG, M-ROW	Disposable	Obsolete matrix commands.
Mx	Disposable	Your WP 43S features two dedicated <i>data types</i> for matrices. Thus you can simply multiply two matrices using <b>X</b> and copy matrices like any other objects.
M.COPY		
M.IJ, M.REG	Disposable	Obsolete matrix commands.
nBITS	#B	
nCOL, nROW	Disposable	Obsolete matrix commands.
Norml	Norml <sub>▲</sub>	
Norml <sub>u</sub>	Norml <sub>▲</sub>	
Poiss	Poiss <sub>▲</sub>	
Poiss <sub>u</sub>	Poiss <sub>▲</sub>	
REALM?	Disposable	Your WP 43S features a dedicated <i>data type</i> for <i>reals</i> – it does not need a <i>real mode</i> .
REGS, REGS?	Disposable	The number of global general purpose <i>registers</i> is fixed to 100 on your WP 43S.
SENDA, SENDP, SENDR, SENDΣ	SEND	SEND combines all those four commands of the WP 34S.
SEPOFF, SEPON	GAP	
SHOW	RBR	
sRCL	Disposable	Your WP 43S features various <i>data types</i> .
TRANSP	[M] <sup>T</sup>	
TSOFF	GAP 0	
TSON	GAP 3	

WP 34S	WP 43S	Remarks
$t(x)$	$t_{\Delta}(x)$	
$t_u(x)$	$t_{\Delta}(x)$	
VIEW $\alpha$ , VIEW $\alpha+$	Dispensable	Use <b>VIEW</b> instead; <i>alpha strings</i> are just another <i>data type</i> . Combine text and numeric data easily using <b>+</b> as shown in the OM, Sect. 2.
Weibl	Weibl <sub><math>\Delta</math></sub>	
Weiblu	Weibl <sub><math>\Delta</math></sub>	
XEQ $\alpha$	Dispensable	Use <b>XEQ</b> with an appropriate parameter instead.
XTAL?	Dispensable	A quartz crystal is installed by default.
YDOFF, YDON	Dispensable	Your WP 43S displays <i>y</i> whenever possible and wanted. See DSTACK.
$\alpha$ DATE, $\alpha$ DAY	Dispensable	You can combine text and numeric data easily using <b>+</b> as shown in Section 2 of the OM.
$\alpha$ GTO	Dispensable	Use <b>GTO</b> w/ an appropriate parameter instead.
$\alpha$ IP, $\alpha$ MONTH	Dispensable	You can combine text and numeric data easily using <b>+</b> as shown in Section 2 of the OM.
$\alpha$ RCL, $\alpha$ RC#	Dispensable	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 <b>+</b> .
$\alpha$ STO	Dispensable	Simply press <b>STO</b> instead (any <i>register</i> can take an <i>alpha string</i> ).
$\alpha$ TIME	Dispensable	See $\alpha$ DATE.
$\alpha$ XEQ	Dispensable	Use <b>XEQ</b> with an appropriate parameter instead.
$\beta$	$\beta(x,y)$	
$\Gamma$	$\Gamma(x)$	
$\Delta$ DAYS	Dispensable	Simply subtract two <i>dates</i> .
$\zeta$	$\zeta(x)$	

<b>WP 34S</b>	<b>WP 43S</b>	<b>Remarks</b>
$\Phi(x) \dots$	Disposable	Use NORML... with $\mu=0$ and $\sigma=1$ instead.
$\chi^2(x)$	$\chi^2_{\Delta}(x)$	
$\chi^2_u(x)$	$\chi^2_{\Delta}(x)$	
$\rightarrow H$	$\rightarrow HR$	
$\blacksquare PLOT$	n/a	See gCLR.
$\blacksquare C r_{XY}$	Disposable	Use $\blacksquare r$ instead.
$\blacksquare a,$ $\blacksquare a+,$ $\blacksquare +a$	Disposable	Combine text and numeric data easily using $\blacksquare +$ as shown in Section 2 of the OM. Then use $\blacksquare r$ .
$\blacksquare ?$	Disposable	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, not counting the unit conversions and constants provided;

55 of them are even new or extended compared to earlier WP calculators. The commands are printed below as spelled on your WP 43S.

<b>Command</b>	<b>WP 43S</b>	<b>WP 31S</b>	<b>WP 34S</b>
<b>2<sup>x</sup> AGM</b>	●	—	new
<b>ALL</b>	●	●	extended
<b>AND ASR NOT OR XOR</b>	●	—	extended
<b>BACK CASE SKIP</b>	●	—	new
<b>BATT?</b>	●	●	new

Command	WP 43S	WP 31S	WP 34S
BC? FB	●	—	new
BestF	extended	●	●
BestF?	new	—	—
Binom <sub>p</sub> Binom <sub>A</sub> (of Binomial distribution)	●	●	new
B <sub>n</sub> B <sub>n</sub> * CEIL FLOOR	●	—	new
Cauch <sub>p</sub> Cauch <sub>A</sub> Cauch <sub>A</sub> Cauch <sup>-1</sup>	●	●	new
CauchF GaussF HypF ParabF RootF	new	—	—
CLCVAR	new	—	—
CLFall CLPall CONJ CONVG? COV	●	—	new
CX→RE RE→CX	new	—	—
DATE TIME	●	—	(●)
DATE→ DAY MONTH YEAR →DATE	●	—	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	—	—
ENTRY?	●	—	new
EQ.DEL EQ.EDI EQ.NEW	new	—	—
erf erfc ERR MSG	●	—	new
EVEN? ODD?	●	—	new
Expon <sub>p</sub> Expon <sub>A</sub> Expon <sub>A</sub> Expon <sup>-1</sup>	●	●	new
EXPT MANT	●	—	new
FBR	new	—	—
FC?F FC?S FF FS?F FS?S	●	—	new
FIB	●	—	new

Command	WP 43S	WP 31S	WP 34S
FILL	●	●	new
FLASH? FP?	●	—	new
$F_p(x)$ $F_{\Delta}(x)$ (of $F$ distribution)	●	●	new
$f' f''$	new	—	—
$f'(x) f''(x)$	extended	—	new
GAP	extended	●	new
GCD LCM	●	●	new
$g_d g_d^{-1}$	●	—	new
Geom <sub>p</sub> Geom <sub>Δ</sub> Geom <sub>Δ</sub> Geom <sup>-1</sup>	●	—	new
H <sub>n</sub> H <sub>nP</sub> L <sub>m</sub> L <sub>ma</sub> P <sub>n</sub> T <sub>n</sub> U <sub>n</sub>	●	—	new
Hyper <sub>p</sub> Hyper <sub>Δ</sub> Hyper <sub>Δ</sub> Hyper <sup>-1</sup>	new	—	—
IDIV	●	—	new
IDIVR IM RE	new	—	—
INT? I <sub>xyz</sub> $\Gamma_p$ $\Gamma_q$	●	—	new
J <sub>y</sub> (x)	new	—	—
J/G	extended	●	new
KEY? KTyp? LBL? LEAP?	●	—	new
LgNrm <sub>p</sub> LgNrm <sub>Δ</sub> LgNrm <sub>Δ</sub> LgNrm <sup>-1</sup>	●	—	new
LN $\beta$ LN $\Gamma$ LOADP LOADR LOADSS LOADΣ LocR LocR? LOG <sub>2</sub> LOG <sub>xy</sub>	●	—	new
LOAD SAVE	●	●	new
Logis <sub>p</sub> Logis <sub>Δ</sub> Logis <sub>Δ</sub> Logis <sup>-1</sup>	●	●	new
max min MIRROR	●	—	new
MOD	●	●	new
MUL $\pi$ MUL $\pi$ →	new	—	—
M.LU M.SQR? NAND NaN? NEIGHB NOR	●	—	new
NBin <sub>p</sub> NBin <sub>Δ</sub> NBin <sub>Δ</sub> NBin <sup>-1</sup>	new	—	—
NEXTP PRIME?	●	●	new

Command	WP 43S	WP 31S	WP 34S
Norml <sub>p</sub> Norml <sub>A</sub> Norml <sub>A</sub> Norml <sub>-1</sub>	●	●	new
nΣ (callable by name)	●	●	new
OrthoF PLOT POINT	new	—	—
PAUSE	●	—	extended
Poiss <sub>p</sub> Poiss <sub>A</sub> Poiss <sub>A</sub> Poiss <sub>-1</sub>	●	●	new
PopLR PRCL PSTO PUTK	●	—	new
RANGE RANGE?	new	—	—
RBR	●	●	new
RCLCFG STOCFG	extended	—	new
RCLS STOS	●	—	new
RCL↑ RCL↓ STO↑ STO↓	●	—	new
RDP RECV SEND	●	—	new
Re <sup>z</sup> Im	new	—	—
RJ	●	—	new
RL RLC RR RRC	●	—	extended
RMD	●	●	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 <sub>m</sub> S <sub>mw</sub> S <sub>w</sub>	●	●	new
SNAP	new	—	—
SSIZE?	●	●	new
STATUS	extended	—	extended

Command	WP 43S	WP 31S	WP 34S
$s_{xy}$	●	—	new
s(a) TDISP	new	—	—
TICKS	●	—	new
TIMER	●	—	(●)
TOP? ULP?	●	—	new
$t_p(x)$ $t_{\Delta}(x)$ (of <i>t distribution</i> )	●	●	new
$t \gtrless y \gtrless z \gtrless \xi$	●	—	new
(undo)	●	new	—
V <sub>4</sub>	new	—	—
VERS? WDAY WHO?	●	●	new
Weibl <sub>p</sub> Weibl <sub>a</sub> Weibl <sub>b</sub> Weibl <sup>-1</sup>	●	●	new
$W_m$ $W_p$ $W^{-1}$ WSIZE? $\bar{x}_G$ XNOR	●	—	new
$\bar{x}_G$	●	●	new
$\bar{x}_H$ $\bar{x}_{RMS}$ x→DATE	new	—	—
$x < ?$ $x \leq ?$ $x = ?$ $x \neq ?$ $x \geq ?$ $x > ?$	extended	—	extended
$x = +0?$ $x = -0?$ $x \approx ?$	●	—	new
y <sup>x</sup>	extended	●	●
Y.MD	●	●	new
αLENG?	extended	—	●
αPOS?	extended	—	—
αRL αRR αSL αSR	●	—	extended
$\beta(x,y)$ $\Gamma_{xy}$ $\gamma_{xy}$ $\epsilon$ $\epsilon_m$ $\epsilon_b$ $\zeta(x)$ $\Pi$ $\Sigma$ $\sigma_w$	●	—	new
$\Sigma^1/x$ $\Sigma^{1/x^2}$ $\Sigma^1/y$ $\Sigma^1/y^2$ $\Sigma \ln y/x$ $\Sigma x^2/y$ $\Sigma x^3$ $\Sigma x^4$ $\Sigma x/y$	new	—	—
$\Sigma \ln^2 x$ $\Sigma \ln^2 y$ $\Sigma \ln x$ $\Sigma \ln xy$ $\Sigma \ln y$ $\Sigma x$ $\Sigma x^2$ $\Sigma x^2 y$ $\Sigma x \ln y$ $\Sigma xy$ $\Sigma y$ $\Sigma y \ln x$ $\Sigma y^2$ (callable by names)	●	●	new
$\chi^2_p(x)$ $\chi^2_{\Delta}(x)$ (of chi-square distribut.)	●	●	new
$(-1)^x$ ×MOD ^MOD	●	—	new
±∞?	new	—	—

Command	WP 43S	WP 31S	WP 34S
→DEG →RAD	●	●	new
→D.MS →MUL. $\pi$	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 (please see its manual).

## 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. All the manuals listed below are entirely contained in a document set distributed by the *Museum of HPC* (see <http://www.hpmuseum.org/cd/cddesc.htm>). They can be also found in the internet, some even provided by HP still.

Topic	Recommended literature
General calculation examples and applications	All vintage HP calculator manuals can be recommended

Topic	Recommended literature
Root finding and numeric integration	<i>HP-34C Owner's Handbook and Programming Guide</i> <sup>75</sup> <i>HP-15C Owner's Handbook</i> <sup>76</sup> <i>HP-15C Advanced Functions Handbook</i> <sup>77</sup> <i>HP-42S Programming Examples &amp; Techniques</i> <sup>78</sup>
Accuracy of numerical calculations	<i>HP-15C Advanced Functions Handbook</i> , pp. 172 – 211. <sup>77</sup>
Statistical distributions and their application	<i>HP-21S Owner's Manual</i> <sup>79</sup>
Financial calculations	<i>HP-17BII+ User's Guide</i> <sup>80</sup>
Manipulating bits and short integers	<i>HP-16C Owner's Handbook</i> <sup>81</sup>
Programming	<i>HP-42S Owner's Manual</i> <sup>82</sup> <i>HP-42S Programming Examples &amp; Techniques</i> <sup>78</sup>

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

<sup>75</sup> Read here: <https://www.yumpu.com/en/document/read/19323790/hp34c-slide-rule-museum>

<sup>76</sup> Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c03030589.pdf>

<sup>77</sup> Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c03308725.pdf>

<sup>78</sup> Download from <http://www.hp41.net/forum/fileshp41net/hp42s-programming-examples.pdf>

<sup>79</sup> Download from <https://www.manualslib.com/manual/940232/Hp-Hp-21s.html#manual>

<sup>80</sup> Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c00363348.pdf>

<sup>81</sup> Download from <http://www.hp41.net/forum/fileshp41net/hp16c.pdf>

<sup>82</sup> Download from <http://www.hp41.net/forum/fileshp41net/manuel-hp42s-us.pdf>

you compute (and check footnote 90 on p. 233 below as well as the last paragraph on p. 16 of the OM).

Note that the floating point standard *IEEE 754* was developed in 1985, i.e. after most of the pocket calculators mentioned above were launched already (see e.g. [https://en.wikipedia.org/wiki/Floating-point\\_arithmetic](https://en.wikipedia.org/wiki/Floating-point_arithmetic) as a starter, also about floating point numbers in general).

DRAFT

## **APPENDIX E: EMULATING A WP 43S ON YOUR COMPUTER**

### **1) Under Windows, you can ...**

**1.a)** use **MSYS2 MinGW 64-bit**, a runtime environment for *gcc*. You get it here <https://www.msys2.org>. Install it following the on-site instructions until beginning of step 6. Then enter in the black window:

```
pacman -S mingw-w64-x86_64-gcc git base-devel  
mingw-w64-x86_64-gtk3
```

**cd wp43s** for changing to the proper directory.

**git pull** for pulling the files from *gitlab* repository.

**make** for building *wp43s.exe*.

**rm backup.bin** for starting with the simulator reset to default.

**./wp43s** for starting the simulator. Continue with c).

**1.b)** alternatively do the following:

Open the folder

[https://gitlab.com/Over\\_score/wp43s/tree/master/windows%20binaries](https://gitlab.com/Over_score/wp43s/tree/master/windows%20binaries) .

Open *README.md* and proceed as described therein.

Eventually run *wp43s.exe*. Proceed to c).

### **2) Under Linux:**

You have to install the standard development tools, git, gtk+ 3 dev, and libgmp dev packages.

Then, the first time, simply run:

```
git clone https://gitlab.com/Over_score/wp43s.git
```

**cd wp43s** for changing to the proper directory.

- make** for building a new executable.
- ./wp43s** for starting the simulator. Continue with c).

### 3) Updating the simulator:

The following works under *Windows* (within the *MinGW* window) and *Linux*:

- cd wp43s** only if necessary under *Linux*.
- git pull** for pulling the changed files from gitlab repository.<sup>83</sup>
- make** (or **make rebuild** under *Linux*)<sup>84</sup>
- ./wp43s** for starting the simulator. Continue with c).

- c) The simulator will open (like one of the pictures below though larger).

---

<sup>83</sup> Sometimes, this step may terminate with an error due to conflicting local changes. The message reads “Please commit or stash your changes before you merge” (or a bad translation into your language). Then enter **git reset --hard** and try again thereafter.

<sup>84</sup> There may be files updated by **git pull** but no new build possible sometimes. Then **make** will throw a corresponding message.

There may be also other obstacles; then **make mrproper** will clean the field for a subsequent **make** (this cannot, however, overcome real errors in the software).

Operate the simulator with the mouse. The ten digits as well as **.**, **ENTER**, **+**, **-**, **x**, and **/** may also be entered via the numeric keypad directly, **▲** and **▼** via the cursor keys, and **←** via [**Backspace**]. Further computer keyboard shortcuts to simulator keys are presented overleaf.



(The landscape window will open if screen resolution does not suffice for portrait display; the lower picture shows an outdated keyboard layout.)



Pressing ...

- ... **h** copies the entire simulator screen image to the clipboard.
- ... **x** copies the full content of **X** to the clipboard.
- ... **z** copies the full contents of all 12 lettered *registers* thereto.
- ... **Z** copies the full contents of all 112 global *registers* thereto.

Current content of *register L* is shown top left in the simulator window. Instead of the low-battery indicator **■** making no sense on a PC application, 'SL' is displayed far right in the *status bar* whenever *automatic stack lift* is enabled (cf. Section 1 of the OM).

---

<sup>85</sup> Capitals are printed green here for better differentiation.

## **APPENDIX F: FLASHING AND UPDATING YOUR WP 43S**

There are two ways to get your *WP 43S*, in principle:

1. You can flash an existing *DM42* or
2. you can buy a *WP 43S* off the shelf.

Way 1 allows you to repurpose a *DM42* you own already, so you may save costs – on the other hand, you will have to live with stickers on the keys then; this way is explained in next chapter. The chapter thereafter (starting on p. 218) describes how to update your existing *WP 43S* when a new firmware becomes available.

### **How to Flash Your *WP 43S***

#### **1) Under Windows:**

- a) Start your computer. Take your *DM42* and turn it on. Then press



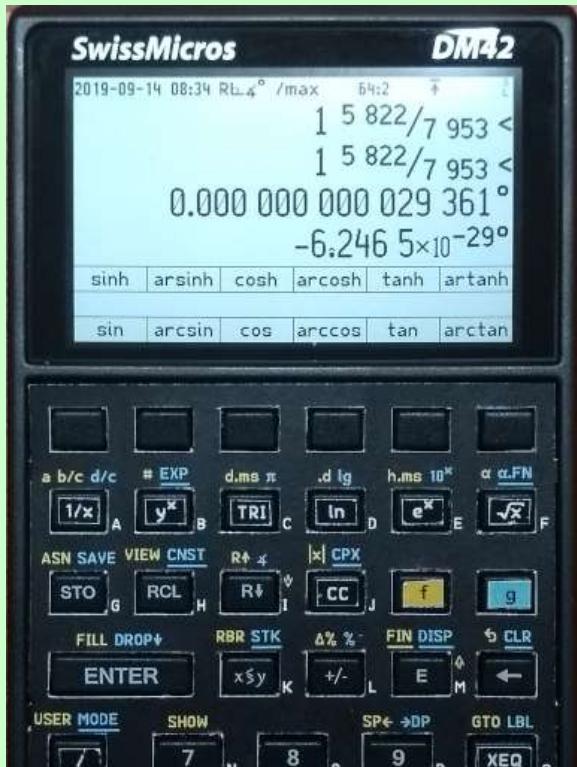
- [5]** System
- [2]** Enter system menu
- [4]** Reset to *DMCP* menu

Now connect your computer to the calculator *USB* socket using a proper data transfer cable. The flash disk of your *DM42* should show up as an external mass storage volume after you pressed ...

- [6]** Activate *USB* disk

- b) Start the internet browser on your computer and go to [https://gitlab.com/Over\\_score/wp43s/tree/master/DM42 binary](https://gitlab.com/Over_score/wp43s/tree/master/DM42%20binary).

Copy *WP43S.pgm* to the *DM42* flash disk. (Option: Copy also *keymap.bin* to the *DM42* flash disk. This will reassign keys to



match the WP 43S layout also when leaving WP 43S. Unless you have done this, EXIT/ON remains bottom left.)

c) Press  **SETUP**

**5**

**2**

**4**

**3**

WP43S.pgm **ENTER**

**ENTER**. Wait for flashing completed (~15 s).

Then press  **EXIT**  **EXIT**

**1**  **EXIT**. Now, your WP 43S is up and waiting for your commands.

The two files supplied (Key\_stickers.xcf, WP43S\_overlay.xcf) are GIMP images to ease your life as long as you rely on a converted DM42 to get your WP 43S. Print them, cut, and apply (see above picture of an earlier WP 43S layout).

To leave the WP 43S program, enter  **g MODE g SYSTEM** to return to the DMCP system. If you chose the option above, the key assignments will stay as they were in WP 43S when navigating therein (due to keymap.bin).

To retrieve the original DM42 keyboard layout (cf. p. 163), copy the file original\_DM42\_keymap.bin to the DM42 flash disk, rename it keymap.bin and RESET the DM42. Look here for more information: [http://www.swissmicros.com/dm42-devel/dmcp\\_devel\\_manual/](http://www.swissmicros.com/dm42-devel/dmcp_devel_manual/)

## 2) Under OSX (Mac):

Here is *Harald Overbeek's* step by step solution if you want the *WP 43S* running on OSX:

- a) Install *XCode* from the *App Store*.
- b) Clone the source code of the *WP 43S* project by opening *Xcode* and clone it using [https://gitlab.com/Over\\_score/wp43s.git](https://gitlab.com/Over_score/wp43s.git). Cloning the project has some advantages over downloading the code. It makes sure *XCode* tracks the changes, for example.
- c) Install *MacPorts* using  
<https://guide.macports.org/chunked/installing.macports.html>
- d) Thereafter install the following *MacPorts* one by one:

```
sudo port install gcc9
sudo port install gtk3
sudo port install freeType
sudo port install pkgconfig
sudo port install x11
sudo port install dbus
```

- e) Then run the *makefile* form the root directory of the project (probably **wp43s**)
- f) When the project has successfully compiled, run the program, for example like this: **./wp43s**

In the terminal window the following warning may appear:

Gtk-WARNING \*\*: 11:23:52.903: Locale not supported by C library.  
Using the fallback 'C' locale.

Solve this by entering:

```
export LC_ALL="en_US"
export LANG="en_US"
export LANGUAGE="en_US"
export C_CTYPE="en_US"
export LC_NUMERIC=
export LC_TIME=en_US ... or similar locale settings.
```

If you get this error:

```
dbus[1369]: Dynamic session lookup supported but failed: launchd did  
not provide a socket path, verify that org.freedesktop.dbus-  
session.plist is loaded!
```

use this:

```
sudo launchctl load -w  
/Library/LaunchDaemons/org.freedesktop dbus-system.plist  
launchctl load /Library/LaunchAgents/org.freedesktop dbus-  
session.plist  
export  
DBUS_SESSION_BUS_ADDRESS="launchd:env=DBUS_FINK_  
SESSION_BUS_SOCKET"
```

After that I get no more warnings or error messages.

On the first 'make' I got error messages about header files that could not be found. I solved this by adding the following lines to the *makefile*. After:

```
else ifeq ($(detected_OS),Darwin) # Mac OS X  
CFLAGS += -D OSX
```

add:

```
CFLAGS += -I/opt/local/include/  
CFLAGS += -I/opt/local/include/glib-2.0/  
CFLAGS += -I/opt/local/lib/glib-2.0/include/  
CFLAGS += -I/opt/local/include/gtk-3.0/  
CFLAGS += -I/opt/local/include/pango-1.0/  
CFLAGS += -I/opt/local/include/cairo/  
CFLAGS += -I/opt/local/include/gdk-pixbuf-2.0/  
CFLAGS += -I/opt/local/include/atk-1.0/  
CFLAGS += -I/opt/local/include/freetype2
```

On my machine I put the project into ~/wp43s. However, the application searches for the `wp43s_pre.css` in the local home directory. If no calculator window comes up, copy the css-file:

```
cp wp43s_pre.css ~
```

Let me know if this does not work.

## How to Update Your WP 43S

If you have *Free42 (DM42)* still running on your calculator then proceed as demonstrated in previous chapter.

Else *WP 43S* is installed on your calculator already. Then:

1. Start your computer.
2. Take your calculator and turn it on.

Use **g [SAVE]** to save your programs and data. Then leave *WP 43S* pressing **g [MODE] g [SYSTEM]** to return to the *DMCP* system. If you had copied `keymap.bin` of *WP 43S* before last flashing, key assignments will stay as they were when navigating in the *DMCP* system – else the *Free42* assignments will become valid (cf. p. 163).

3. Connect your computer to the calculator *USB* socket using a proper data transfer cable. The flash disk of your calculator should show up as an external mass storage volume after you pressed ...

**6** Activate *USB* Disk

4. Start the internet browser on your computer and go to  
[https://gitlab.com/Over\\_score/wp43s/tree/master/DM42%20binary](https://gitlab.com/Over_score/wp43s/tree/master/DM42%20binary).

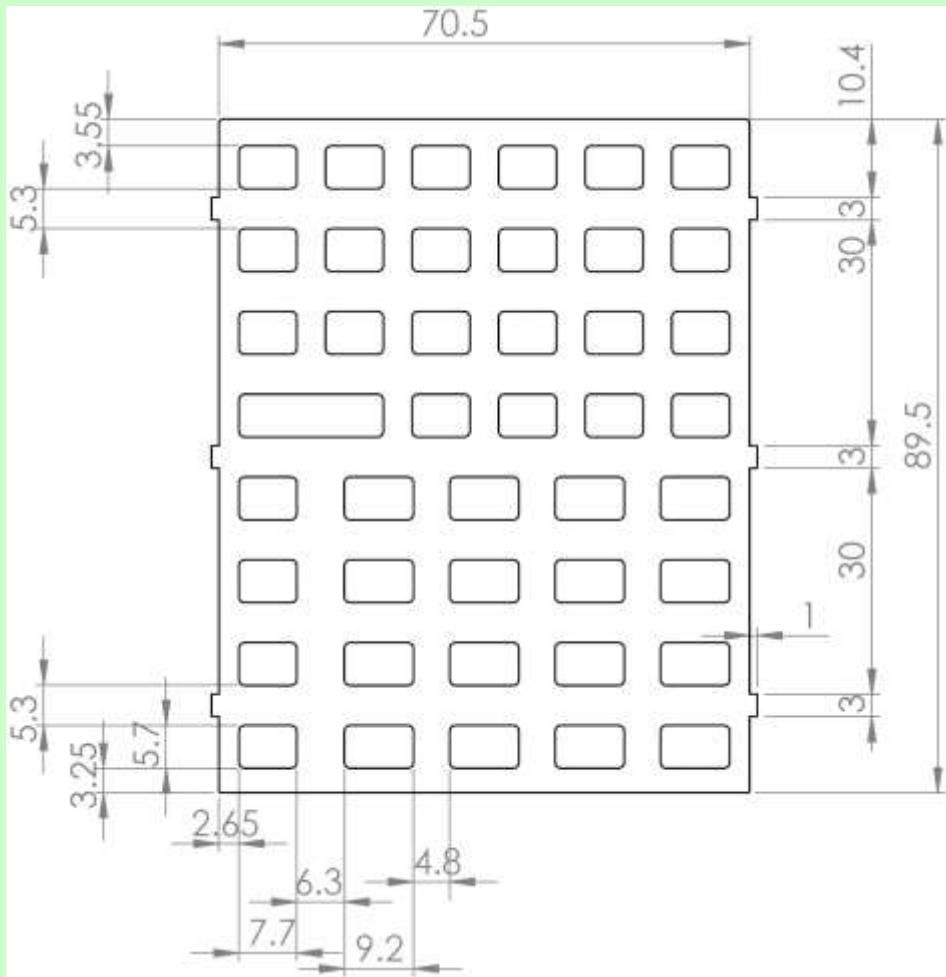
Copy `WP43S.pgm` to the *DM42* flash disk.

Option: Copy `keymap.bin` to the *DM42* flash disk if you have not done so far. This will reassign keys to match the *WP 43S* layout also when leaving *WP 43S*. Unless you have done this, **ON** remains bottom left.

5. Press **EXIT** **ENTER** **3** (Load Program), select `WP43S.pgm`, and press **ENTER** **ENTER**.
6. Wait for flashing completed (~15 s). Then press **EXIT** **EXIT** **1** **+**. Recall your saved programs and data via **f [I/O] LOAD**. You can resume your work with your updated *WP 43S* now.

## Overlays

See overleaf the drawing for a blank overlay. All dimensions are given in *millimeters*. Note the overall width is 72.5 mm.



Find the original print of your keys and keyplate for the current version of WP 43S on the last page of this manual, printed to scale.

## **APPENDIX G: TROUBLESHOOTING GUIDE**

### **Calculator Frozen**

There are several ways to put your calculator in a freeze state wherein it will not react on any keys you press, even without flashing WP 43S. Usually, pressing the RESET button on its rear side should bring it back to life. If this does not work, however, the following should do:

1. Open your calculator by unfastening the two bolts at the top of its backside. You will probably find a printed circuit board (*PCB*) whose top looks like this →

(cf. p. 166 for an earlier *PCB*).

In any case, you will see two small buttons, one labeled RESET.

The other one is called PGM.



2. Now:

- a. Press and hold the PGM button.
- b. Press and release the RESET button.
- c. Release the PGM button.

This shall reset your *DM42* and put it in bootloader mode.<sup>86</sup>

3. Then you can reflash your calculator using `dm_tool.exe` as written in [https://www.swissmicros.com/dm42/doc/dm42\\_user\\_manual/](https://www.swissmicros.com/dm42/doc/dm42_user_manual/).

---

<sup>86</sup> If this method should not work, however, this may point to a real hardware problem. We recommend contacting SwissMicros then.

## Keymap Trouble

If you find you have loaded a `keymap.bin` not matching the layout you wanted or you do not find the keys properly assigned then reset your calculator this way (assuming both calculator and computer running and connected):

1. Find where **MODE** went on the calculator and call it.
2. With **MODE** open, enter **g SYSTEM** to return to the *DMCP* system.
3. Press **6** (Activate *USB Disk*).
4. Start the internet browser on your computer and go to [https://gitlab.com/Over\\_score/wp43s/tree/master/DM42%20binary](https://gitlab.com/Over_score/wp43s/tree/master/DM42%20binary).
5. Copy `original_DM42_keymap.bin` to the flash disk, rename it `keymap.bin` and RESET the *DM42*.
6. Press **EXIT**, then the bottom left key – else you will run into the same troubles again.

Then your *WP 43S* is up and waiting for your orders.

## APPENDIX H: 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* and *complex* domain functions.

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. 13ff for general information)
$B_n$	$B_n$ returns the Bernoulli number for an integer $n > 0$ given in <b>X</b> : $B_n = (-1)^{n+1} \cdot n \cdot \zeta(1 - n)$ $B_n^*$ works with the old definition instead:
$B_n^*$	See p. 259 for $\zeta(x)$ . $B_n^* = 2 \cdot \frac{(2n)!}{(2\pi)^{2n}} \cdot \zeta(2n)$

Name	Remarks (see pp. 13ff for general information)
COMB, PERM	<p>For <math>y \geq x \geq 0</math> and <math>x, y \in \mathbb{N}</math>, <math>C_{y,x} = \binom{y}{x} = \frac{y!}{x!(y-x)!}</math> is the number of <i>combinations</i> and <math>P_{y,x} = \frac{y!}{(y-x)!} = x!C_{y,x}</math> the number of <i>permutations</i> of <math>x</math> and <math>y</math> as explained in the IOP (see pp. 28 and 62, respectively).</p> <p>Note <math>C_{y,0} = 1</math>, <math>C_{y,1} = y</math>, and <math>C_{y,2} = \frac{1}{2}y(y-1)</math>.</p> <p><math>C_{y,x}</math> applies to the <i>binomial distribution</i> (see p. 224): In a <i>Galton box</i><sup>87</sup> (a.k.a. <i>bean machine</i>) featuring <math>y</math> rows of pins and fed with <math>2^y</math> balls, <math>C_{y,x}</math> is the number of balls expected in column <math>x</math> of that box (start column counting with zero).</p> <p>Generally, <math>P_{y,x} = \frac{\Gamma(y+1)}{\Gamma(y-x+1)}</math> and <math>C_{y,x} = \frac{\Gamma(y+1)}{\Gamma(x+1) \cdot \Gamma(y-x+1)}</math> work also for non-integer numbers and in <i>complex domain</i>.</p>
FIB	<p>For integers, FIB returns the <i>Fibonacci</i> number <math>f_n</math> with <math>n = x</math>. The <i>Fibonacci</i> numbers are defined as <math>f_0 = 0</math>, <math>f_1 = 1</math>, and <math>f_n = f_{n-1} + f_{n-2}</math> for <math>n \geq 2</math>. With UNSIGNED, <math>f_{93}</math> 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 <i>real</i> or <i>complex</i> number <math>x</math>, with <math>\Phi = \frac{1+\sqrt{5}}{2}</math> denoting the golden ratio.</p>

<sup>87</sup> 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 eight continuous ones thereafter. Typical plots are shown for the PMF's or PDF's.

**Binom:** Binomial distribution with the number of successes **g** in X, the gross probability of a success  $p_0$  in I and the sample size **n** in J.

$\text{BINOM}_P$  returns

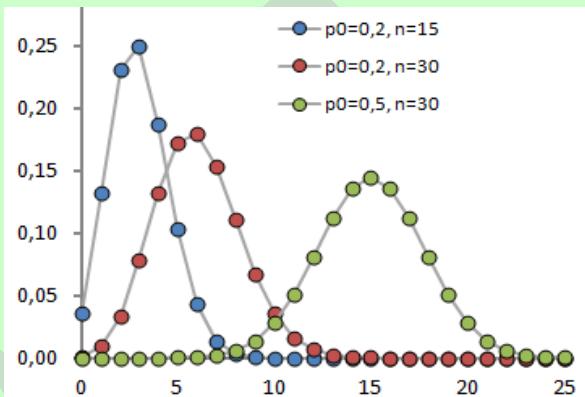
$$p_B(g; n; p_0) = \binom{n}{g} p_0^g (1 - p_0)^{n-g} = C_{n,g} p_0^g (1 - p_0)^{n-g} \quad (\text{see COMB on p. 223 for the explanation of the notation}).$$

$\text{BINOM}_{\Delta}$  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.

**Example:** What is the probability for finding no faulty item 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<sub>Δ</sub>



returning 0.633 – so the odds are almost two out of three that you will not detect any defect in your sample! <sup>88</sup>

Read here for more information:

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

**Geom:** Geometric distribution:

GEOM<sub>P</sub> returns

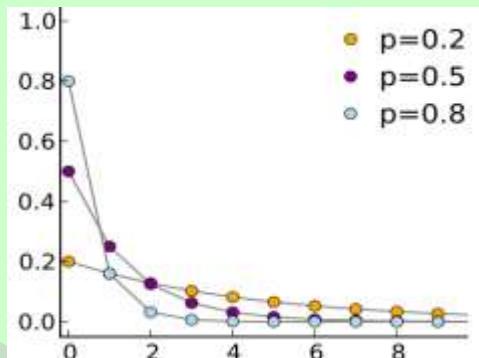
$$p_{Ge}(n) = p_0(1-p_0)^n$$

GEOM<sub>A</sub> returns

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

being the probability for a first success after  $m=x$  Bernoulli experiments.

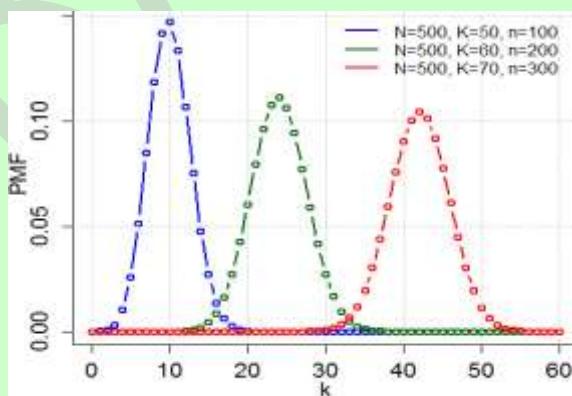
The probability  $p_0$  for a success in each such experiment must be specified in I.



Start reading here for more:

[http://en.wikipedia.org/wiki/Geometric\\_distribution](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 (in the diagram,  $g=k$ ,  $p_0=K/N$ , and  $n_0=N$ ).



<sup>88</sup> The exact result for said boundary conditions 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 (often simplified) statistical model used matches reality no better than that.

HYPER<sub>P</sub> 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. 223 for the explanation of the notation).

While the *binomial distribution* assumes that each sample part is returned to the batch after 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](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*  $p_0$  in **I**, and  $n$  in **J**.

NBIN<sub>P</sub> returns  $p_{NB}(f; n; p_0) = \binom{n-1}{f-1} \cdot p_0^f \cdot (1-p_0)^{n-f}$   
 $= C_{n-1, f-1} \cdot p_0^f \cdot (1-p_0)^{n-f}$  (see COMB on p. 223 and cf. BINOM).

Start reading here for more:

[http://en.wikipedia.org/wiki/Negative\\_binomial\\_distribution](http://en.wikipedia.org/wiki/Negative_binomial_distribution).

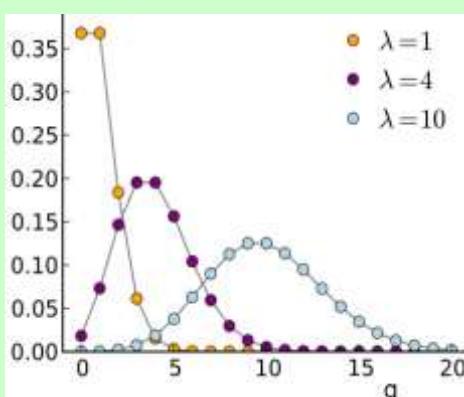
**Poiss:** Poisson distribution with the *number of successes g* in **X** and the *Poisson parameter λ* in **J**.

POISS<sub>P</sub> 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



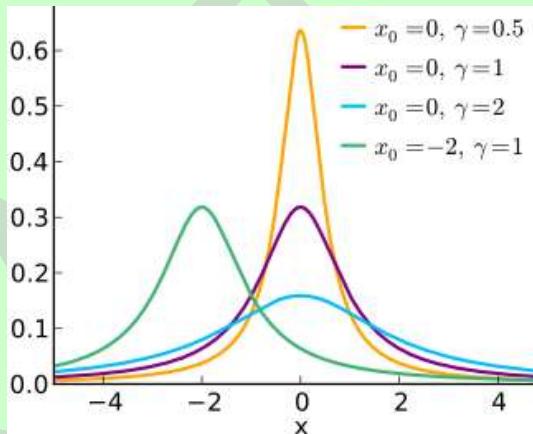
simplest model for industrial sampling tests – use  $\lambda = np_0$  with the gross error probability  $p_0$  and the sample size  $n$  (cf. BINOM). For the example introduced with BINOM above, POISS returns 0.638.

Read here for more information:

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

## Continuous distributions:

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



CAUCH<sub>P</sub> returns  $f_{Ca}(x) = \left\{ \pi\gamma \cdot \left[ 1 + \left( \frac{x - x_0}{\gamma} \right)^2 \right] \right\}^{-1}$ ,

CAUCH<sub>A</sub> returns  $F_{Ca}(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x - x_0}{\gamma}\right)$ ,

CAUCH<sup>-1</sup> returns  $F_{Ca}^{-1}(p) = x_0 + \gamma \tan\left[\pi \cdot \left(p - \frac{1}{2}\right)\right]$ .

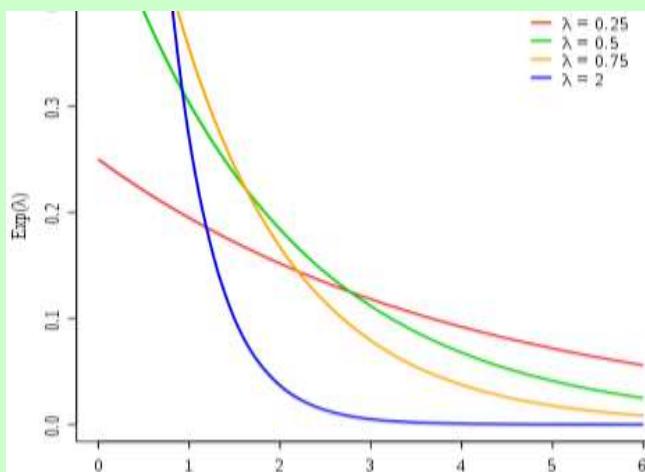
This distribution is quite popular in physics. It is a special case of Student's *t distribution*. Start reading here for more:

[http://en.wikipedia.org/wiki/Cauchy\\_distribution](http://en.wikipedia.org/wiki/Cauchy_distribution).

**Expon:** Exponential distribution with the rate  $\lambda$  in **I**.

$\text{EXPON}_P$  returns  $f_{Ex}(x) = \lambda e^{-\lambda x}$ .

$\text{EXPON}_{\Delta}$  returns  $F_{Ex}(x) = 1 - e^{-\lambda x}$ .



Read here for more information:

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

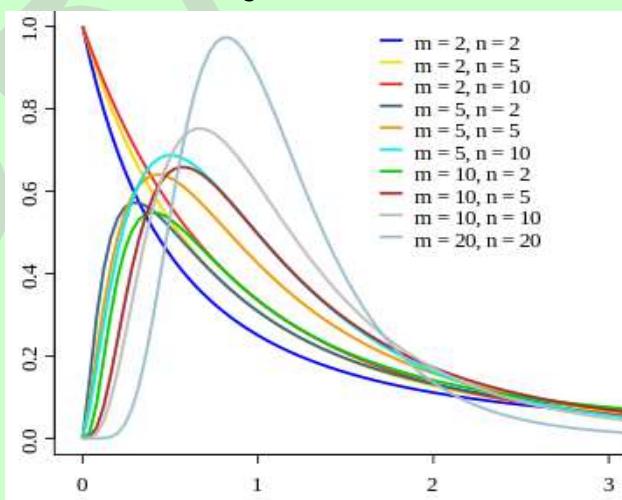
**F(x):** Fisher's F distribution with the degrees of freedom in **I** and **J**.

It is used e.g. for analyses of variance (ANOVA).

The diagram 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>

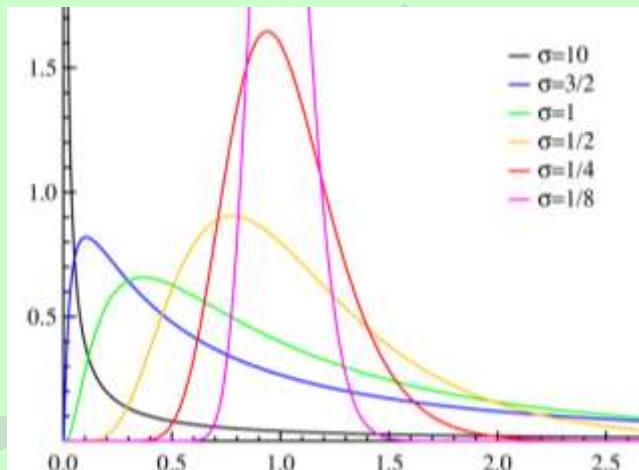


**LgNrm:** *Log-normal distribution* with the parameters  $\mu = \ln \bar{x}_g$  in **I** and  $\sigma = \ln \varepsilon$  in **J** (see some *PDF* plots below).

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

LGNRM<sub>A</sub> returns  $F_{Ln}(x) = \Phi\left(\frac{\ln(x)-\mu}{\sigma}\right)$  with  $\Phi(z)$  denoting the

*standardized normal CDF*  
as presented  
on p. 232.



Read here for  
more information:

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

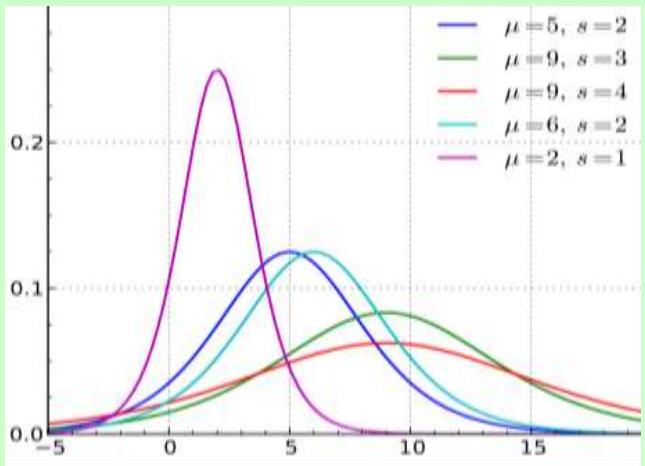
**Logis:** *Logistic distribution* with an arbitrary mean  $\mu$  given in **I** and a scale parameter  $s$  in **J**.

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

LOGIS<sub>P</sub> returns  $f_{Lg}(x) = \frac{e^{-\xi}}{(1+e^{-\xi})^2 s}$  (plotted overleaf) and

LOGIS<sub>A</sub> returns  $F_{Lg}(x) = \frac{1}{1+e^{-\xi}}$ .

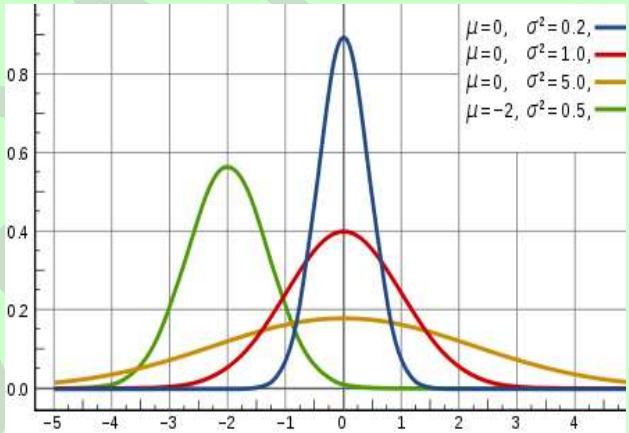
LOGIS<sup>-1</sup> returns  $F_{Lg}^{-1}(p) = \mu + s \ln\left(\frac{p}{1-p}\right)$ .



Start reading here for more:

[http://en.wikipedia.org/wiki/Logistic\\_distribution](http://en.wikipedia.org/wiki/Logistic_distribution).

**Norml:** *Normal distribution* with an arbitrary *mean*  $\mu$  given in **I** and an arbitrary *standard deviation*  $\sigma$  in **J**. The red curve (for  $\mu=0$  and  $\sigma=1$ ) is the *standardized normal* (a.k.a. *Gaussian*) distribution  $\varphi$ .



NORML<sub>P</sub> returns  $f_N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \varphi\left(\frac{x-\mu}{\sigma}\right)$  and

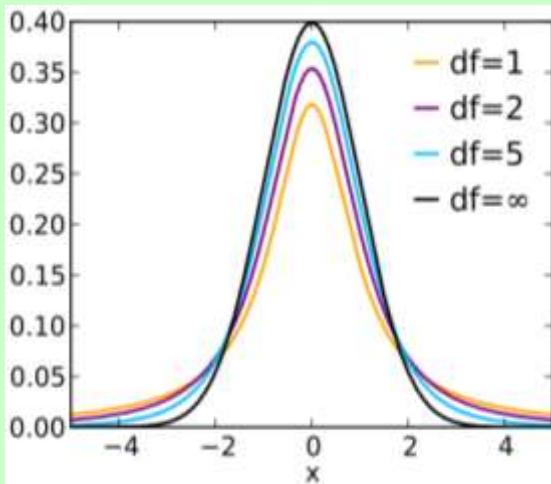
NORML<sub>A</sub> returns  $F_N(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$  with  $\Phi(z)$  denoting the *standard normal* (or *Gaussian*) CDF as presented on p. 232.

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**.

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. 230).



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<sub>P</sub> returns  $f_W(t) = \frac{b}{T} \cdot \left(\frac{t}{T}\right)^{b-1} e^{-(t/T)^b}$  for  $t \geq 0$ , else 0. This is a very flexible function – see the curves plotted overleaf.

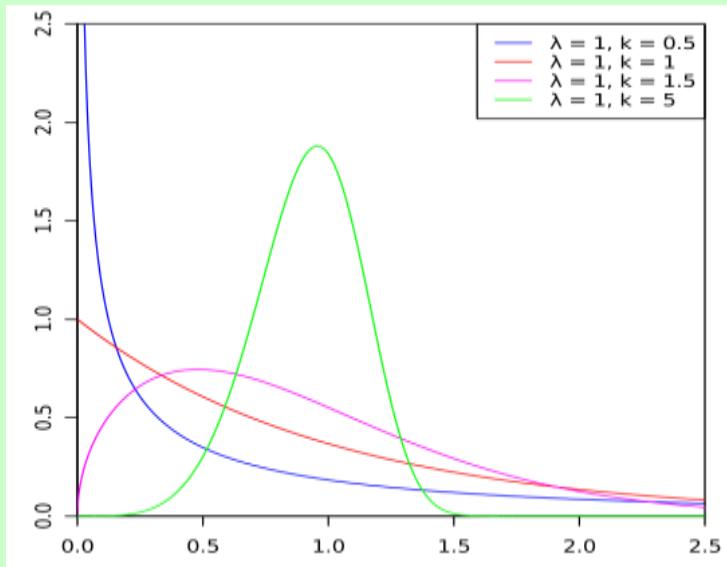
WEIBL<sub>A</sub> returns  $F_W(t) = 1 - e^{-(t/T)^b}$

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

Read here for more information:

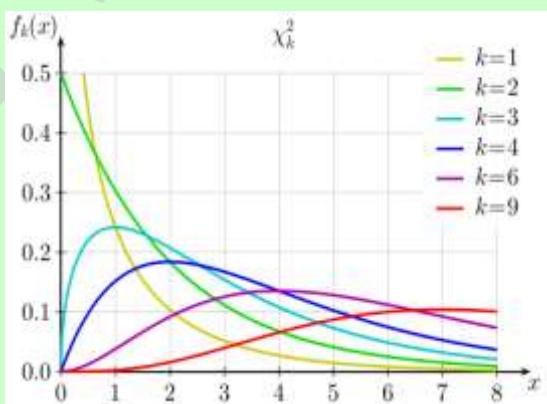
<http://www.itl.nist.gov/div898/handbook/eda/section3/eda3668.htm>

You may even find some more application fields mentioned in [https://en.wikipedia.org/wiki/Weibull\\_distribution#Applications](https://en.wikipedia.org/wiki/Weibull_distribution#Applications).



$\varphi(x)$  and  $\Phi(x)$ :  $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$  is the *standardized normal PDF* (i.e. the famous *Gaussian bell curve* as drawn in red under NORML on p. 230), while  $\Phi(x) = \int_{-\infty}^x \varphi(\tau) d\tau$  is the corresponding CDF (cf. the *error function* on p. 256). Take NORML instead.

$\chi^2(x)$ : *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's 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  $\varepsilon_m$ ).
- For samples drawn out of other kinds of processes other measures apply.

Generally, the statistical model shall be chosen that matches observations best – within their statistical errors.<sup>89</sup> Be assured not everything is *Gaussian* in real world.<sup>90</sup> Process characteristics can be detected (and should be checked well in advance of calculating e.g. means) using suitable tests – turn to applicable statistical reference literature.

---

<sup>89</sup> In real-life cases, however, dramatic deviations from the model distribution are often found – then you cannot expect the calculated consequences matching reality any better.

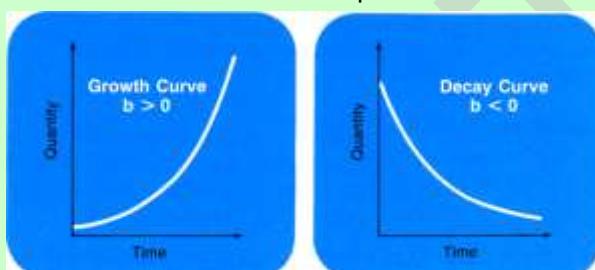
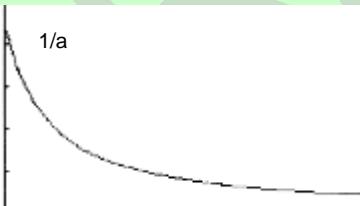
As mentioned in the main text, we recommend you look deeply into statistics textbooks to ensure you fully understand what you do with the functions provided in your *WP 43S*. The real world shows lots of sad examples where people full of good will caused large damages by applying tools they did not know 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 with fools becoming busy*”), a former boss of mine used to say (compare also D.T. recently).

<sup>90</sup> Since the *PDF* of a *Gaussian* (a.k.a. *normal*) 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 and powerful model describing a lot of real world observations very well. Just never forget the limits of such models.

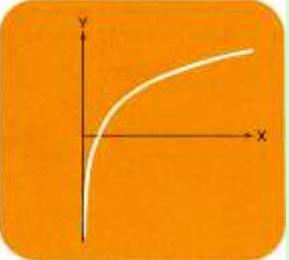
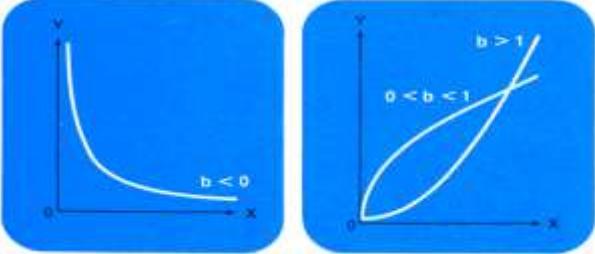
The following functions as named in the left column (sorted alphabetically) are all found in STAT:

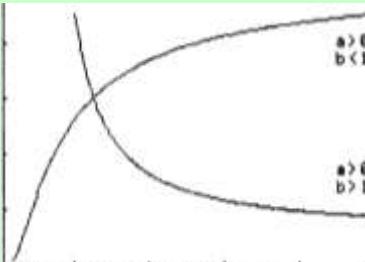
Name	Remarks (see pp. 13ff for general information)
Cauchy F	Selects a <i>Cauchy</i> (a.k.a. <i>Lorentz</i> , <i>Breit-Wigner</i> ) peak fit model $R(x) = \frac{1}{[a_0(x + a_1)^2 + a_2]}$ for least squares regression. <sup>91</sup> See p. 227 for shapes of such peaks.
CORR	For any set of data points $(x_i, y_i)$ , the <i>coefficient of correlation</i> is $r = \frac{s_{xy}}{s_x \cdot s_y}$ . See $s_{XY}$ and $s$ below. For an arbitrary fit model $R(x)$ , $r^2 = 1 - \frac{\sum [R(x_i) - y_i]^2}{\sum (\bar{y} - y_i)^2}$ is its <i>coefficient of determination</i> indicating the fraction of the variation of the dependent data $y$ determined by the variation of the inde- pendent 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 $x$ . Note BESTF picks the fit model showing the maximum $r^2$ out of the models allowed. A two-parameter regression (like the majority of the fit models provided on your WP 43S) is said being (statistically) <i>significant</i> at a 99% <i>confidence level</i> if $\sqrt{\frac{r^2}{1 - r^2}(n - 2)} > t_{n-2}^{-1}(0.99)$ with the right side being the inverse of the <i>t distribution</i> for the <i>degrees of freedom</i> $n - 2$ (see p. 231).

<sup>91</sup> Note that *least squares regression* is best for data point errors in direction  $y$  being significantly greater than the errors in direction  $x$ . See pp. 232ff for the formulas and more about the fit models provided.

Name	Remarks (see pp. 13ff for general information)
COV	For any set of data points $(x_i, y_i)$ , the <i>population covariance</i> is $COV_{xy} = \frac{1}{n^2} \left( n \sum x_i y_i - \sum x_i \sum y_i \right)$ <p style="text-align: right;">Compare <math>s_{XY}</math> below.</p>
ExpF	Selects the exponential curve fit model $R(x) = a_0 e^{a_1 x}$ for least squares regression. <sup>91</sup> Generally, this will be a good choice if the measured data follow the shape of one of the two curves pictured here (think of human population growth or nuclear decay, for example). <sup>92</sup> 
Gauss F	Selects a <i>Gauss peak</i> fit model $R(x) = a_0 e^{\frac{(x-a_1)^2}{a_2}}$ for least squares regression. <sup>91</sup> See p. 230 for the shapes of such peaks.
HypF	 <p>Selects the hyperbolic fit model <math>R(x) = \frac{1}{(a_0 + a_1 x)}</math> for least squares regression.<sup>91</sup></p>
LinF	Selects the linear fit model $R(x) = a_0 + a_1 x$ for least squares regression. <sup>91</sup> Generally, this will be a good choice if the measured data follow a straight line, raising or falling (but compare ORTHOF).

<sup>92</sup> Color plots on this page and the next are taken from the HP-27 manual;  $b=a_1$  on your WP 43S.

Name	Remarks (see pp. 13ff for general information)
LogF	 <p>Selects the logarithmic curve fit model <math>R(x) = a_0 + a_1 \ln(x)</math> for least squares regression.<sup>91</sup> Generally, this will be a good choice if the measured data follow a curve looking like drawn at left.</p>
L.R.	<p>Uses the fit model selected and computes the two or three parameters of the regression for the data accumulated.</p> <p>For all curve fit models provided on your WP 43S, a regression parameter is (statistically) <i>significant</i> at a 99% confidence level 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> <math>n - 2</math> (cf. p. 231).</p>
OrthoF	Selects the linear fit model $R(x) = a_0 + a_1 x$ like LINF, but 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> . See pp. 239ff and the OM for more.
ParabF	Selects a parabolic fit model $R(x) = a_0 + a_1 x + a_2 x^2$ for least squares regression. <sup>91</sup>
PowerF	<p>Selects the power curve fit model <math>R(x) = a_0 x^{a_1}</math> for least squares regression.<sup>91</sup> Generally, this will be a good choice if measured data follow the shape of one of the curves pictured here (look for <i>Tower of Pisa</i> in the OM).<sup>92</sup></p> 

Name	Remarks (see pp. 13ff for general information)
RootF	 <p>Selects the root curve fit model  <math>R(x) = a b^{1/x} = a_0 a_1^{1/x}</math> for a least squares regression.<sup>91</sup></p>
$s_{XY}$	<p>For any set of data points <math>(x_i, y_i)</math>, the <i>sample covariance</i> is</p> $s_{xy} = \frac{1}{n(n-1)} \left( n \sum x_i y_i - \sum x_i \sum y_i \right)$ <p>Compare COV above.</p>
$s, s_m$	<p>The <i>sample standard deviation (SD)</i> is the positive square root of the <i>sample variance</i></p> $s_x^2 = \frac{1}{n(n-1)} \left[ n \sum x_i^2 - \left( \sum x_i \right)^2 \right] = \frac{1}{n-1} \left( \sum x_i^2 - n \bar{x}^2 \right)$ <p>And the <i>standard error</i> (i.e. the SD of the mean <math>\bar{x}</math>) is <math>s_m = s / \sqrt{n}</math></p>
$s_w, s_{mw}$	<p>The <i>sample SD</i> for <u>weighted</u> data (where the weight <math>y_i</math> of each data point <math>x_i</math> was entered via <b>[Σ+]</b>) is</p> $s_w = \sqrt{\frac{\sum y_i \sum y_i x_i^2 - (\sum y_i x_i)^2}{\sum y_i (\sum y_i - 1)}}$ <p>And the corresponding <i>standard error</i> (the SD of the mean <math>\bar{x}_w</math>) is</p> $s_{mw} = \frac{1}{\sum y_i} \sqrt{\frac{\sum y_i \sum y_i x_i^2 - (\sum y_i x_i)^2}{\sum y_i - 1}}$

Name	Remarks (see pp. 13ff for general information)
$\bar{x}$	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 $\bar{x}_G = \sqrt[n]{\prod x_i} = e^{\left[\frac{1}{n} \sum \ln(x_i)\right]}$
$\bar{x}_H$	The <i>harmonic mean</i> is calculated as $\bar{x}_H = \frac{n}{\sum \frac{1}{x_i}}$
$\bar{x}_{RMS}$	The <i>quadratic mean</i> is calculated as $\bar{x}_{RMS} = \sqrt{\frac{1}{n} \sum x_i^2}$
$\bar{x}_w$	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 $\bar{x}_w = \frac{\sum x_i y_i}{\sum y_i}$
$\varepsilon$	The <i>scattering factor</i> $\varepsilon_x$ for a sample of <i>log-normally</i> distributed data is calculated via: $\ln(\varepsilon_x) = \sqrt{\frac{1}{n-1} \left[ \sum \ln^2(x_i) - 2n \ln(\bar{x}_G) \right]}$ Compare s.
$\varepsilon_m$	The <i>scattering factor</i> of the <i>geometric mean</i> is $\varepsilon_m = \varepsilon^{\frac{1}{\sqrt{n}}}$ . Compare s <sub>m</sub> .
$\varepsilon_p$	The <i>scattering factor</i> $\varepsilon_p$ for a population of <i>log-normally</i> distributed data is calculated via: $\ln(\varepsilon_p) = \sqrt{\frac{n-1}{n}} \ln(\varepsilon)$ Compare $\sigma$ .

Name	Remarks (see pp. 13ff for general information)
$\sigma$	The <i>SD</i> of a population of <i>normally</i> distributed data is calculated via $\sigma = \sqrt{\frac{n-1}{n}} s$
$\sigma_w$	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 the Curve Fitting Models Provided

Actually, a proper linear regression is computed for LINF and ORTHOF only. For the other three standard 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 22 sums listed on pp. 96f and increments **n**.
2. The evaluation depends on the fit model you select (cf. pp. 235ff):
  - a. If you choose LINF then the least squares regression line parameters  $a_0$  and  $a_1$  will be computed following the formulas:

$$a_0 = \frac{\sum x_i^2 \cdot \sum y_i - \sum x_i \cdot \sum x_i y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

$$a_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} = \frac{s_{xy}}{s_x^2} = r \frac{s_y}{s_x}$$

Their *standard errors* can be calculated using the formulas

$$s(a_1) = \frac{s_y}{s_x} \sqrt{\frac{1-r^2}{n-2}} \quad \text{and} \quad s(a_0) = s(a_1) \cdot \sqrt{\frac{n-1}{n} s_x^2 + \bar{x}^2} \quad \text{with}$$

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

- 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

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

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

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

The standard errors of  $a_{0,tEXP}$  and  $a_{1,tEXP}$  can be calculated using the formulas for LINF on p. 240 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 for LINF on p. 240 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 for LINF on p. 240 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 HYPF then the parameters of the least squares regression curve  $R(x) = \frac{1}{a_0 + a_1 x}$  are computed to be

$$a_{0,HYP} = \frac{\sum x_i^2 \cdot \sum \frac{1}{y_i} - \sum x_i \cdot \sum \frac{x_i}{y_i}}{n \sum x_i^2 - (\sum x_i)^2} \text{ and } a_{1,HYP}$$

$$= \frac{n \sum \frac{x_i}{y_i} - \sum x_i \cdot \sum \frac{1}{y_i}}{n \sum x_i^2 - (\sum x_i)^2}$$

$$r_{HYP}^2 = \frac{a_{0,HYP} \sum \frac{1}{y_i} + a_{1,HYP} \sum \frac{x_i}{y_i} - \frac{1}{n} \left( \sum \frac{1}{y_i} \right)^2}{\sum \frac{1}{y_i^2} - \frac{1}{n} \left( \sum \frac{1}{y_i} \right)^2}$$

- f. If you choose ROOTF then the least squares regression curve parameters will be computed using

$$A = n \sum \frac{1}{x_i^2} - \left( \sum \frac{1}{x_i} \right)^2$$

$$B = \frac{1}{A} \left[ \sum \frac{1}{x_i^2} \cdot \sum \ln(y_i) - \sum \frac{1}{x_i} \cdot \sum \frac{\ln(y_i)}{x_i} \right]$$

$$C = \frac{1}{A} \left[ n \sum \frac{\ln(y_i)}{x_i} - \sum \frac{1}{x_i} \cdot \sum \ln(y_i) \right]$$

The parameters of the fit curve  $R(x) = a_0 a_1^{1/x}$  turn out being just  $a_{0,t\sqrt{-}} = e^B$  and  $a_{1,t\sqrt{-}} = e^C$ .

$$r_{t\sqrt{-}}^2 = \frac{B \sum \ln(y_i) + C \sum \frac{\ln(y_i)}{x_i} - \frac{1}{n} [\sum \ln(y_i)]^2}{\sum [\ln(y_i)]^2 - \frac{1}{n} [\sum \ln(y_i)]^2}$$

- g. If you choose PARABF then the least squares regression curve parameters will be computed using

$$A = n \sum x_i^2 - \left( \sum x_i \right)^2, \quad B = n \sum x_i^2 y_i - \sum x_i^2 \cdot \sum y_i,$$

$$C = n \sum x_i^3 - \sum x_i^2 \cdot \sum x_i, \quad D = n \sum x_i y_i - \sum x_i \cdot \sum y_i,$$

$$E = n \sum x_i^4 - \left( \sum x_i^2 \right)^2$$

The parameters of the fit curve  $R(x) = a_0 + a_1x + a_2x^2$  will then be

$$a_{2,PAR} = \frac{A B - C D}{A E - C^2}, \quad a_{1,PAR} = \frac{D - a_2 C}{A},$$

$$\text{and } a_{0,PAR} = \frac{1}{n} \left( \sum y_i - a_{2,PAR} \sum x_i^2 - a_{1,PAR} \sum x_i \right).$$

$$\text{And } r_{PAR}^2 = \frac{a_{0,PAR} \sum y_i + a_{1,PAR} \sum x_i y_i + a_{2,PAR} \sum x_i^2 y_i - \frac{1}{n} (\sum y_i)^2}{\sum y_i^2 - \frac{1}{n} (\sum y_i)^2}$$

**h. If you choose GAUSSF** then the least squares regression curve parameters will be computed using the auxiliary terms A, B, C, D, and E exactly as for PARABF. Furthermore,

$$F = \frac{A B - C D}{A E - C^2}, \quad G = \frac{D - F C}{A},$$

$$\text{and } H = \frac{1}{n} \left( \sum \ln(y_i) - F \sum x_i^2 - G \sum x_i \right).$$

The parameters of the fit curve  $R(x) = a_0 e^{(x-a_1)^2/a_2}$  will then be

$$a_{2,GAU} = \frac{1}{F}, \quad a_{1,GAU} = -\frac{G}{2} a_{2,GAU} \quad \text{and} \quad a_{0,GAU} = e^{H - F a_{1,GAU}^2}.$$

$$r_{GAU}^2 = \frac{H \sum \ln(y_i) + G \sum x_i \ln(y_i) + F \sum x_i^2 \ln(y_i) - \frac{1}{n} [\sum \ln(y_i)]^2}{\sum [\ln(y_i)]^2 - \frac{1}{n} [\sum \ln(y_i)]^2}$$

- i. If you choose CAUCHF then the least squares regression curve parameters will be computed using the auxiliary terms A and E exactly as in PARABF. The other terms will be

$$B = n \sum \frac{x_i^2}{y_i} - \sum x_i^2 \cdot \sum \frac{1}{y_i}$$

$$C = n \sum x_i^3 - \sum x_i \cdot \sum x_i^2$$

$$D = n \sum \frac{x_i}{y_i} - \sum x_i \cdot \sum \frac{1}{y_i}$$

F and G will be calculated as for GAUSSF but with the components computed here; and

$$H = \frac{1}{n} \left( \sum \frac{1}{y_i} - R_{12} \sum x_i - R_{13} \sum x_i^2 \right)$$

The fit curve  $R(x) = 1/[a_0(x + a_1)^2 + a_2]$  will be specified by:

$$a_{0,CAU} = F, \quad a_{1,CAU} = \frac{G}{2a_0}, \quad \text{and} \quad a_{2,CAU} = H - F a_1^2.$$

$$r_{CAU}^2 = \frac{H \sum \frac{1}{y_i} + G \sum \frac{x_i}{y_i} + F \sum \frac{x_i^2}{y_i} - \frac{1}{n} \left( \sum \frac{1}{y_i} \right)^2}{\sum \left( \frac{1}{y_i} \right)^2 - \frac{1}{n} \left( \sum \frac{1}{y_i} \right)^2}$$

- j. 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 through i, if allowed (cf. the IO!). The model delivering the greatest absolute  $r$  value will be selected.
- k. If you choose ORTHOF then the least squares regression line parameters  $a_0$  and  $a_1$  will be computed following the formulas:

$$a_1 = \frac{1}{2s_{xy}} \left[ s_y^2 - s_x^2 \pm \sqrt{(s_y^2 - s_x^2)^2 + 4s_{xy}^2} \right] \quad \text{and} \quad a_0 = \bar{y} - a_1 \bar{x}$$

The other formulas can be taken from model a (i.e. from LINF).

The output of L.R. is formatted like this, allowing for up to 12 significant digits displayed for each model parameter:

Linear	$a_1 = -312.345\ 678\ 901$
$y = a_0 + a_1 x$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Exponential	$a_1 = -12.345\ 678\ 901$
$y = a_0 e^{(a_1 x)}$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Power	$a_1 = -12.345\ 678\ 901$
$y = a_0 x^{a_1}$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Logarithmic	$a_1 = -12.345\ 678\ 901$
$y = a_0 + a_1 \ln(x)$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Hyperbolic	$a_1 = -12.345\ 678\ 901$
$y = (a_0 + a_1 x)^{-1}$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Root	$a_1 = -12.345\ 678\ 901$
$y = a_0 a_1^{(1/x)}$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Parabolic	$a_2 = -2.345\ 678\ 901\ 2$
$y =$	$a_1 = -12.345\ 678\ 901$
$a_0 + a_1 x + a_2 x^2$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Gauss peak	$a_2 = -2.345\ 678\ 901\ 2$
$\ln(y) =$	$a_1 = -12.345\ 678\ 901$
$a_0 + (x - a_1)^2 / a_2$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Cauchy peak	$a_2 = -2.345\ 678\ 901\ 2$
$1/y =$	$a_1 = -12.345\ 678\ 901$
$a_0 (a_1 + x)^2 + a_2$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$
Orthogonal	$a_1 = -312.345\ 678\ 901$
$y = a_0 + a_1 x$	$a_0 = -1.234\ 567\ 8 \times 10^{-3}$

## About Error Propagation

Experimental data are always attended with errors (cf. footnote 43), caused by e.g. the uncertainty of the measuring method, the instrument used, and/or environmental variations. Even under controlled environmental and 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  $\Delta x_i$ . Now, if  $R = f(x_1, \dots, x_n)$  then its error

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

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  $\Delta R$ , 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

---


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

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 of  $f$ , the following simpler algorithm will do, requiring down to half as many steps:

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

---


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

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

## 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  $\delta$  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  $\Delta t = t_{i+1} - t_i$ :

$$f_1 = f_0 + \left( \frac{df}{dt} \right)_0 \text{ and } \left( \frac{df}{dt} \right)_1 = \left( \frac{df}{dt} \right)_0 + a\Delta t, \\ f_2 = f_1 + \left( \frac{df}{dt} \right)_1 \text{ and } \left( \frac{df}{dt} \right)_2 = \left( \frac{df}{dt} \right)_1 + a\Delta t, \text{ etc.}$$

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  $\Delta t$  again, we get

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

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. 248, we will get the following new set:

$$\begin{aligned} \frac{df}{dt}_{1/2} &\approx \frac{df}{dt}_0 + \left[ a - b \left(\frac{df}{dt}\right)_0^2 \right] \frac{\Delta t}{2} \\ \left(\frac{df}{dt}\right)_{i+1/2} &\approx \left(\frac{df}{dt}\right)_{i-\frac{1}{2}} + \left[ a - b \left(\frac{df}{dt}\right)_{i-\frac{1}{2}}^2 \right] \Delta t \\ f_{i+1} &\approx f_i + \left(\frac{df}{dt}\right)_{i+1/2} \Delta t \end{aligned}$$

Proceeding from time zero in small steps  $\Delta t$  again, we get

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

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

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

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 two-dimensional 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:

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

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

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

## Orthogonal Polynomials

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

Name	Remarks (see pp. 13ff for general information)
$H_n$	<p><b>Hermite polynomials</b> for <u>probability</u>: <math>H_n(x) = (-1)^n \cdot e^{x^2/2} \cdot \frac{d^n}{dx^n} \left( e^{-x^2/2} \right)</math></p> <p>with <math>n</math> in <b>Y</b>, 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> <p>Legend:</p> <ul style="list-style-type: none"><li><math>H_0</math> (blue line)</li><li><math>H_1</math> (magenta line)</li><li><math>H_2</math> (yellow line)</li><li><math>H_3</math> (green line)</li><li><math>H_4</math> (cyan line)</li></ul>

Name	Remarks (see pp. 13ff for general information)
$H_{np}$	<p><b>Hermite polynomials</b> for <u>physics</u>: <math>H_{np}(x) = (-1)^n \cdot e^{x^2} \cdot \frac{d^n}{dx^n}(e^{-x^2})</math></p> <p>with <math>n</math> in <math>\mathbb{Y}</math>, solving the same differential equation. See the first five polynomials plotted below.</p>

DF

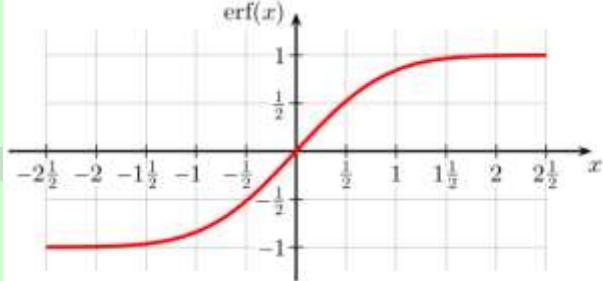
Name	Remarks (see pp. 13ff for general information)
$L_m$	<p><b>Laguerre polynomials</b> (compare <math>L_{n\alpha}</math> below):</p> $L_n(x) = \frac{e^x}{n!} \cdot \frac{d^n}{dx^n} (x^n e^{-x}) = L_n^{(0)}(x) \text{ with } n \text{ in Y, solving the differential equation } x \cdot f''(x) + (1-x) \cdot f'(x) + n \cdot f(x) = 0.$ <p>See the first five Laguerre polynomials plotted here.</p>
$L_{m\alpha}$	<p><b>Laguerre's generalized polynomials</b> (compare <math>L_n</math> above):</p> $L_n^{(\alpha)}(x) = \frac{x^{-\alpha} e^x}{n!} \cdot \frac{d^n}{dx^n} (x^{n+\alpha} e^{-x}) \text{ with } n \text{ in Y and } \alpha \text{ in Z. Some of them are plotted below (k = \alpha).}$

Name	Remarks (see pp. 13ff for general information)
$P_n$	<p><b>Legendre polynomials:</b> <math>P_n(x) = \frac{1}{2^n n!} \cdot \frac{d^n}{dx^n} [(x^2 - 1)^n]</math> with <math>n</math> in <math>\mathbb{Y}</math>, 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><b>Chebyshev (a.k.a. Čebyšev, Tschebyschow, Tschebyscheff) polynomials of first kind</b></p> $T_n(x) = \begin{cases} \cos(n \arccos(x)) & \text{for } -1 \leq x \leq 1 \\ \cosh(n \operatorname{arcosh}(x)) & \text{for } x > 1 \\ (-1)^n \cosh(n \operatorname{arcosh}(-x)) & \text{for } x < -1 \end{cases} \quad \text{with } n \text{ in } \mathbb{Y}, \text{ solving}$ <p>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 <math>T_0(x) \dots T_5(x)</math>.</p>

Name	Remarks (see pp. 13ff for general information)
$U_n$	<p><b>Chebyshev polynomials of second kind</b> <math>U_n(x)</math> with <math>n</math> in <math>\mathbb{Y}</math>, solving the differential equation</p> $f''(x) - \frac{3x}{1-x^2} f'(x) + \frac{n(n+2)}{1-x^2} f(x) = 0$ <p>The plot below shows <math>U_0(x) \dots U_5(x)</math>:</p>

## Even More Mathematical Functions

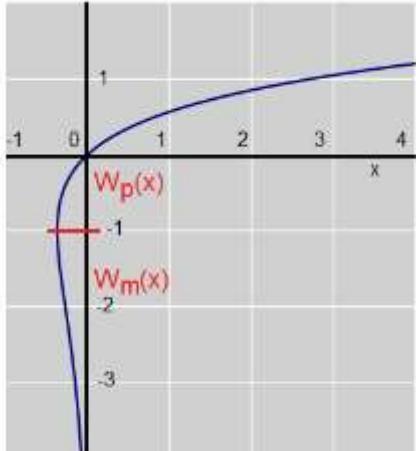
All the following functions are found in X.FN. Some of them are for pure mathematics only but were useful at some stages of the *WP 34S* or *WP 43S* projects, so we made them accessible for the public.

Name	Remarks (see pp. 13ff for general information)
AGM	Returns the <i>arithmetic-geometric mean</i> . Find more about it here: <a href="http://mathworld.wolfram.com/Arithmetic-GeometricMean.html">http://mathworld.wolfram.com/Arithmetic-GeometricMean.html</a> .
erf	<p>Returns the <i>error function</i> <math>\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\tau^2} d\tau</math>.</p> <p>Note that <math>\text{erf}\left(\frac{x}{\sqrt{2}}\right) = 2 \cdot \Phi(x) - 1</math> with <math>\Phi(x)</math> representing the <i>standard normal CDF</i> as described on p. 232.</p> <p>Beyond statistics, the <i>error function</i> may be helpful in heat conduction and diffusion problems, for instance.</p> 
erfc	This command returns the <i>complementary error function</i> $\text{erfc}(x) = 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-\tau^2} d\tau$ . This function is related to the <i>error probability</i> of the <i>standard normal distribution</i> .

Name	Remarks (see pp. 13ff for general information)
$g_d$ , $g_d^{-1}$	<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 <i>real</i> values. The <i>inverse</i> of this function is</p> $g_d^{-1}(x) = \int_0^x \frac{d\xi}{\cos \xi}.$ <p>Start reading here for more:  <a href="http://en.wikipedia.org/wiki/Gudermannian_function">http://en.wikipedia.org/wiki/Gudermannian_function</a>.</p>
$I_{xyz}$	<p>Returns the <i>regularized (incomplete) Beta function</i> <math>\frac{\beta_x(x, y, z)}{B(y, z)}</math></p> <p>with <math>\beta_x(x, y, z) = \int_0^x t^{y-1} (1-t)^{z-1} dt</math> being the <i>incomplete Beta function</i></p> <p>and <math>B(y, z)</math> being <i>Euler's Beta function</i> (see p. 93 and <a href="https://en.wikipedia.org/wiki/Beta_function">https://en.wikipedia.org/wiki/Beta_function</a>).</p>
$I\Gamma_p$	<p>Returns the <i>regularized Gamma function</i></p> $P(x, y) = \frac{\gamma(x, y)}{\Gamma(x)}$ <p>See <math>\gamma_{XY}</math> below for <math>\gamma(x, y)</math> and p. 93 for <math>\Gamma(x)</math>.</p>
$I\Gamma_q$	<p>Returns the <i>regularized Gamma function</i></p> $Q(x, y) = \frac{\Gamma_u(x, y)}{\Gamma(x)}$ <p>See <math>\Gamma_{XY}</math> below for <math>\Gamma_u(x, y)</math> and p. 93 for <math>\Gamma(x)</math>.</p>

See here for more:  
[https://en.wikipedia.org/wiki/Incomplete\\_gamma\\_function](https://en.wikipedia.org/wiki/Incomplete_gamma_function)

Name	Remarks (see pp. 13ff for general information)
$J_y(x)$	<p>Generally, the <i>Bessel functions</i> solve the differential equation</p> $x^2 f''(x) + xf'(x) + (x^2 - \nu^2)f(x) = 0 \quad \text{with } \nu \in \mathbb{C}.$ <p><math>J_y(x)</math> returns the <i>Bessel function of first kind</i> and order <math>y = \nu</math>. For arbitrary <math>\nu</math>, 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 <math>\nu</math>, 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:  <a href="http://en.wikipedia.org/wiki/Bessel_function">http://en.wikipedia.org/wiki/Bessel_function</a>.</p>

Name	Remarks (see pp. 13ff for general information)
$W_p$ , $W_m$	<p>Returns <i>Lambert's W</i> with its principal branch (called <math>W_p</math> here) and its negative branch (called <math>W_m</math> for minus). The connecting point is <math>(-1/e, -1)</math>. The diagram shows the <i>real</i> values of both branches.</p> <p>Start reading here for more information: <a href="http://en.wikipedia.org/wiki/Lambert_W_function">http://en.wikipedia.org/wiki/Lambert_W_function</a>. Learn more here: <a href="http://mathworld.wolfram.com/LambertW-Function.html">http://mathworld.wolfram.com/LambertW-Function.html</a>.</p> 
$\gamma_{xy}$	<p>Returns the <i>lower incomplete Gamma function</i></p> $\gamma(x, y) = \int_0^y t^{x-1} e^{-t} dt .$ Required for $I\Gamma_p$ above.
$\Gamma_{xy}$	<p>Returns the <i>upper incomplete Gamma function</i></p> $\Gamma_u(x, y) = \int_y^\infty t^{x-1} e^{-t} dt .$ Required for $I\Gamma_q$ above.
$\zeta(x)$	<p>Returns <i>Riemann's Zeta</i> for <i>real</i> arguments, with</p> $\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x} \text{ for } x > 1 ,$ and its analytical continuation for $x < 1 :$ $\zeta(x) = 2^x \pi^{x-1} \sin\left(\frac{\pi}{2} x\right) \cdot \Gamma(1-x) \cdot \zeta(1-x).$

Name	Remarks (see pp. 13ff for general information)
	<p>Note the different vertical scales for negative and positive <math>x</math> in the plot overleaf.</p> <p>Look here for more:  <a href="http://mathworld.wolfram.com/RiemannZetaFunction.html">http://mathworld.wolfram.com/RiemannZetaFunction.html</a>.</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. 224ff) is a competent source. And *Mathworld* (quoted on pp. 256ff) may contain more details than you ever want to know. Further references are found at these sites.

## APPENDIX I: INFORMATION FOR ADVANCED USERS

### Recursive Programming

Using local registers allows for creating a subroutine that calls itself recursively. Each invocation deals with its local data only. Of course, the *RPN stack* is global so be careful not to corrupt it.

Below is a recursive implementation of the factorial. It is an **example** for demonstration purposes only, since this routine will neither set the *stack* correctly nor will it work for input greater than some hundred:

LBL 'FACT'

IP

x> 1 ?

GTO 00

1

RTN

LBL 00

LocR 01

STO .00

DEC X

XEQ 'FACT'

RCLx .00

RTN

Assume  $x=4$  when you call FACT. Then it will allocate one local register (**R.00**) and store **4** therein. After decrementing  $x$ , FACT will call itself.

Then FACT<sub>2</sub> will allocate a local register (**R.00<sub>2</sub>**) and store **3** therein. After decrementing  $x$ , FACT will call itself again.

Then FACT<sub>3</sub> will allocate a local register (**R.00<sub>3</sub>**) and store **2** therein. After decrementing  $x$ , FACT will call itself once more.

Then FACT<sub>4</sub> will return to FACT<sub>3</sub> with  $x=1$ . This  $x$  will be multiplied by **r.00<sub>3</sub>** there, returning to FACT<sub>2</sub> with  $x=2$ . This  $x$  will be multiplied by **r.00<sub>2</sub>** there, returning to FACT with  $x=6$ , where it will be multiplied by **r.00** and will finally become 24.

## Building WP 43S Almost from Scratch

### How to build the simulator on Windows:

1. Navigate to <https://www.msys2.org/>. Download and run msys2-x86\_64-yyyymmdd.exe
2. Click **Next**
3. Enter the installation folder **C:\msys2\_64** and click **Next**
4. Tick **Run MSYS2 now** and click **Finish**

The following **commands printed blue** must be executed in the black MSYS2 window:

5. **pacman -Syu** . Confirm each question with ENTER and wait until finished.
6. Close the black window by Alt-F4.
7. Reopen *MSYS2 MinGW 64-bits* (every time you need to open MSYS2, open the 64-bits version).
8. **pacman -Syu** . Confirm each question with ENTER.
9. **pacman -S mingw-w64-x86\_64-gcc git base-devel mingw-w64-x86\_64-gtk3** . Confirm each question with ENTER.
10. **cd ~** to navigate to your home directory.
11. **git clone https://gitlab.com/Over\_score/wp43s.git** to get a local copy of the gitlab source repository on your PC.
12. **cd ~/wp43s** to navigate to the *WP 43S* program sources.
13. **git pull** to get the latest version from gitlab.com.
14. **make mrproper** to clean the build environment (this is not required but recommended).
15. **make** to build the *WP 43S* simulator.
16. **./wp43s** to run the simulator.
17. Return to step 13. (or to *App. F*) to get a new version of the sources.

## How to build the WP43S.pgm for the *DM42* hardware:

1. Navigate to <http://gnutoolchains.com/arm-eabi/>, download and run **arm-eabi-gcc9.2.1.exe** or a more recent version.
2. Installation directory **C:\msys2\_64\arm-eabi** shall be the same base directory as in step 3 on previous page.
3. Select **Current user**
4. Tick **Hardlink duplicate files**
5. Untick **Add binary directory to %PATH%**
6. Tick **I accept the terms of the license agreement**
7. Click **Install**
8. Click **OK** on the Installation succeeded dialog.
9. Start *MSYS2 64 bits* if not yet started.
10. **cd ~/wp43s/DMCP\_build** to navigate to the *DMCP* build directory.
11. **./build\_GMP\_static\_ARM\_library** this is a long process: about 12 minutes on my PC.
12. **cd ~/wp43s/DMCP\_build** to navigate to the *DMCP* build directory if not yet there.
13. **git pull** to get the latest version from [gitlab.com](#)
14. **./build\_WP43S.pgm\_for\_DM42\_hardware** to build *WP43S.pgm* for the *DM42*. Maybe the first time the process ends with an error – then run the same command a second time.
15. Copy the file *~/wp43s/DMCP\_build/build/WP43S.pgm* via *USB* to your *DM42* and flash it (cf. *App. F*).
16. Loop to step 12.

## **Index of Everything Provided**

This index lists each and every item provided, be it a command, constant (C), menu or submenu (M unless trailed by a colon), predefined label (L) or variable (V), reserved character (c), or *system flag* (S). Note that the sorting order in your *WP 43S* deviates for good reasons from the order *MS Word* applied here:

DRAFT

$\hat{x}$	85	$\rightarrow$ RAD	99	ACOS	17	c (C)	23
-	97	$\rightarrow$ REAL	99	ADM (V)	105	C (V)	106
$\bar{x}$	85	$\rightarrow$ REC	99	ADV	17	$c_1$ (C)	23
$\bar{x}_w$	87	$\downarrow$ Lim (V)	109	AGM	17	$c_2$ (C)	23
$\not\rightarrow$ (M)	102	$\not\rightarrow$	99	AGRAPH	18	cal $\rightarrow$ J	23
$\bar{x}_G$	86	$^{\circ}C \rightarrow ^{\circ}F$	16	ALL	18	CARRY (S)	111
#	105	$^{\circ}F \rightarrow ^{\circ}C$	16	ALLSCI (S)	113	CASE	23
#B	105	$\mu_p$ (C)	93	ALP.IN (S)	113	CATALOG	24
#DEC (V)	109	$\mu_0$ (C)	93	ALPHA (S)	111	Cauch $^{-1}$	24
%	100	$\mu_B$ (C)	93	$a_{\text{Moon}}$ (C)	18	Cauch	24
%+MG	101	$\mu_e$ (C)	93	AND	18	Cauch:	24
%MRR	100	$\mu_e/\mu_B$ (C)	93	ANGLES (M)	19	Cauch $_e$	24
%T	101	$\mu_n$ (C)	93	arccos	19	CauchF	24
% $\Sigma$	101	$\mu_p$ (C)	93	arcosh	19	Cauch $_p$	24
(-1) $^x$	96	$\mu_u$ (C)	93	arcsin	19	CB	24
( $\chi^2$ ) $^{-1}$	96	$\not\equiv$ #	105	arctan	19	CEIL	24
/	97	$\not\equiv$ ADV	102	artanh	19	CF	25
[M] $^{-1}$	96	$\not\equiv$ CHAR	103	ASIN	19	CHARS (M)	25
[M] $^T$	96	$\not\equiv$ DLAY	103	ASLIFT (S)	113	CLALL	25
$\wedge$ MOD	97	$\not\equiv$ LCD	103	ASR	20	CLCVAR	25
	100	$\not\equiv$ MODE	103	ASSIGN	20	CLFALL	25
M	100	$\not\equiv$ PROG	103	ATAN	20	CLK (M)	25
x	100	$\not\equiv$ r	103	atm $\rightarrow$ Pa	20	CLLCD	25
+	96	$\not\equiv$ REGS	104	au $\rightarrow$ m	20	CLMENU	25
+/-	96	$\not\equiv$ STK	104	AUTOFF (S)	113	CLP	25
$\pm\infty$ ?	97	$\not\equiv$ TAB	104	AUTXEQ (S)	113	CLPALL	25
x	97	$\not\equiv$ USER	104	B (V)	106	CLR (M)	26
$\times$ MOD	97	$\not\equiv$ WIDTH	104	BACK	20	CLREGS	26
$\sqrt{x}$	101	$\not\equiv$ $\Sigma$	104	bar $\rightarrow$ Pa	21	CLSTK	26
$\sqrt[3]{x}$	17	$\not\equiv$ 1/x	16	BATT?	21	CLX	26
$\sqrt[3]{y}$	87	$\not\equiv$ $10^x$	16	bbl $\rightarrow$ m $^3$	21	CL $\Sigma$	26
$\not\rightarrow$	102	1COMPL	16	BC?	21	CNST	26
$\int$	102	2COMPL	17	BEEP	21	COMB	26
$\int f$ (M)	102	$2^x$	17	BeginP	21	CONJ	26
$\int f dx$ (M)	102	$-\infty$	97	BestF	21	CONST (M)	26
$\uparrow$ Lim (V)	109	$\infty$ (C)	101	BestF?	22	CONVG	27
$\rightarrow$ (c)	97	$a_{\oplus}$ (C)	20	Binom	22	CORR	27
$\rightarrow$ D.MS	98	$a$ (C)	17	Binom $^{-1}$	22	cos	27
$\rightarrow$ DATE	97	A (V)	105	Binom:	22	cosh	27
$\rightarrow$ DEG	98	A... $\Omega$ (M)	91	Binom $_e$	22	COV	27
$\rightarrow$ GRAD	98	A:	20	Binom $_p$	22	CPX (M)	27
$\rightarrow$ H.MS	98	$a_0$ (C)	17	BITS	22	CPX?	28
$\rightarrow$ HR	98	ABS	17	$B_n^*$	23	CPXj (S)	110
$\rightarrow$ INT	98	ac $_{us} \rightarrow$ m $^2$	17	$B_n$	23	CPXRES (S)	110
$\rightarrow$ MUL $\pi$	98	ac $\rightarrow$ m $^2$	17	BS?	23	CPXS (M)	28
$\rightarrow$ POL	98	ACC (V)	105	Btu $\rightarrow$ J	23	CROSS	28

c <sub>t</sub> →kg	28	ENDP	33	FIN (M)	36	h (C)	40	
cwt→kg	28	ENG	33	FIX	36	h̄ (C)	41	
CX→RE	28	ENORM	33	FLAGS (M)	36	ha→m <sup>2</sup>	40	
D (V)	106	ENTER†	33	FLASH (M)	36	H <sub>n</sub>	41	
D.MS	32	ENTRY?	34	FLASH?	36	H <sub>np</sub>	41	
D.MS→	32	EQ.DEL	34	FLOOR	36	hp <sub>UK</sub> →W	41	
D.MS→D	32	EQ.EDI	34	fm.→m	37	hp <sub>E</sub> →W	41	
D.MY	32	EQ.NEW	34	FP	37	hp <sub>M</sub> →W	41	
D→D.MS	32	EQN (M)	34	F <sub>p</sub> (x)	36	Hyper	41	
D→J	32	erf	34	FP?	37	Hyper <sup>-1</sup>	41	
D→R	32	erfc	34	fr→dB	37	Hyper:	41	
DATE	29	ERR	34	FRACT (S)	111	Hyper <sub>e</sub>	41	
DATE→	29	EVEN?	34	FS?	37	Hyper <sub>p</sub>	41	
DATES (M)	29	EXITALL	34	FS?C	37	HypF	41	
DAY	29	EXP (M)	34	FS?F	37	I-	44	
dB→fr	29	ExpF	35	FS?S	37	I (V)	107	
dB→pr	29	Expon <sup>-1</sup>	35	ft.→m	37	i%/a (V)	107	
DBL/	29	Expon	35	ft <sub>US</sub> →m	37	I/O (M)	44	
DBL×	29	Expon:	35	FV (V)	106	I+	44	
DBLR	29	Expon <sub>e</sub>	35	fz <sub>UK</sub> →m <sup>3</sup>	37	IDIV	42	
DEC	30	Expon <sub>p</sub>	35	fz <sub>US</sub> →m <sup>3</sup>	37	IDIVR	42	
DECIM. (S)	112	EXPT	35	G (C)	38	IGNIER (S)	113	
DECOMP	30	e <sup>x</sup>	34	g <sub>⊕</sub>	40	iHg→Pa	42	
DEG	30	e <sup>x</sup> -1	35	G <sub>0</sub> (C)	38	Im	42	
DEG→	30	F	35	GAP	38	in.→m	43	
DENANY (S)	111	F <sub>s</sub>	37	GaussF	38	INC	42	
DENFIX (S)	111	F <sub>a</sub>	37	G <sub>c</sub> (C)	38	INDEX	42	
DENMAX	30	f'	(M)	37	GCD	39	INFO (M)	42
DENMAX (V)	106	f'' (M)	37	g <sub>d</sub>	39	INPUT	42	
DET	30	f''(x)	38	g <sub>d</sub> <sup>-1</sup>	39	INT?	43	
DIGITS (M)	30	F&p:	38	g <sub>e</sub> (C)	39	INTING (S)	113	
DISP (M)	30	F <sup>-1</sup> (p)	36	Geom <sup>-1</sup>	39	INTS (M)	43	
DMY (S)	110	f'(x)	38	Geom	39	INVRT	43	
DOT	31	F(x)	36	Geom:	39	IP	43	
DROP	31	F:	37	Geom <sub>e</sub>	39	ISE	43	
DROPy	31	FB	35	Geom <sub>p</sub>	39	ISG	43	
DSE	31	FBR	35	gl <sub>US</sub> →m <sup>3</sup>	39	ISZ	43	
DSL	31	FC?	35	gl <sub>UK</sub> →m <sup>3</sup>	39	I <sub>xyz</sub>	44	
DSTACK	31	FC?C	36	GM <sub>⊕</sub>	39	IΓ <sub>p</sub>	44	
DSZ	32	FC?F	36	GRAD	40	IΓ <sub>q</sub>	44	
e (C)	33	FC?S	36	GRAD→	40	J-	44	
E:	35	FCNS (M)	35	GRAMOD (V)	107	J (V)	107	
e <sub>E</sub> (C)	33	F <sub>e</sub> (x)	36	GROW (S)	113	J/G	44	
EIGVAL	33	FF	36	GTO	40	J+	44	
EIGVEC	33	FIB	36	GTO.	40	J→Btu	44	
END	33	FILL	36	Ȑ <sub>H</sub>	86	J→cal	44	

J→D	45	LOADR	49	m→ly	56	MULπ	53
J→Wh	45	LOADSS	49	m→mi.	56	MULπ→	53
J <sub>y</sub> (x)	44	LOADΣ	49	m→nmi.	56	MVAR	53
k (C)	45	LocR	49	m→pc	56	MyMenu	53
K (V)	107	LocR?	49	m→pt.	56	Myα (M)	53
KEY	45	LOG <sub>10</sub>	49	m→yd.	56	N→lbf	58
KEY?	45	LOG <sub>2</sub>	49	m <sup>2</sup> →ac <sub>us</sub>	51	N <sub>A</sub>	57
KEYG	45	LogF	49	m <sup>2</sup> →ac	51	NaN (C)	57
KEYX	45	Logis <sup>-1</sup>	50	m <sup>2</sup> →ha	51	NaN?	57
kg→ct	46	Logis	50	m <sup>3</sup> →bbl	51	NAND	57
kg→cwt	46	Logis:	50	m <sup>3</sup> →fz <sub>us</sub>	51	NBin <sup>-1</sup>	57
kg→lb.	46	Logis <sub>e</sub>	50	m <sup>3</sup> →fz <sub>UK</sub>	51	NBin	57
kg→oz	46	Logis <sub>p</sub>	50	m <sup>3</sup> →gl <sub>UK</sub>	51	NBin:	57
kg→s.t	46	LOG <sub>x,y</sub>	50	m <sup>3</sup> →gl <sub>us</sub>	51	NBIn <sub>e</sub>	57
kg→scw	46	LOOP (M)	50	MANT	51	NBIn <sub>p</sub>	57
kg→sto	46	LOWBAT (S)	112	MASKL	51	NEIGHB	57
kg→ton	46	I <sub>PL</sub> (C)	50	MASKR	51	NEXTP	58
kg→trz	46	ly→m	50	Mat_A (V)	107	nmi.→m	58
K <sub>j</sub> (C)	46	m <sub>⊖</sub>	56	Mat_B (V)	107	NOP	58
KTYP?	46	m <sub>⊕</sub>	56	Mat_X	52	NOR	58
L (V)	107	m <sub>μ</sub> (C)	54	Mat_X (V)	107	Norml	58
L.INTS (M)	50	M.DELR	54	MATR?	51	Norml <sup>-1</sup>	58
L.R.	50, 79	M.DIM	54	MATRS (M)	51	Norml:	58
LASTx	47	M.DIM?	54	MATX (M)	51	Normle	58
lb.→kg	47	M.DY	54	max	52	Normlp	58
lbf→N	47	M.EDI	54	MDY (S)	110	NOT	58
LBL	47	M.EDIN	54	m <sub>e</sub> (C)	52	NPER (V)	108
LBL?	47	M.EDIT (M)	54	MEM?	52	NUM.IN (S)	113
LCM	47	M.GET	55	MENU	52	nΣ	58
LEAD.0 (S)	111	M.GOTO	55	MENUS (M)	52	ODD?	59
LEAP?	47	M.GROW	55	mi.→m	52	OFF	59
LgNrnm	47	M.INSR	55	min	52	OR	59
LgNrnm <sup>-1</sup>	47	M.LU	55	MIRROR	52	OrthoF	59
LgNrnm:	48	M.NEW	55	M <sub>Moon</sub> (C)	52	ORTHOG (M)	59
LgNrnm <sub>e</sub>	47	M.OLD	55	m <sub>n</sub> (C)	52	OVERFL (S)	111
LgNrnm <sub>p</sub>	47	M.PUT	55	m <sub>n</sub> /m <sub>p</sub> (C)	52	oz→kg	59
LinF	48	M.RZR	56	MOD	53	P.FN (M)	62
LJ	48	M.SIMQ (M)	56	MODE (M)	53	P.FN2 (M)	62
L <sub>ma</sub>	48	M.SQR?	56	MONTH	53	P:	62
L <sub>m</sub>	48	M.WRAP	56	m <sub>p</sub> (C)	53	P <sub>0</sub> (C)	59
LN	48	m:	56	m <sub>p</sub> /m <sub>e</sub> (C)	53	Pa→atm	59
LN(1+x)	48	m→au	56	m <sub>PL</sub> (C)	53	Pa→bar	59
LNβ	48	m→fm.	56	MSG	53	Pa→iHg	59
LNΓ	48	m→ft <sub>us</sub>	56	m <sub>u</sub> (C)	53	Pa→psi	59
LOAD	48	m→ft.	56	m <sub>u</sub> c <sup>2</sup> (C)	53	Pa→tor	59
LOADP	49	m→in.	56	MULTx (S)	112	ParabF	59

PARTS (M)	59	RANGE	63	RSUM	69	S <sub>mw</sub>	76
PAUSE	59	RANGE?	63	R-SWAP	71	SNAP	76
pc→m	60	RANI#	63	RTN	69	SOLVE	76
PER/a (V)	108	RBR	64	RTN+1	70	Solver (M)	77
PERM	60	RCL	64	RUNIO (S)	112	SOLVING (S)	113
PGMINT	60	RCL-	64	RUNTIM (S)	112	SPCRES(S)	112
PGMSLV	60	RCL/	64	s	71	SPEC?	77
PIXEL	60	RCL+	64	S.INTS (M)	79	SR	77
PLOT	60	RCLx	64	s.t→kg	79	SSIZE?	77
PMT (V)	108	RCL↑	65	s→year	79	SSIZE8 (S)	112
P <sub>n</sub>	60	RCL↓	65	Sa (C)	72	ST.A (V)	109
POINT	60	RCLCFG	64	SAVE	72	ST.B (V)	109
Poiss <sup>-1</sup>	61	RCLEL	64	SB	72	ST.C (V)	109
Poiss	61	RCLIJ	64	Sb (C)	72	ST.D (V)	109
Poiss:	61	R-CLR	70	SCI	72	ST.X (V)	109
Poisse	61	RCLS	64	scw→kg	72	ST.Z (V)	109
Poiss <sub>p</sub>	61	R-COPY	70	SDIGS?	72	ST.T (V)	109
PopLR	61	RDP	65	SDL	72	ST.Y (V)	109
PowerF	61	Re	65	SDR	72	STAT (M)	77
pr→dB	62	r <sub>e</sub> (C)	65	Se <sup>2</sup> (C)	72	STATUS	77
PRCL	61	RE→CX	66	Se' <sup>2</sup> (C)	73	STK (M)	77
PRIME?	61	Re $\nexists$ Im	66	SEED	73	STO	77
PRINT (M)	61	REAL?	65	SEND	73	STO-	78
PRINT (S)	112	REALDF (V)	108	SETCHN	73	STO/	78
PROB (M)	61	REALS (M)	65	SETDAT	73	STO+	78
PROG (M)	62	RECTN (S)	110	SETEUR	73	STO×	78
PROGS (M)	62	RECV	65	SETIND	73	STO↑	78
PROPFR (S)	111	REGS (V)	108	SETJPN	73	sto→kg	78
PRTACT (S)	113	RESET	66	SETSIG	73	STO↓	78
psi→Pa	62	RJ	66	SETTIM	73	STOCFG	77
PSTO	62	R <sub>k</sub> (C)	66	SETUK	73	STOEL	77
pt.→m	62	RL	67	SETUSA	73	STOIJ	78
PUTK	62	RLC	67	SF	74	STOP	78
PV (V)	108	RM	67	Sf <sup>-1</sup> (C)	74	STOS	78
QUIET (S)	112	RM?	68	SHOW	74	STRI?	78
R <sub>∞</sub> (C)	71	RMD	68	SIGN	74	STRING (M)	78
R <sub>⊕</sub> (C)	71	R <sub>Moon</sub> (C)	68	SIGNMT	74	SUM	79
R (C)	63	̄x <sub>RMS</sub>	87	SIM_EQ	74	s <sub>w</sub>	79
R <sub>⊖</sub> (C)	71	RNORM	68	sin	74	s <sub>xy</sub>	79
R↑	71	RootF	68	sinc	75	SYS.FL (M)	79
R→D	71	ROUND	68	sinh	75	SYSTEM	79
R↓	71	ROUNDI	68	SKIP	75	t <sup>-1</sup> (p)	80
RAD	63	RR	69	SL	75	t(x)	80
RAD→	63	RRC	69	SLOW (S)	112	t:	82
RAM (M)	63	RSD	69	s <sub>m</sub>	76	t $\nexists$	82
RAN#	63	R-SORT	71	SMODE?	76	T <sub>0</sub>	80

$\tan$	80	Weibl <sub>e</sub>	84	$\alpha$ INTL (M)	90	$\Sigma x^2$	94
tanh	80	Weibl <sub>p</sub>	84	$\alpha$ LEN <sup>G?</sup>	90	$\Sigma x^2/y$	94
TDISP	80	Wh $\rightarrow$ J	84	$\alpha$ MATH (M)	90	$\Sigma x^2y$	94
TDM24 (S)	110	WHO?	84	$\alpha$ POS?	90	$\Sigma x^3$	95
$t_e(x)$	80	$W_m$	84	$\alpha$ RL	90	$\Sigma x^4$	95
TEST (M)	80	$W_p$	84	$\alpha$ RR	90	$\Sigma x \ln y$	95
TICKS	80	WSIZE	85	$\alpha$ SL	90	$\Sigma xy$	95
TIME	81	WSIZE?	85	$\alpha$ SR	91	$\Sigma y$	95
TIMER	81	$x < ?$	88	$\gamma$ (C)	91	$\Sigma y^2$	95
TIMES (M)	81	$x = ?$	88	$\Gamma(x)$	91	$\Sigma y \ln x$	95
$T_n$	81	$x \neq ?$	88	$\gamma_E$ (C)	91	$\Phi$ (C)	95
ton $\rightarrow$ kg	81	$x = -0?$	88	$\gamma_p$ (C)	91	$\Phi_0$ (C)	95
TONE	81	$x = +0?$	88	$\gamma_{xy}$	91	$\chi^2(x)$	96
TOP?	81	$x > ?$	88	$\Gamma_{xy}$	91	$\chi^2:$	96
tor $\rightarrow$ Pa	81	$x \approx ?$	88	$\Delta\%$	92	$\chi^2_e(x)$	96
$T_p$ (C)	81	$x \leq ?$	88	$\delta x$ (L)	92	$\chi^2_p(x)$	96
$t_p(x)$	80	$x \geq ?$	88	$\Delta v_{Cs}$ (C)	92	$\omega$ (C)	96
$t_{PL}$ (C)	81	$x!$	87	$\varepsilon$	92		
TRACE (S)	112	X.FN (M)	87	$\varepsilon_0$ (C)	92		
TRANS	81	$x:$	87	$\varepsilon_m$	92		
TRI (M)	81	$x \rightarrow$ DATE	87	$\varepsilon_p$	92		
trz $\rightarrow$ kg	82	$x \rightarrow \alpha$	88	$\zeta(x)$	92		
TVM (M)	82	$x \gtrless$	88	$\lambda_C$ (C)	93		
$U \rightarrow$ (M)	82	$x \gtrless y$	88	$\lambda_{Cn}$ (C)	93		
ULP?	82	$x^2$	85	$\lambda_{CP}$ (C)	93		
$U_n$	82	$x^3$	85	$\pi$ (C)	93		
UNITV	82	XEQ	86	$\Pi_n$	93		
UNSIGN	82	XIN	86	$\sigma$	93		
USER (S)	112	XNOR	86	$\Sigma-$	95		
V:	83	XOR	86	$\Sigma$ (M)	93		
V $\downarrow$	83	XOUT	86	$\Sigma 1/x$	94		
VAR (M)	83	$\hat{y}$	89	$\Sigma 1/x^2$	94		
VARMNU	83	Y.MD	89	$\Sigma 1/y$	94		
VARS (M)	83	$y \gtrless$	89	$\Sigma 1/y^2$	94		
VERS?	83	yd. $\rightarrow$ m	89	$\Sigma +$	95		
VIEW	83	YEAR	89	$\sigma_B$ (C)	93		
$V_m$ (C)	83	year $\rightarrow$ s	89	$\Sigma \ln^2 x$	94		
VMDISP (S)	113	YMD (S)	110	$\Sigma \ln^2 y$	94		
$W^{-1}$	84	$y^x$	89	$\Sigma \ln x$	94		
$W \rightarrow h_{PUK}$	85	$z \gtrless$	89	$\Sigma \ln xy$	94		
$W \rightarrow h_{PE}$	85	$Z_0$ (C)	89	$\Sigma \ln y$	94		
$W \rightarrow h_{PM}$	85	$\alpha$ (C)	90	$\Sigma \ln y/x$	94		
WDAY	84	$\alpha \cdot$ (M)	91	$\Sigma_n$	94		
Weibl	84	$\alpha$ .FN (M)	91	$\sigma_w$	94		
Weibl $^{-1}$	84	$\alpha \rightarrow x$	91	$\Sigma x$	94		
Weibl:	84	$\alpha$ CAP (S)	112	$\Sigma x/y$	95		

The same items sorted as they are in your WP 43S:

$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	16	ASSIGN	19	Cauch	24	CPXS (M)	27
$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$	16	ATAN	19	Cauch <sub>e</sub>	24	CPX?	28
$10^x$	16	atm $\rightarrow$ Pa	20	CauchF	24	CROSS	28
1COMPL	16	AUTOFF (S)	112	Cauch <sub>p</sub>	24	ct $\rightarrow$ kg	28
$1/x$	16	AUTXEQ (S)	112	Cauch <sup>-1</sup>	24	cwt $\rightarrow$ kg	28
2COMPL	16	au $\rightarrow$ m	20	Cauch:	24	CX $\rightarrow$ RE	28
$2^x$	16	A:	20	CB	24	D (V)	105
$\sqrt[3]{x}$	16	a <sub>⊕</sub> (C)	20	CEIL	24	DATE	28
A (V)	105	B (V)	105	CF	24	DATES (M)	28
a (C)	17	BACK	20	CHARS (M)	24	DATE $\rightarrow$	29
a <sub>0</sub> (C)	17	bar $\rightarrow$ Pa	20	CLALL	24	DAY	29
ABS	17	BATT?	20	CLCVAR	25	DBLR	29
ACC (V)	105	bbl $\rightarrow$ m <sup>3</sup>	20	CLFALL	25	DBL $\times$	29
ACOS	17	BC?	20	CLK (M)	25	DBL/	29
ac <sub>us</sub> $\rightarrow$ m <sup>2</sup>	17	BEEP	21	CLLCD	25	dB $\rightarrow$ fr	29
ac $\rightarrow$ m <sup>2</sup>	17	BeginP	21	CLMENU	25	dB $\rightarrow$ pr	29
ADM (V)	105	BestF	21	CLP	25	DEC	29
ADV	17	Binom	22	CLPALL	25	DECIM. (S)	112
AGM	17	Binom <sub>e</sub>	22	CLR (M)	25	DECOMP	30
AGRAPH	17	Binom <sub>p</sub>	22	CLREGS	25	DEG	30
ALL	17	Binom <sup>-1</sup>	22	CLSTK	25	DEG $\rightarrow$	30
ALLSCI (S)	112	Binom:	22	CLX	26	DENANY (S)	110
ALP.IN (S)	113	BITS	22	CLΣ	26	DENFIX (S)	111
ALPHA (S)	111	B <sub>n</sub>	22	CNST	26	DENMAX	30
a <sub>Moon</sub> (C)	18	B <sub>n</sub> *	22	COMB	26	DENMAX (V)	105
AND	18	BS?	22	CONJ	26	DET	30
ANGLES (M)	18	Btu $\rightarrow$ J	22	CONST (M)	26	DIGITS (M)	30
arccos	18	C (V)	105	CONVG	26	DISP (M)	31
arcosh	18	c (C)	22	CORR	27	DMY (S)	110
arcsin	18	c <sub>1</sub> (C)	22	cos	27	DOT	30
arctan	19	c <sub>2</sub> (C)	22	cosh	27	DROP	31
artanh	19	cal $\rightarrow$ J	22	COV	27	DROPy	31
ASIN	19	CARRY (S)	111	CPX (M)	27	DSE	31
ASLIFT (S)	113	CASE	23	CPXj (S)	110	DSL	31
ASR	19	CATALOG	23	CPXRES (S)	110	DSTACK	31

DSZ	31	F	35	f'(x)	37	Hyper <sub>e</sub>	41
D.MS	32	FB	35	f''(M)	37	Hyper <sub>p</sub>	41
D.MS→	32	FBR	35	f''(x)	37	Hyper <sup>-1</sup>	41
D.MS→D	32	FC?	35	F&p:	37	Hyper:	41
D.MY	32	FC?C	35	F:	37	HypF	41
D→D.MS	32	FC?F	35	G(C)	38	I(V)	106
D→J	32	FC?S	35	G <sub>0</sub> (C)	38	IDIV	41
D→R	32	FCNS (M)	35	GAP	38	IDIVR	41
e (C)	32	F <sub>e</sub> (x)	35	GaussF	38	IGN1ER (S)	113
e <sub>E</sub> (C)	32	FF	36	G <sub>C</sub> (C)	38	iHg→Pa	41
EIGVAL	32	FIB	36	GCD	38	Im	41
EIGVEC	32	FILL	36	g <sub>d</sub>	38	INC	42
END	33	FIN (M)	36	g <sub>d</sub> <sup>-1</sup>	38	INDEX	42
ENDP	33	FIX	36	g <sub>e</sub> (C)	38	INFO (M)	42
ENG	33	FLAGS (M)	36	Geom	39	INPUT	42
ENORM	33	FLASH (M)	36	Geom <sub>e</sub>	39	INT?	42
ENTER↑	33	FLASH?	36	Geom <sub>p</sub>	39	INTING (S)	113
ENTRY?	33	FLOOR	36	Geom <sup>-1</sup>	39	INTS (M)	42
EQN (M)	33	fm.→m	36	Geom:	39	INVRT	42
EQ.DEL	34	FP	36	gl <sub>us</sub> →m <sup>3</sup>	39	in.→m	42
EQ.EDI	34	F <sub>p</sub> (x)	35	gl <sub>uk</sub> →m <sup>3</sup>	39	IP	43
EQ.NEW	34	FP?	36	GM <sub>⊕</sub>	39	ISE	43
erf	34	fr→dB	36	GRAD	39	ISG	43
erfc	34	fract (S)	110	GRAD→	39	ISZ	43
ERR	34	FS?	37	GRAMOD (V)	106	I <sub>xyz</sub>	43
EVEN?	34	FS?C	37	GROW (S)	112	IΓ <sub>p</sub>	43
EXITALL	34	FS?F	37	GTO	39	IΓ <sub>q</sub>	43
EXP (M)	34	FS?S	37	GTO.	40	I+	43
ExpF	34	ft.→m	37	g <sub>⊕</sub>	40	I-	43
Expon	34	ft <sub>us</sub> →m	37	h(C)	40	I/O (M)	43
Expon <sub>e</sub>	34	FV (V)	106	ℏ(C)	41	i%/a (V)	106
Expon <sub>p</sub>	34	fz <sub>uk</sub> →m <sup>3</sup>	37	ha→m <sup>2</sup>	40	J (V)	106
Expon <sup>-1</sup>	34	fz <sub>us</sub> →m <sup>3</sup>	37	H <sub>n</sub>	40	J <sub>y</sub> (x)	44
Expon:	35	F <sub>α</sub>	37	H <sub>nP</sub>	40	J+	44
EXPT	35	F <sub>δ</sub>	37	hp <sub>E</sub> →W	40	J-	44
e <sup>x</sup>	34	F(x)	35	hp <sub>M</sub> →W	40	J/G	44
e <sup>x</sup> -1	35	F <sup>-1</sup> (p)	35	hp <sub>uk</sub> →W	40	J→Btu	44
E:	35	f' (M)	37	Hyper	41	J→cal	44

J>D	44	LNB	48	MATX (M)	51	M.EDIN	54
J>Wh	44	LNF	48	Mat_A (V)	107	M.EDIT (M)	54
K (V)	106	LN(1+x)	48	Mat_B (V)	107	M.GET	54
k (C)	44	LOAD	48	Mat_X	51	M.GOTO	54
KEY	44	LOADP	48	Mat_X (V)	107	M.GROW	54
KEYG	45	LOADR	48	max	51	M.INSR	54
KEYX	45	LOADSS	48	MDY (S)	110	M.LU	55
KEY?	45	LOADΣ	48	m_e (C)	51	M.NEW	55
kg→ct	45	LocR	48	MEM?	51	M.OLD	55
kg→cwt	45	LocR?	49	MENU	51	M.PUT	55
kg→lb.	45	LOG <sub>10</sub>	49	MENUS (M)	51	M.RZR	55
kg→oz	45	LOG <sub>2</sub>	49	min	52	M.SIMQ (M)	55
kg→scw	45	LogF	49	MIRROR	52	M.SQR?	55
kg→sto	45	Logis	49	mi.→m	52	M.WRAP	55
kg→s.t	45	Logis <sub>e</sub>	49	M <sub>Moon</sub> (C)	52	m:	55
kg→ton	45	Logis <sub>p</sub>	49	m <sub>n</sub> (C)	52	m→au	56
kg→trz	45	Logis <sup>-1</sup>	49	m <sub>n</sub> /m <sub>p</sub> (C)	52	m→fm.	56
K <sub>J</sub> (C)	45	Logis:	49	MOD	52	m→ft <sub>us</sub>	56
KTYP?	46	LOG <sub>x,y</sub>	49	MODE (M)	52	m→ft.	56
L (V)	106	LOOP (M)	49	MONTH	52	m→in.	56
LASTx	46	LOWBAT (S)	111	m <sub>p</sub> (C)	52	m→ly	56
lb.→kg	46	l <sub>PL</sub> (C)	49	m <sub>PL</sub> (C)	52	m→mi.	56
lbf→N	46	ly→m	49	m <sub>p</sub> /m <sub>e</sub> (C)	52	m→nmi.	56
LBL	46	L.INTS (M)	49	MSG	52	m→pc	56
LBL?	46	L.R.	50	m <sub>u</sub> (C)	53	m→pt.	56
LCM	46	m <sup>2</sup> →ac	50	m <sub>u</sub> c <sup>2</sup> (C)	53	m→yd.	56
LEAD.0 (S)	111	m <sup>2</sup> →ac <sub>us</sub>	50	MULTx (S)	112	m <sub>⊕</sub>	56
LEAP?	47	m <sup>2</sup> →ha	50	MULπ	53	m <sub>⊕</sub>	56
LgNrm	47	m <sup>3</sup> →bbl	50	MULπ→	53	N <sub>A</sub>	56
LgNrm <sub>e</sub>	47	m <sup>3</sup> →fz <sub>UK</sub>	50	MVAR	53	NaN (C)	56
LgNrm <sub>p</sub>	47	m <sup>3</sup> →fz <sub>us</sub>	50	MyMenu	53	NAND	56
LgNrm <sup>-1</sup>	47	m <sup>3</sup> →gl <sub>UK</sub>	50	Myα (M)	53	NaN?	56
LgNrm:	47	m <sup>3</sup> →gl <sub>us</sub>	50	m <sub>μ</sub> (C)	53	NBin	56
LinF	47	MANT	50	M.DELR	53	NBin <sub>e</sub>	56
LJ	47	MASKL	51	M.DIM	53	NBin <sub>p</sub>	56
L <sub>m</sub>	47	MASKR	51	M.DIM?	53	NBin <sup>-1</sup>	56
L <sub>ma</sub>	47	MATRS (M)	51	M.DY	54	NBin:	57
LN	47	MATR?	51	M.EDI	54	NEIGHB	57

NEXTP	57	P <sub>n</sub>	60	RCLCFG	63	RTN	68
nmi. $\rightarrow$ m	57	POINT	60	RCLEL	63	RTN+1	68
NOP	57	Poiss	60	RCLIJ	63	RUNIO (S)	111
NOR	57	Poiss <sub>e</sub>	60	RCLS	63	RUNTIM (S)	111
Norml	57	Poiss <sub>p</sub>	60	RCL+	64	R-CLR	69
Normle	57	Poiss <sup>-1</sup>	60	RCL-	64	R-COPY	69
Normlp	57	Poiss:	60	RCLx	64	R-SORT	69
Norml <sup>-1</sup>	57	PopLR	60	RCL/	64	R-SWAP	70
Norml:	58	PowerF	60	RCL $\uparrow$	64	R <sub><math>\infty</math></sub> (C)	70
NOT	58	PRCL	61	RCL $\downarrow$	64	R <sub><math>\oplus</math></sub> (C)	70
NPER (V)	107	PRIME?	61	RDP	64	R <sub><math>\odot</math></sub> (C)	70
NUM.IN (S)	113	PRINT (M)	61	Re	64	R $\rightarrow$ D	70
nΣ	58	PRINT (S)	111	r <sub>e</sub> (C)	64	R $\uparrow$	70
N $\rightarrow$ lbf	58	PROB (M)	61	REAL?	64	R $\downarrow$	70
ODD?	58	PROG (M)	61	REALDF (V)	107	s	70
OFF	58	PROGS (M)	61	REALS (M)	64	Sa (C)	70
OR	58	PROPFR (S)	110	RECTN (S)	110	SAVE	70
OrthoF	58	PRTACT (S)	113	RECV	64	SB	71
ORTHOG (M)	58	pr $\rightarrow$ dB	61	REGS (V)	108	Sb (C)	71
OVERFL (S)	111	psi $\rightarrow$ Pa	61	RESET	65	SCI	71
oz $\rightarrow$ kg	58	PSTO	61	RE $\rightarrow$ CX	65	scw $\rightarrow$ kg	71
P <sub>0</sub> (C)	59	pt. $\rightarrow$ m	62	Re $\gtrless$ Im	65	SDIGS?	71
ParabF	59	PUTK	62	RJ	65	SDL	71
PARTS (M)	59	PV (V)	107	R <sub>K</sub> (C)	65	SDR	71
PAUSE	59	P.FN (M)	62	RL	66	Se <sup>2</sup> (C)	71
Pa $\rightarrow$ atm	59	P.FN2 (M)	62	RLC	66	SEED	71
Pa $\rightarrow$ bar	59	P:	62	RM	66	SEND	71
Pa $\rightarrow$ iHg	59	QUIET (S)	112	RMD	67	SETCHN	71
Pa $\rightarrow$ psi	59	R (C)	62	R <sub>Moon</sub> (C)	67	SETDAT	72
Pa $\rightarrow$ tor	59	RAD	62	RM?	67	SETEUR	72
pc $\rightarrow$ m	59	RAD $\rightarrow$	62	RNORM	67	SETIND	72
PERM	59	RAM (M)	62	RootF	67	SETJPN	72
PER/a (V)	107	RANGE	62	ROUND	67	SETSIG	72
PGMINT	60	RANGE?	62	ROUNDI	67	SETTIM	72
PGMSLV	60	RAN#	63	RR	67	SETUK	72
PIXEL	60	RANI#	62	RRC	68	SETUSA	72
PLOT	60	RBR	63	RSD	68	Se' <sup>2</sup> (C)	72
PMT (V)	107	RCL	63	RSUM	68	SF	72

Sf <sup>-1</sup> (C)	72	sto→kg	77	t <sub>p</sub> (x)	79	W→hp <sub>UK</sub>	84
SHOW	73	STO↑	77	TRACE (S)	111	W→hp <sub>E</sub>	84
SIGN	73	STRING (M)	77	TRANS	81	W→hp <sub>M</sub>	84
SIGNMT	73	STRI?	78	TRI (M)	81	ŷ	84
SIM_EQ	73	ST.A (V)	108	trz→kg	81	ȳ	84
sin	73	ST.B (V)	108	TVM (M)	81	x <sup>2</sup>	84
sinc	73	ST.C (V)	108	t <sup>-1</sup> (p)	79	x <sup>3</sup>	84
sinh	74	ST.D (V)	108	t(x)	79	XEQ	85
SKIP	74	ST.T (V)	108	t:	81	ȳ <sub>G</sub>	85
SL	74	ST.X (V)	108	tꝫ	81	ȳ <sub>H</sub>	85
slow (S)	111	ST.Y (V)	108	ULP?	81	xIN	85
s <sub>m</sub>	75	ST.Z (V)	108	U <sub>n</sub>	81	XNOR	85
SMODE?	75	SUM	78	UNITY	81	XOR	85
s <sub>mw</sub>	75	s <sub>w</sub>	78	UNSIGN	81	xOUT	85
SNAP	76	s <sub>xy</sub>	78	USER (S)	111	ȳ <sub>RMS</sub>	86
SOLVE	76	SYSTEM	78	U→ (M)	81	ȳ <sub>w</sub>	86
Solver (M)	76	SYS.FL (M)	78	VAR (M)	82	ȳȳ	86
SOLVING (S)		S.INTS (M)	78	VARMNU	82	x!	86
112		s.t→kg	78	VARS (M)	82	X.FN (M)	86
SPCRES (S)	111	s→year	78	VERS?	82	x:	86
SPEC?	76	T <sub>0</sub>	79	VIEW	82	x→DATE	86
SR	76	tan	79	V <sub>m</sub> (C)	82	x→α	87
SSIZE8 (S)	112	tanh	79	VMDISP (S)	113	ȳȳ	87
SSIZE?	76	TDISP	79	V:	82	xȳy	87
STAT (M)	76	TDM24 (S)	109	V <sub>4</sub>	82	x < ?	87
STATUS	76	TEST (M)	79	WDAY	83	x ≤ ?	87
STK (M)	77	t <sub>e</sub> (x)	79	Weibl	83	x = ?	87
STO	77	TICKS	79	Weibl <sub>e</sub>	83	x ≠ ?	87
STOCFG	77	TIME	80	Weibl <sub>p</sub>	83	x ≈ ?	87
STOEL	77	TIMER	80	Weibl <sup>-1</sup>	83	x ≥ ?	87
STOIJ	77	TIMES (M)	80	Weibl:	83	x > ?	87
STOP	77	T <sub>n</sub>	80	WHO?	83	x = +0?	87
STOS	77	TONE	80	Wh→J	83	x = -0?	87
STO+	77	ton→kg	80	W <sub>m</sub>	83	ŷ	88
STO-	77	TOP?	80	W <sub>p</sub>	83	yd.→m	88
STO×	77	tor→Pa	80	WSIZE	84	YEAR	88
STO/	77	T <sub>p</sub> (C)	80	WSIZE?	84	year→s	88
STO†	77	t <sub>PL</sub> (C)	80	W <sup>-1</sup>	83	ymd (S)	109

$y^x$	88	$\lambda_{CP}$ (C)	92	$\Sigma \ln x$	94	$\uparrow \text{Lim}$ (V)	109
$Y.MD$	88	$\mu_0$ (C)	92	$\Sigma +$	94	$\downarrow \text{Lim}$ (V)	109
$y^z$	88	$\mu_B$ (C)	92	$\Sigma -$	94	$\gtrless$	99
$Z_0$ (C)	88	$\mu_e$ (C)	92	$\phi$ (C)	95	$ M $	99
$z^z$	88	$\mu_e/\mu_B$ (C)	92	$\Phi_0$ (C)	95	$ x $	99
$\alpha$ (C)	89	$\mu_n$ (C)	92	$\chi^2_e(x)$	95	$  $	100
$\alpha\text{CAP}$ (S)	111	$\mu_p$ (C)	92	$\chi^2_p(x)$	95	$\%$	100
$\alpha\text{INTL}$ (M)	89	$\mu_u$ (C)	92	$\chi^2(x)$	95	$\%MRR$	100
$\alpha\text{LENG?}$	89	$\mu_\mu$ (C)	92	$\chi^2:$	95	$\%T$	100
$\alpha\text{MATH}$ (M)	89	$\pi$ (C)	92	$\omega$ (C)	95	$\%Z$	100
$\alpha\text{POS?}$	89	$\Pi_n$	92	$(\chi^2)^{-1}$	95	$\%+MG$	100
$\alpha\text{RL}$	89	$\Sigma$ (M)	93	$(-1)^x$	95	$\sqrt{x}$	101
$\alpha\text{RR}$	89	$\sigma$	93	$[\mathbf{M}]^T$	95	$\int$	101
$\alpha\text{SL}$	89	$\Sigma 1/x$	93	$[\mathbf{M}]^{-1}$	95	$\int f(M) dx$	101
$\alpha\text{SR}$	90	$\Sigma 1/x^2$	93	$+ 96$		$\int f dx$ (M)	101
$\alpha \cdot$ (M)	90	$\Sigma 1/y$	93	$+/- 96$		$\infty$ (C)	101
$\alpha.FN$ (M)	90	$\Sigma 1/y^2$	93	$\pm\infty?$	96	$\not\equiv$	101
$A \dots \Omega$ (M)	90	$\sigma_B$ (C)	93	$- 96$		$\not\rightarrow$ (M)	102
$\alpha \rightarrow x$	90	$\Sigma \ln^2 x$	93	$-\infty 96$		$\blacksquare \text{ADV}$	102
$\beta(x,y)$	90	$\Sigma \ln^2 y$	93	$\times 96$		$\blacksquare \text{CHAR}$	102
$\gamma$ (C)	90	$\Sigma \ln x$	93	$\times \text{MOD} 96$		$\blacksquare \text{DLAY}$	102
$\gamma_{EM}$ (C)	90	$\Sigma \ln xy$	93	$/ 96$		$\blacksquare \text{LCD}$	102
$\gamma_p$ (C)	90	$\Sigma \ln y$	93	$^{\wedge} \text{MOD} 96$		$\blacksquare \text{MODE}$	102
$\Gamma_{xy}$	90	$\Sigma \ln y/x$	93	$\rightarrow (c) 97$		$\blacksquare \text{PROG}$	102
$\gamma_{xy}$	90	$\Sigma_n$	93	$\rightarrow \text{DATE} 97$		$\blacksquare r$	103
$\Gamma(x)$	91	$\sigma_w$	93	$\rightarrow \text{DEG} 97$		$\blacksquare \text{REGS}$	103
$\delta x$ (L)	91	$\Sigma x$	94	$\rightarrow \text{D.MS} 97$		$\blacksquare \text{STK}$	103
$\Delta v_{Cs}$ (C)	91	$\Sigma x^2$	94	$\rightarrow \text{GRAD} 97$		$\blacksquare \text{TAB}$	103
$\Delta\%$	91	$\Sigma x^2 y$	94	$\rightarrow \text{HR} 97$		$\blacksquare \text{USER}$	103
$\varepsilon$	91	$\Sigma x^2/y$	94	$\rightarrow \text{H.MS} 97$		$\blacksquare \text{WIDTH}$	104
$\varepsilon_0$ (C)	91	$\Sigma x^3$	94	$\rightarrow \text{INT} 97$		$\blacksquare \Sigma$	104
$\varepsilon_m$	91	$\Sigma x^4$	94	$\rightarrow \text{MUL}\pi 97$		$\blacksquare \#$	104
$\varepsilon_p$	92	$\Sigma x \ln y$	94	$\rightarrow \text{POL} 98$		$\#$	104
$\zeta(x)$	92	$\Sigma xy$	94	$\rightarrow \text{RAD} 98$		$\#B$	104
$\lambda_c$ (C)	92	$\Sigma y$	94	$\rightarrow \text{REAL} 98$		$\#DEC$ (V)	109
$\lambda_{ch}$ (C)	92	$\Sigma y^2$	94	$\rightarrow \text{REC} 98$			

## APPENDIX J: RELEASE NOTES

	Date	Release notes
0	29.11.12	Official project start with first publication of the 43S concept and a layout on one of the forums of the <i>Museum of HP Calculators</i> ( <a href="https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234685#234685">https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234685#234685</a> ). Though there are found 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 WP 34S. 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 <i>Jake's</i> 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 <sub>w</sub> to $s_{mw}$ ; refined the <i>Key Response Table</i> . Passed to <i>Michael</i> for information.
0.4	28.3.16	Renamed LOGS to EXP and <b>EEX</b> to <b>E</b> . Added hardware information from 2 <sup>nd</sup> manufacturer.
0.5	29.10.16	Returned <b>EEX</b> . <b>Changed keyboard layout</b> .
0.6	22.8.17	Merged the Applications and Owner's Manual. Changed the input order of complex number parts on Pauli's request. <b>Changed keyboard layout introducing D.MS, SST, BST, and % while removing ÿ, RAN#, 'FRC, and 'CFIT</b> . Put 'CFIT into 'STAT and 'FRC into 'MODE. Placed OFF below EXIT for easier customizing. Renamed cc to C5, <b>EEX</b> to <b>E</b> , 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.
0.7		<b>Changed keyboard layout. Replaced the labels BST by <math>\triangleleft\triangleright</math>, SST by <math>\square\triangleright</math>, and UNDO by <math>\square\triangleleft</math>; added some alpha input mode reminders on the keyboard.</b> Added AGRAPH, CLLCD, EQ.xxx, HYP, J/G, M.GOTO, ORTHOF, PIXEL, POINT, TDISP, and $\blacksquare$ USER. Moved the background considerations out of <i>ReM App. D</i> . Introduced <b>K</b> 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 HP-42S.

	Date	Release notes
	2.4.18	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 <u>SUMS</u> in <u>STAT</u> . Renamed the trigonometric and hyperbolic functions according to mathematical standards, and $\blacksquare\text{CHR}$ to $\blacksquare\text{CHAR}$ . 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 HP-42S.
0.8	7.5.18	Changed keyboard layout: introduced <u>TRG</u> containing trigonometric functions, removed <u>HYP</u> into <u>EXP</u> and $\blacksquare\text{H}$ to g-shifted $\blacksquare\text{J}$ , 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.
	20.9.18	Corrected errors and inconsistencies. Added one more example. Moved the key response table into an appendix.
0.9	3.1.19	Removed <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 functionalities of <u>EXIT</u> and <u><math>\text{V}\text{x}</math></u> to match HP-42S. Added a chapter about curve fitting. Modified functionalities of <u>ENTER</u> and <u><math>\text{G}</math></u> . Expanded <i>App. K</i> . Renamed DOUBLE to $\rightarrow\text{DP}$ . Added $\rightarrow\text{SP}$ and conversions of quarts. Rearranged <u>X.FN</u> . Replaced <u>USR</u> by <u>UM</u> . Changed keyboard moving <u>UM</u> , <u><math>\text{X}</math></u> , and <u>TRI</u> . Moved $\blacksquare$ to $\blacksquare\text{f}$ <u>R/S</u> . 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.
0.10	3.3.19	Returned <i>angle data type</i> and $\alpha\text{SR}$ . Added IDIVR and VANGLE. Refined FP, IP, IMPFRC, PROFRC, SDIGS?, $\rightarrow\text{DP}$ , $\rightarrow\text{HR}$ , $\rightarrow\text{INT}$ , $\rightarrow\text{REAL}$ , $\rightarrow\text{SP}$ , 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><math>\alpha</math></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. .

	Date	Release notes
0.11	8.5.19	Changed keyboard making <b>CC</b> primary and user mode shifted, removing $x^2$ , $x\sqrt{x}$ , and DSP, adding $ x $ , DROP, and SHOW, and moving some shifted labels. Modified <u>BITS</u> , CLREGS, <u>CNST</u> , <u>CPX</u> , <u>DISP</u> , <u>EXP</u> , <u>INTS</u> , <u>MODE</u> , <u>PARTS</u> , SHOW, STAT, $\underline{U}\rightarrow$ , <u>aMATH</u> , the division matrix, <i>data type</i> conversions, and the <i>Quick Reference Guide</i> . Added conversions of <i>barrels</i> , <i>carats</i> , and <i>fathoms</i> . Deleted DSP. – Separated predefined variables. Refined Sect. 6. Added $\bar{x}_H$ , $\bar{x}_{RMS}$ , nine statistical sums and five curve fit models. Split <u>STAT</u> in <u>STAT</u> and <u>SUMS</u> ; renamed RMDR to RMD, $L_n$ to $L_m$ , $L_{na}$ to $L_{ma}$ , $\Pi$ to $\Pi_n$ , $\Sigma$ to $\Sigma_n$ , and some constants to avoid search ambiguities. Refined App. J, Sect. 3 and 4, $\rightarrow$ INT, <u>CLR</u> , and the functions of $\Delta$ and $\nabla$ . Put <u>SUMS</u> instead of RMD on the keyboard, moved <u>ADV</u> , <u>BITS</u> , <u>CATALOG</u> , <u>EQN</u> , <u>FILL</u> , <u>INTS</u> , <u>MATX</u> , <u>MODE</u> , <u>PROB</u> , RTN, SHOW, STAT, and <u>a.FN</u> . Rearranged <u>A...Ω</u> and Sect. 2 of the OM.
0.12	16.10.19	Rearranged the appendices of the <i>ReM</i> from App. D on. Expanded App. A of the OM and App. K. Deleted the standardized normal distribution $\Phi$ and rearranged <u>PROB</u> . Updated <u>CNST</u> following CODATA 2018. Renamed the angular conversions. Changed the composing and cutting functionality of <b>CC</b> . Refined exiting <i>short integer</i> input. Expanded App. D. Specified maximum size of <i>long integer</i> . Changed keyboard adding $\sqrt{x}$ , moving <u>CPX</u> , <u>FIN</u> , <u>RBR</u> , <u>R↑</u> , and <u>SHOW</u> , removing %. Renamed VANGLE to $V\sqrt{x}$ . Modified <u>CPX</u> , <u>MATX</u> , <u>TRI</u> , and <u>X.FN</u> . Rearranged Section 1 of the OM. Added some internal <i>data types</i> to App. B; reduced the range of <i>long integer</i> results and DP real inputs to $10^{-999}$ . Defined the domains of $e^{x-1}$ , IDIVR, LN(1+x), MOD, and RMD according to the HP-42S; modified PLOT and $\Sigma+$ . Refined the <i>Addressing Tables</i> . Added a <i>data type</i> matrix for IDIVR. Refined the <i>Special Results</i> in App. B.
0.13	30.11.19	Expanded the alpha keyboard and App. I. Modified <u>CPX</u> , <u>INTS</u> , <u>MODE</u> , <u>PROB</u> , <u>STK</u> , <u>TEST</u> , <u>a●</u> , SHOW, and STATUS. Refined the sorting order of <i>items</i> , ALL, CX $\rightarrow$ RE, MEM?, RE $\rightarrow$ CX, RBR, RM, SLVQ, and $\underline{U}\rightarrow$ . Started filling App. F and G. Refined App. 2. Added a <i>long integer</i> example, CPXR?, LZ?, $\Delta v_{Cs}$ , conversions of <i>hectares</i> , and a proposal for system status information.
0.14	7.3.20	Introduced <i>system flags</i> for status information. Split <u>I/O</u> . Added <u>CATALOG</u> 'SYS.FL, <u>PRINT</u> , <u>PROG</u> , <u>RANI#</u> , <u>VAR</u> , auxiliary constants, some predefined variables, and an index in App. I. Changed keyboard swapping <u>MODE</u> and <u>FLAGS</u> , $U\rightarrow$ and $\underline{U}\rightarrow$ , moving <u>CPX</u> , <u>FILL</u> , <u>RBR</u> , <u>R↑</u> , <u>USER</u> , <u>a.FN</u> , <u>aINTL</u> , $\sqrt{x}$ , and $\underline{x}$ , displaying <u>PRINT</u> , <u>RMD</u> , <u>STATUS</u> , $x^2$ , and ‘:’, and removing <u>c/d</u> , <u>dx</u> , $\rightarrow$ SP, and $\rightarrow$ DP. Renamed <u>DISP</u> to <u>DSP</u> and <u>SUMS</u> to $\Sigma$ , changed <u>G</u> to <u>H</u> . Refined the addressing tables and catalog access, <u>a b/c</u> , <u>ADV</u> , <u>BATT?</u> , <u>BITS</u> ,

Date	Release notes
	<p><u>CATALOG'CHARS</u> and '<u>MENUS</u>, <u>CLALL</u>, <u>CLFALL</u>, <u>CPX</u>, <u>EXP</u>, <u>GAP</u>, <u>INTS</u>, <u>I/O</u>, <u>MODE</u>, <u>NEIGHB</u>, <u>PARTS</u>, <u>PRIME?</u>, <u>P.FN</u>, <u>SHOW</u>, <u>STAT</u>, <u>STK</u>, <u>X.FN</u>, <u>aINTL</u>, and <u>a•</u>. Deleted all 16-digit (i.e. SP) data types as well as <u>A...Z</u> and the commands CLK12, CLK24, CPXi, CPXj, CPXRES, CPXR?, DBL?, DENANY, DENFAC, DENFIX, ENGOVR, FAST, IMPFRC, LZOFF, LZON, LZ?, MULTx, MULT-, POLAR, PROFRC, QUIET, RDX., RDX,, REALRE, RECT, SCIOVR, SLOW, SSIZE4, SSIZE8, →DP, and →SP. Corrected.</p>
0.15 27.5.20	<p>Added <u>BESTF?</u>, <u>RANGE</u>, <u>RANGE?</u>, <u>REGIST</u>, <u>SNAP</u>, and <u>s(a)</u> as well as errors 28 and 31 – 34. Changed DSZ and ISZ to comply with HP-16C. <b>Changed keyboard shifting N, O, P, and Q, swapping ? and Z, moving CNST, CPX, FLAGS, RBR, RTN, R↑, VIEW, and □, removing :, and adding MOD, ✓, and SNAP.</b> Renamed <u>DSP</u> to <u>DISP</u>, <u>CNST</u> to <u>CONST</u>, <u>CONST</u> to <u>CNST</u>, ASL.BLK to ASLIFT, SSIZE to SSIZE8, TDM to TDM24, and the left and right sided probabilities. Refined <u>ASSIGN</u>, <u>CATALOG</u>, <u>CNST</u>, <u>DISP</u>, <u>INFO</u>, <u>NEXTP</u>, <u>PRIME?</u>, <u>PROB</u>, <u>RBR</u>, <u>RESET</u>, <u>SHOW</u>, <u>SINC</u>, <u>STAT</u>, <u>VIEW</u>, <math>x=+0?</math>, <math>x=-0?</math>, <math>a\rightarrow x</math>, 4, pp. 54 – 57 and 205 – 207 (and consequences) as well as <i>Section 6</i> of the OM, pp. 108 – 117 and <i>App. E</i> of the ReM, and some looping and statistical explanations. Reduced the maximum number of <i>local registers</i> from 100 to 99. Corrected.</p>



# 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. Q-7), probability distributions (see p. Q-8), 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 **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,  $\leq 7$  characters long, and contain  $\geq 1$  letter.

**Flags:** There are 112 global *user flags*. and some 35 named *system flags*.

**Programs** consist of  $\geq 4$  program steps: LBL with a global label, at least one action step, RTN, and END. Each program may contain subroutines (up to 8 levels deep). See p. Q-5 for more.

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

## DATA TYPES

**Long integers** are the simplest type of data. Any number you enter without using , , , or is taken as a *long integer* of base 10.

**Real numbers:** Any number you enter using and/or 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 (SF RECTN and press **1.23 [CC] 4.56 [+/-] [ENTER]**) or its magnitude and phase like  $-7.89 \angle 120^\circ$  in polar mode (CF RECTN and press **7.89 [+/-] [CC] 120 [ENTER]**).

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

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

**Dates:** Any real number input 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 pp. Q-7 f.

**Short integers:** Any purely 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:** Enter *alpha input mode* (**AIM**) by pressing **@**. Data entered in *AIM* become an alphanumeric string when closed (unless they are function parameters). All Latin letters (also accented ones) are found in **f[A]**. Greek letters are accessed via **g** plus the corresponding Latin letter (see calculator backside). Turn to lower case by **[▼]** and back to upper by **[▲]** for all letters.

**f** plus one of the keys **[+]**, **[-]**, **[x]**, **[.]**, and **[0]** – **[9]** will 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

MODE	SYSTEM		RM	SETSIG	DENMAX	
	SF	DEG	RAD	GRAD	MULπ	CF

SF *n*, CF *n* Set (or clear) the flag specified.

DEG Selects *degrees* as angular display mode (ADM).

RAD Selects *radians* as ADM.

GRAD Selects *gradians*, a.k.a. *gon*, as ADM.

MULπ Selects *multiples of π* as ADM.

RM *n* Sets rounding mode.

SETSIG *n* Sets calculator precision (1 ... 34 significant digits).

DENMAX *n* Sets the maximum denominator for calculating with fractions.

SYSTEM Returns the calculator to the DMCP system for updating.

## DISPLAY FORMATS

DISP	GAP		RANGE	RANGE?		DSTACK	
	CHINA	EUROPE	INDIA	JAPAN	UK	USA	
	SDL	SDR			RDP	RSD	

FIX *n* Fixed number of *n* decimals.

SCI *n*, ENG *n* Scientific (or engineering) notation.

ALL *n* Displays all digits required as far as possible.

ROUND Rounds a *time*, real, or complex *x* to current display format.

ROUNDI Rounds to next integer.

RDP *n* Rounds *x* to *n* decimal places (1 ... 99, think of FIX)

RSD *n* Rounds *x* to *n* significant digits (1 ... 34, think of SCI).

SDL *n*, SDR *n* Shifts digits left (right) by *n* decimal positions.

CHINA, EUROPE, INDIA, JAPAN, UK, USA Set local display preferences.

GAP *n* Selects a digit group gap inserted after every *n* digits.

RANGE *n* Sets the maximum exponent to be displayed for real numbers

RANGE? Returns the range setting

DSTACK *n* Sets the maximum number of stack registers to be displayed (1 ... 4).

## EXECUTING FUNCTIONS AND PROGRAMS

Any function or program can be executed via **[XEQ]  $\alpha$  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] PROG** 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  **$\equiv \nabla$**  (or  **$\nabla$**  if no multi-view menu is displayed). Press  **$\equiv \Delta$**  (or  **$\Delta$** ) for browsing backwards.

**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]  $\alpha$  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 keying in an actual parameter, you can specify the variable or *register* containing the parameter. Just press the softkey **[ $\rightarrow$ ]**. E.g. to display the contents of the variable or *register* specified in **R12**, key in **[VIEW] [ $\rightarrow$ ] 12**. This works with *stack registers* as well.

## CLEARING AND DELETING

<b>CLR</b>	CLall					RESET	
	CLREGS	CLPall	CLFall		CLLCD	CLSTK	
	CLΣ	CLP	CF	CLMENU	CLCVAR	CLX	

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

**CATALOG** VARS ... allows for deleting the user variable selected.

**CATALOG** MENUS ... allows for deleting the user *menu* selected.

**CATALOG** PROGS ... allows for deleting the user 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      (use or unless a *multi-view menu* is displayed).  
 moves it to next step

deletes the *current program step* entirely.

**EXIT** exits *program entry mode*.

## 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 (cf. p. Q-4).

**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 ( $\leq 100$ ). 16 *local flags* come with them. Local data are valid in the calling routine only.

## Tests (Do if True, Skip if False)

When a binary test step is executed, the program step immediately following said step is executed if the test result is “true”; if the result is “false”, the step following the test step 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 ccccc.fffii with ccccc being the current counter value, fff the final counter value, and ii the increment (or decrement) size (default is 1); DSZ and ISZ count to 0 in steps of 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). The example 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 (see pp. Q-10f) 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** **α** **name** **ENTER↑** for a matrix in a variable. If the variable already exists, DIM re-dimensions it.

**To edit the matrix in X**, use **[MATX] EDIT** .

**To edit 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. Any time 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		NBin:	Geom:	Hyper:	Binom:	Poiss:	
	LgNrm:	Cauch:		Expon:	Logis:	Weibl:	
	Norml:	t:	C <sub>yx</sub>	P <sub>yx</sub>	F:	$\chi^2$ :	
Binom:	Binom <sub>p</sub>		Binom <sub>▲</sub>	Binom <sub>▲</sub>		Binom <sup>-1</sup>	

- C<sub>yx</sub> , P<sub>yx</sub>      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 the *Gamma function* value of  $x$ .

These 14 continuous (c) and discrete (d.) distributions (d.) are provided:

- Binom: d *Binomial d.*      ( $i = p_0$  = gross probability of a success,  
                                         $j = n$  = sample size)  
 Cauch: c *Cauchy-Lorentz* (a.k.a. *Breit-Wigner*) d.      ( $i$  = location,  $j$  = shape)  
 Expon: c *Exponential d.*      ( $i$  = rate)  
 F: c *Fisher's F d.*      ( $i$  = degrees of freedom 1 ( $dof_1$ ),  $j$  =  $dof_2$ )  
 Geom: d *Geometric d.*      ( $i = p_0$ )  
 Hyper: d *Hyperbolic d.*      ( $i = p_0$ ,  $j = n$ ,  $k$  = batch size)  
 LgNrm: c *Log-normal d.*      ( $i = \mu$ ,  $j = \sigma$ )  
 Logis: c *Logistic d.*      ( $i = \mu$ ,  $j$  = scale parameter)  
 NBin: d *Negative Binomial d.*      ( $i = p_0$ ,  $j = n$ )  
 Norml: c *(General) normal d.*      ( $i = \mu$ ,  $j = \sigma$ )  
 Poiss: d *Poisson d.*      ( $i = n p_0$  = Poisson parameter)  
 t: c *Student's t d.*      ( $i$  = dof)  
 Weibl: c *Weibull d.*      ( $i$  = shape,  $j$  = characteristic lifetime)  
 X<sup>2</sup>: c *Chi-square d.*      ( $i$  = dof)

Following naming convention holds for most distributions, e.g. for the *normal d.*:  
**Norml<sub>p</sub>** denotes the *probability density function*, **Norml<sub>▲</sub>** the *cumulated d. function*, **Norml<sub>▲</sub>** the *error probability*, and **Norml<sup>-1</sup>** the *quantile function*.

Store the required 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 functions* require a probability given in **X** ( $0 \leq x \leq 1$ ).

## STATISTICS

Statistical data are accumulated in 23 dedicated summation *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 individual data value: **x-value Σ+**.
- For each weighted data value: **weight-value ENTER↑ x-value Σ+**.
- For each x-y data pair or point: **y-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 complete matrix in **X** and then press **Σ+**.

To undo input errors or remove erroneous data,

- either press **UNDO** (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

	GaussF	CauchF	ParabF	HypF	RootF		
	LinF	ExpF	LogF	PowerF		BestF	
		$\bar{x}_{RMS}$				OrthoF	
		$\bar{x}_H$					
	L.R.	r	$s_{xy}$	cov	$\hat{x}$	$\hat{y}$	
STAT	CLΣ	$\bar{x}_G$	$\varepsilon$	$\varepsilon_p$	$\varepsilon_m$	PLOT	
	$\Sigma^-$	$\bar{x}_w$	$s_w$	$\sigma_w$	$s_{mw}$		
	$\Sigma^+$	$\bar{x}$	s	$\sigma$	$s_m$	SUM	

$\bar{x}$ , s,  $\sigma$ ,  $s_m$  Arithmetic mean value, sample standard deviation (SD), population SD, standard error (a.k.a. SD of the mean).

$\bar{x}_w$ ,  $s_w$ ,  $\sigma_w$ ,  $s_{mw}$  Same for weighted data.

$\bar{x}_G$ ,  $\varepsilon$ ,  $\varepsilon_p$ ,  $\varepsilon_m$  Geometric mean value, sample scattering factor (SF), population SF, SF of the mean.

$\bar{x}_H$ ,  $\bar{x}_{RMS}$  Harmonic and quadratic mean values.

SUM Recalls  $\Sigma y$  and  $\Sigma x$ .

PLOT See the ReM.

L.R. Computes the parameters  $a_0$  and  $a_1$  (and  $a_2$ , if applicable) of the fit model selected (see below).

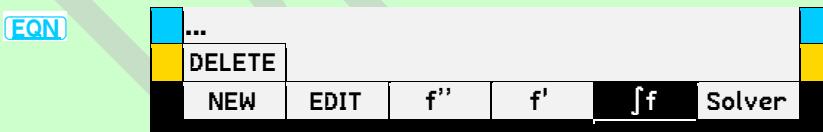
r	Returns the correlation coefficient.
S <sub>xy</sub> , cov	Return the sample or population covariance.
$\hat{x}, \hat{y}$	Return the forecast for x or y according to the fit model selected.
LinF	Linear fit model: $y = a_0 + a_1x$ .
ExpF	Exponential fit model: $\ln(y) = \ln(a_0) + a_1x$ or $y = a_0 e^{a_1 x}$ .
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 $y = a_0 x^{a_1}$ .
RootF	Root fit model: $y = a_0 a_1^{1/x}$ .
HypF	Hyperbolic fit model: $y = 1/(a_0 + a_1 x)$ .
ParabF	Parabolic fit model: $y = a_0 + a_1 x + a_2 x^2$ .
CauchF	Cauchy peak fit model: $y = 1/[a_0 (x + a_1)^2 + a_2]$ .
GaussF	Gauss peak fit model: $y = a_0 e^{\frac{(x-a_1)^2}{a_2}}$ .
BestF	Blindly selects the model returning the best correlation coefficient.
OrthoF	Works like LINF but assumes equal errors in x and y. Note ORTHOF is not part of the fit model pool BESTF investigates.

## ADVANCED OPERATIONS

[EQN] is for interactive editing, storing, recalling, solving, integrating, & deriving equations.

[ADV] is for programmed summing, multiplying, solving, integrating, & 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 **g**-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)	f''(x)	$\prod_n$	$\sum_n$	$\int f dx$
SOLVE	SLVQ					

**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.

**$\prod_n$  label** calculates the product of the terms given by the routine specified, using the loop control number given in **X** (cf. p. Q-6).

**$\sum_n$  label** calculates the sum of the terms given by the routine specified, using the loop control number given in **X** (cf. p. Q-6).

**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**.

**f<sub>fd</sub> 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 **f<sub>fd</sub>**. 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 (**↓LIM**), the upper limit (**↑LIM**), and the accuracy factor (**ACC**).
5. Press **ʃ** 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

	1COMPL	2COMPL	UNSIGN	SIGNMT		WSIZE	
	LJ					RJ	
	SL	RL	RLC	RRC	RR	SR	
<b>BITS</b>	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 # $n$  in  $x$  is set (or clear).

#B Returns the number of bits set in  $x$ .

SL, SR  $n$  Shift  $x$  left (or right) by  $n$  places.

RL, RR  $n$  Rotate  $x$  left (or right) by  $n$  places.

RLC, RRC  $n$  Rotate  $x$  left (or right) by  $n$  places through Carry.

LJ, RJ Adjust the bits set in  $x$  to the left (or right).

1COMPL, 2COMPL 1's (2's) complement mode.

UNSIGN	Unsigned mode.					
SIGNMT	Sign-and-mantissa mode.					
WSIZE	Sets the word size (1 ... 64 bits).					

INTS	1COMPL	2COMPL	UNSIGN	SIGNMT	WSIZE	
	DBL /	DBLR	DBL ×	^MOD	CEIL	GCD
	IDIV	RMD	MOD	×MOD	FLOOR	LCM
	A	B	C	D	E	F

A ... F	Digits for short integers of bases >10.						
IDIV	R Z	Integer divide – works for real numbers (R) and long integers (Z) as well.					
RMD, MOD	R Z	Remainder and modulo.					
×MOD	R Z	Returns $(z \cdot y) \bmod x$ .					
^MOD	R Z	Returns $(z^y) \bmod x$ .					
FLOOR	R	Returns the greatest integer $\leq x$ .					
CEIL	R	Returns the smallest integer $\geq x$ .					
LCM	Z	Returns the least common multiple of $x$ and $y$ .					
GCD	Z	Returns the greatest common divisor of $x$ and $y$ .					
DBL /, DBLR, DBLx	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.

a.FN	FBR				αLENG?	αPOS?	
	x→α	αRL	αRR	αSL	αSR	α→x	

x→α s	Converts a code $x$ to the corresponding character and appends it to the string in $s$ .
αRL, αRR s	Rotates the string in $s$ by $x$ characters to the left (or right).
αSL, αSR s	Deletes the first (or last) $x$ characters of the string in $s$ .
α→x s	Pushes the code of the first character in $s$ on the stack.
αLENG? s	Pushes the length of the string in $s$ on the stack.
αPOS? s	Returns the position where substring $x$ begins in the string in $s$ .
FBR	Displays all characters defined in both fonts.



## BACKGROUND CONSIDERATIONS AND FACTS

This section is for recording and explaining some of the boundary conditions considered and settings chosen for the *WP 43S* in the course of this project. It is not necessary for operating the *WP 43S* but may foster understanding. A bit of the product philosophy may be found here, too.

### Accessing Items

The hardware offers 43 keys for some 750 *items*. Subtract six for the *softkeys*. Space for primary functions is quickly occupied. These functions are mostly set. For the remaining set, secondary functions compete with *menus*.

Obvious primary functions are the digits **0** ... **9**, **.**, **ENTER↑**, **x<sup>2</sup>y**, **+L**, **E**, **◀**, **+**, **-**, **x**, **/**, **STO**, **RCL**, **XEQ**, **R/S**, **▲** and **▼**, **EXIT**, **f** and **g**, taking 29 key tops. So eight locations are left. Once you want to deal with complex numbers seriously, you need a primary **CC**.

Other functions may be debated: **R↓**, **1/x**, **y<sup>x</sup>**, **x<sup>2</sup>**, **fx**, **e<sup>x</sup>**, **ln**, **10<sup>x</sup>**, **lg**, **sin**, **cos**, and **tan** are the most popular – twelve candidates for seven key tops left. Either **x<sup>2</sup>** or **fx** shall be primary (we chose **x<sup>2</sup>** since a shifted **x<sup>2</sup>** is of little use); and we can ditch **10<sup>x</sup>** and **lg** when we have **e<sup>x</sup>** and **ln** primary. So **R↓**, **1/x**, **y<sup>x</sup>**, **sin**, **cos**, and **tan** compete for the remaining four key tops. We chose the first three and put the latter three (and their inverses) under one primary *menu* key: **TRI**; thus, you can access each of **sin**, **cos**, **tan**, **arcsin**, **arccos**, and **arctan** with two keystrokes maximum.

The losing candidates **fx**, **10<sup>x</sup>**, and **lg** shall become secondary functions. But is it better having them as shifted keyboard functions or unshifted *softkeys*? No definite answer can be given here since it depends on the time the respective *menu* will stay on screen. For these particular functions, we made all three **g**-shifted and put **fx** also in the unshifted row of **EXP** since it looks like the most popular of these three.

The WP 43S features 33 *menus* on its keyboard. Of these, CATALOG, CONST, U→, and the three alpha character *menus* shall be separated since they follow special rules. Each of the remaining 27 *menus* offers six unshifted locations for its most popular *items*. Selecting these can be easy (like in ADV, LOOP, STK, or TRI) or difficult (like in P.FN or X.FN).

Unshifted *softkeys* are (cf. pp. 130ff):

<u>ADV</u>	SOLVE	SLVQ	$f'(x)$	$\Pi_n$	$\Sigma_n$	$\int f dx$
<u>BITS</u>	AND	OR	XOR	NOT	MASKL	MASKR
<u>CLK</u>	DATE	→DATE	DATE→	WDAY	TIME	x→DATE
<u>CLR</u>	CLΣ	CLP	CF	CLMENU	CLCVAR	CLX
<u>CPX</u>	dot	cross	UNITV	Re	conj	Re>Im
<u>DISP</u>	FIX	SCI	ENG	ALL	ROUNDI	ROUND
<u>EQN</u>	NEW	EDIT	$f''$	$f'$	$\int f$	Solver
<u>EXP</u>	$x^3$	$\sqrt[3]{y}$	$\log_{10} y$	lb x	$2^x$	$\sqrt{x}$
<u>FIN</u>	%	%MRR	%T	%Σ	%+MG	TVM
<u>FLAGS</u>	SF	FS?	FF	STATUS	FC?	CF
<u>INFO</u>	SSIZE?	MEM?	RM?	S MODE?	WSIZE?	K TYP?
<u>INTS</u>	A	B	C	D	E	F
<u>I/O</u>	BEEP	LOAD	LOADP	LOADR	LOADSS	LOADΣ
<u>LOOP</u>	DSE	DSZ	DSL	ISE	ISZ	ISG
<u>MATX</u>	NEW	$[M]^{-1}$	M	$[M]^T$	SIM EQ	EDIT
<u>MODE</u>	SF	DEG	RAD	GRAD	MULπ	CF
<u>PARTS</u>	IP	FP	MANT	EXPT	sign	DECOMP
<u>PRINT</u>	☒x	☒r	☒Σ	☒ADV	☒LCD	☒PROG
<u>PROB</u>	Norml:	t:	$C_{yx}$	$P_{yx}$	F:	$\chi^2$ :
<u>P.FN</u>	INPUT	END	ERR	TICKS	PAUSE	P.FN2
<u>STAT</u>	Σ+	Ȑx	s	g	s <sub>m</sub>	SUM

<u>STK</u>	x $\gtrless$	y $\gtrless$	z $\gtrless$	t $\gtrless$	$\gtrless$	DROPy
<u>TEST</u>	x< ?	x≤ ?	x= ?	x≠ ?	x≥ ?	x> ?
<u>TRI</u>	sin	arcsin	cos	arccos	tan	arctan
<u>U→</u>	E:	P:	year→s	F&p:	m:	x:
<u>X.FN</u>	AGM	B <sub>n</sub>	B <sub>n</sub> *	erf	erfc	Orthog
<u>α.FN</u>	x $\rightarrow$ α	αRL	αRR	αSL	αSR	α $\rightarrow$ x
$\Sigma$	n	$\Sigma x$	$\Sigma x^2$	$\Sigma xy$	$\Sigma y^2$	$\Sigma y$
<u>π→</u>	→DEG	→RAD	→GRAD		→D.MS	→MULπ

Repeating any unshifted *softkeys* on the keyboard is of limited value. It makes sense just for occasional use of very popular functions –  $\sqrt{x}$  needs two keystrokes while  $\sqrt[3]{x}$  needs three keystrokes unless EXP is open already.  $|x|$  and  $\angle$  are also featured as shifted *softkeys*: accessing  $|x|$  or  $\angle$  needs two keystrokes while  $|x|$  and  $\angle$  require four maximum (and two if the *menu* is open). In consequence,  $|x|$  and  $\angle$  are actually not needed in any *menu* but are left in CPX and PARTS since space is available there.

## Alpha Register

For long I thought we could do without a dedicated *alpha register* since each and every *register* is capable holding an alpha string. 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 it (cf. the OM, Section 3).

## 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 real numbers’. 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  $\pi$* , and *gon* or *grads*). They were represented internally in a fixed format of 1296 units per turn – similar to *short integers* where a fixed bit pattern may be displayed differently depending on *integer sign modes* and bases selected. *Radians*, however, did not fit into this concept due to the necessity for high precision storage of  $\pi$  for modulo calculations and reduction of rounding errors.

Generally, trigonometric functions shall actually operate on *angles* within  $\pm 180^\circ$  only; thus, angular input beyond this range shall be reduced modulo  $360^\circ$ , then minus  $180^\circ$  (or equivalents in the other *angular display modes* available) before executing the function. Again, the crucial mode are *radians*. *WP 34S* had demonstrated that 451 digits for  $2\pi$  suffice to warrant 16 digits accuracy of respective function results for the number range of *single precision reals*.<sup>93</sup> *WP 43S* uses 1065 digits for  $2\pi$  to warrant 34 digits accuracy of respective function results within  $\pm 10^{999}$ .

## Backward Compatibility

Compatibility to *WP 34S* and *HP-42S* was planned to be kept for many years in a way that programs written for both calculators could have run on the *WP 43S* as well (except matrix operations and some flag allocations). It became difficult with the full implementation of the *data type* concept and had to be eventually abandoned officially when introducing named *system flags* with v0.14.

Nevertheless, *names* of *items* were kept as close as possible to the *names* users are used to. Extra entries are provided catching traditional *names*.

---

<sup>93</sup> See <https://forum.swissmicros.com/viewtopic.php?f=2&t=350#p4349>.

## Calculation Internals

General powers are calculated in  $\mathbb{R}$  and  $\mathbb{C}$  as  $y^x = e^{x \ln(y)}$  and general roots as  $\sqrt[x]{y} = e^{\frac{\ln(y)}{x}}$  for all values of  $x$  except the integers 2 and 3. Some special results in [App. B](#) can be deduced from this.

## Character Sets

The browser FBR displays the characters of both fonts provided as designed and implemented for the *WP 43S*, sorted according to their hexadecimal codes (most of them following Unicode).

The so-called '**numeric**' font uses a matrix of up to  $16 \times 32$  px (variable width, fixed height). Therein, the punctuation space ( $2008_{16}$ , 8 px wide) is employed for separating groups of digits in longer numbers – following *ISO 80000-1* for an unambiguous numeric display. This font is generally used for numeric output of the *WP 43S*. It is also employed for echoing numeric input unless too long. It can be used for echoing command input as well – screen space suffices.

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 fraction numerators. The digits below are for denominators. Numeric indices are for indicating bases of short integers. Non-numeric indices are mainly provided for CONST.

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 pp. B-7ff).

All characters of the **standard** (a.k.a. small) **font** of alphanumeric characters as designed and implemented live in a matrix of up to  $14 \times 20$  px (variable width, fixed height again). Herein, characters usually start at column one and feature two empty columns at their right side. There are a few exceptions: see e.g. the multiplication dot at  $00B7_{16}$  and the root symbols in row  $2210_{16}$ .

Characters with codes < 0020<sub>16</sub> are for control purposes; some of them (4, 10<sub>10</sub>, 27<sub>10</sub>) may be useful for printer control (e.g. of an 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.

Eight blank characters are provided (listed here with their hex addresses and their widths). Using them, any spacing is feasible.

This small character set allows for correctly spelling the languages of more than  $3.5 \times 10^9$  people using either Greek or Latin alphabets:

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, Slovenčina, Slovensky, Srpski, Suomi, Svenska, Tagalog, Tatarça, Türkçe, Türkmençe, Vlaams, Wallon, and Zhōngwén (hànyǔ pīnyīn).

This makes the WP 43S the most versatile multilingual calculator available worldwide.<sup>94</sup> If you know of further living languages covered (with  $\geq 1$  million speakers) beyond the ones listed here, please tell us.

Turn to the OM for examples where these characters are used. See below two sample strings in either font, printed approximately to a

---

<sup>94</sup> Some characters displayed by FBR are not found in any other *menu* of your WP 43S. They are not required for any *item* provided so far and may be for future use.

common realistic scale:

$$\begin{array}{c} -1.602\ 22 \times 10^{-19}\ C \\ -1.602\ 22 \times 10^{-19}\ As \end{array} \quad \begin{array}{c} -1,602\ 22 \cdot 10^{-19}\ C \\ -1,602\ 22 \cdot 10^{-19}\ As \end{array}$$

## Complex Notation and Storage

Like with angles or short integers, there are different ways a complex number can be written: either in Cartesian or polar notation, the latter with all kinds of angular units. As long as you stay away from infinities, any notation will do.

For reasons of mathematical tradition, rectangular notation is found most frequently. We used it for storing complex numbers in the *WP 34S*. Thus, it is used for the *WP 43S* as well, but care must be taken at complex infinities as explained in next chapter.

## Complex Numbers close to Infinity

Since infinities are counted as numeric data being part of the real number range (see p. 174), also complex infinities are part of the complex number plane the *WP 43S* operates on. Coming from rectangular notation, there are eight ‘complex infinities’ possible only, listed here aside to their equivalents in polar notation:

$\text{Re}(z)$	$\text{Im}(z)$	$r(z)$	$\varphi(z)$
$-\infty$	$-\infty$	$\infty$	$-135^\circ$
0	$-\infty$	$\infty$	$-90^\circ$
$\infty$	$-\infty$	$\infty$	$-45^\circ$
$\infty$	0	$\infty$	$0^\circ$
$\infty$	$\infty$	$\infty$	$45^\circ$
0	$\infty$	$\infty$	$90^\circ$

<b>Re(z)</b>	<b>Im(z)</b>	<b>r(z)</b>	<b>φ(z)</b>	
−∞	∞	∞	135°	$3\pi/4$
−∞	0	∞	180°	π

Note the phase is counted counterclockwise, starting with  $\varphi = 0$  at the positive real axis.

Calculating with infinities, any finite numbers may be neglected in comparison; so for  $|x| \neq \infty$  any inputs like e.g.  $x + i \times \infty$  may be replaced by  $0 + i \times \infty$ , easing calculations significantly. Actually, you have to deal with the eight cases listed above only as long as you build your complex calculations on Cartesian notation (see footnote 72 for additional information). With polar notation, on the other hand, an infinite number of complex infinities would have to be treated.

Note, however, that polar notation is advantageous for complex powers and roots: the  $n^{\text{th}}$  root of a complex number  $(r; \varphi)$  in polar notation will return  $(\sqrt[n]{r}; \varphi/n)$ . Hence, e.g.  $\sqrt[3]{(\infty; 180^\circ)} = (\infty; 60^\circ)$ , corresponding to the Cartesian point  $\infty \times (1 + i\sqrt{3})$  which the calculator will display as  $\infty + i \times \infty$ , converted following the calculation rules but mathematically wrong; and  $\sqrt[6]{(\infty; 180^\circ)} = (\infty; 30^\circ)$ , corresponding to the Cartesian point  $\infty \times (\sqrt{3} + i)$ , would return  $\infty + i \times \infty$  as well.

Integer powers of  $(r; \varphi)$ , on the other hand, will return  $(r^n; \varphi \times n)$ . E.g.  $\infty + i \times \infty = (\infty; 45^\circ)$  squared shall return  $(\infty; 90^\circ) = 0 + i \times \infty$ . Naïve pedestrian's approach:  $(\infty + i \times \infty)(\infty + i \times \infty) = \infty - \infty + 2i\infty = 0 + i \times \infty$ . Note the two  $\infty$  are exactly identical here; but  $\infty - \infty$  is generally defined as NaN for good reasons, so the correctly calculated result will deviate from the truth here, too.

Thus, the odds are high that roots and powers of complex numbers near the edge of complex plane return incorrect data, in particular if specified in polar notation.

## Display Limits

Due to the character sizes and their design (cf. pp. B-5f), the screen could take inputs of up to 23 digits, a sign, and an 8-px 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 hardly be readable. With 3-digit separators (*startup default*), 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 again. 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.<sup>95</sup>

With SHOW, any real number can be displayed with 34-digits precision in a single row:

2020-01-06 17:28 CL<sub>E</sub> 4<sup>r</sup> /max 64:2 A S  
-1.428 571 428 571 428 571 428 571 428 571 429 $\times 10^{-235}$   
-1.428 571 428 571 429 $\times 10^{-235}$

Some *temporary information* may limit output precision, though without limiting its use for real-world applications. 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\*  $a_1: -5.234 567 8 \times 10^{-92}$   
 $y = a_0 + a_1 \ln(x)$   $a_0: -1.234 567 890 12$

<sup>95</sup> One blank pixel column had to be added at right since exponential digits are right adjusted (since used for numerators as well) and the screen is framed in black.

**Complex numbers** in Cartesian notation require  $1 + 15 + 12 + 15 + 1 = 44$  px for  $+j\times$  in addition to the space for two reals. Only the real part may need extra space for a 15-px sign. This allows for 8 decimals per part in worst case

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

since  $44 + 15 + 2 \times (16 + 8 + 48 + 8 + 48 + 8 + 32) = 397$  px in total. It 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 ( $15 + 2 \times (4 \times 16 + 8 + 15 + 16 + 4 \times 13) + 44 + 1 = 370$  px, but another digit would need  $2 \times 16$  px at least):

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

Using 8 px wide multiplication dots instead, only  $1 + 15 + 12 + 8 + 1 = 37$  px are necessary for  $+i\cdot$ . Thus, we can show one decimal more since  $15 + 2 \times (5 \times 16 + 3 \times 8 + 16 + 4 \times 13) + 37 + 1 = 397$  px:

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

Alternatively, 13 px wide narrow digits allow for 4 decimals even with multiplication crosses, while 5 decimals are viable with multiplication dots:

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

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

With SHOW, any complex number can be displayed with 34 digits precision in two rows:

2020-01-06 17:15 CLE4R /max 64:2 A ↑ S  
 $-8.403\ 361\ 344\ 537\ 815\ 126\ 050\ 420\ 168\ 067\ 229\times 10^{-126}$   
 $-i\times 5.882\ 352\ 941\ 176\ 470\ 588\ 235\ 294\ 117\ 647\ 059\times 10^{-158}$

$$\begin{aligned} & -1.\times 10^{-33} \\ & -8.403\times 10^{-126}-i\times 5.882\times 10^{-158} \end{aligned}$$

Complex numbers in **polar notation** need  $4 + 16 + 4 = 24$  px for  $\angle$  plus 16 px for the angular unit in addition to the space for two signed

reals. Both magnitude and angle may require a 15 px sign. 7 decimals in FIX occupy  $40 + 2 \times (15 + 8 \times 16 + 3 \times 8) = 374$  px, so we can display them this way:

-4.234 567 8 ↵ -0.234 567 8π .

With SCI or ENG, the minimum number of decimals depends on the angular display mode since output is confined to the interval  $-180^\circ$  to  $+180^\circ$  or its equivalents, e.g.  $-\pi$  to  $+\pi$  in *radians* or  $-200^\circ$  to  $+200^\circ$  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 $\times 10^{-925}$  ↵ -120.234 5° .

For *radians* or *multiples of π*, however, 5 decimals are displayable always at least:

-4.234 56 $\times 10^{-925}$  ↵ -0.234 56π .

Digits in **fractions** are 13 px wide like in exponents. 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 the optional sign, an 11-digit number, and the 16 px gap between integer and fraction ( $15 + 12 \times 16 + 3 \times 8 = 231$  px) in a proper fraction:

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

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

-123 456 789 012 345 678 901 ,

taking  $(1 + 15 + 21 \times 16 + 6 \times 8 = 400$  px). Larger *long integers* employ 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* may be 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 the usual large digits in binary representation:

0 1100 0010 1101 0110 0000,

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

-22 211 200 201 120 001 212<sub>3</sub> .

In base 4 (with narrow blanks every two digits), 19 digits representing 38 bits are displayable:

3 21 23 30 22 11 21 20 32 12<sub>4</sub> .

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

Using the narrow digits provided, up to 25 bits are displayable in binary representation:

0 1110 1100 0010 1101 0110 0000<sub>2</sub> .

Then 24 digits and a sign can be shown 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 0000<sub>2</sub>.

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 base 2, two special 5 px wide characters were created:

`1111 1100 0011 1101 0110 1100 0010 1101 0110 1100 0010 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 <i>bits</i>	25 <i>bits</i>	44 <i>bits</i>	64 <i>bits</i>
3	31 <i>bits</i>	38 <i>bits</i>	64 <i>bits</i>	64 <i>bits</i>
4	38 <i>bits</i>	44 <i>bits</i>		
5	46 <i>bits</i>	55 <i>bits</i>		
6	51 <i>bits</i>	62 <i>bits</i>		
7	56 <i>bits</i>			
8	57 <i>bits</i>			
> 8	64 <i>bits</i>			

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

`[-6,609·10-19 -6,609·10-19 -1,609·10-19 -1,609·10-19]`

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

`[-6.609 2E-19 -6.609 2E-19 -1.609 2E-19 -1.609 2E-19]`

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

`[... -6,609 2·10-19 -6,609 2·10-19 -1,609 2·10-19 ...]`

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. Thus, 5 matrix rows ( $5 \times 20 + 4 = 104$  px) can be put in the space taken by 3 standard numeric rows ( $3 \times 32 + 2 \times 5 = 106$  px). So, a  $5 \times 4$  real matrix can be displayed entirely always, using SCI 3 in worst case.

In consequence, any chosen  $3 \times 3$  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  $3 \times 2$  section can be displayed in FIX 6 format maximum with surrounding ellipses like

[... -6.609 226+i·6.609 226 -1.609 226+i·1.609 226 ...]

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

[... -6.60E<sup>-199</sup>+i·6.60E<sup>-199</sup> -1.60E<sup>-199</sup>+i·1.60E<sup>-199</sup> ...] .

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

[... -6,609+i·6,609 -6,609+i·1,609 -1,609+i·1,609 ...] .

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<sub>4</sub>π /3 546f 64:u<sup>o</sup> A ☰☰♦♣♥

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 17<sup>th</sup> will close input) for a 7-character command indirectly addressing a 7-character variable entered in A/M. This can be done in either font.

Command and variable *names* in menus are discussed in the last paragraph of next chapter. Although seven characters are allowed for such *names*, six may well fill the screen space available there already. Thus, it is recommended to keep such *names* as short as possible, though meaningful.

## Display Segmentation

The *LCD* of the *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*, or
  - c) up to 7 rows of numeric output of *SHOW*, and
- 69 px maximum for the menu section.

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

The reasons for these figures are given here:

- Each regular alphanumeric row (e.g. labels or status or program steps) requires 20 px vertically (cf. pp. B-5f). There shall be at least one 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 last for its upper frame line). Thus, three *menu* rows require 69 px. In U, extra-large labels may appear: 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. p. B-5) plus 4 px separating it from the next such row. Thus, for 4 rows we need  $4 \times 32 + 3 \times 4 = 140$  px. We put the remaining 7 px below this output block.
- If there is a short alpha string in any stack *register*, its base line should be positioned where the base line of the respective numeric output would have been.
- With an alpha string in **X** needing two rows,  $1 + 20 + 1 + 20 + 1 = 43$  px are required vertically matching the  $7 + 32 + 4 = 43$  px available for the lowest numeric row. Longer alphanumeric strings will need SHOW to be shown entirely.
- In *PEM*, on the other hand,  $147 = 7 \times (20 + 1)$  px correspond to a block of 7 alphanumeric program rows.
- With SHOW, also pure numeric output may require more than one row – the small font will be used there as well. Cf. pp. 76 and B-12.

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 2 px 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.

## **Echo and Fallback**

Almost all key presses are echoed and fall back to NOP. Softkeys shall not be echoed since WYS/WYG applies there always. Some functionalities, wherever they may be assigned to, are not echoed (and hence cannot fall back) since

- 1) they are harmless (EXIT, UP, DOWN) or

- 2) reverting or exiting them is a no-brainer (UP, DOWN, P/R, all *menu* calls) requiring no more than one keystroke.

With the presence of UNDO (see p. B-29), the necessity of fallback to NOP becomes debatable overall; there is some benefit remaining in *user mode* as long as an overlay matching the actual assignments is not available. And I admit UNDO is shifted in *startup default* configuration.

## Equations

Equations are entered in **EQN** as written (i.e. following algebraic notation and rules). While editing them, punctuation spaces are automatically inserted after each constant or variable (you know a variable *name* ends when the next operator is entered) as well as after = and each operator like +, -, ×, ÷, !, except ^; a standard space is inserted after :. There is no implicit multiplication.

Other functions like absolute values, roots, or trigs shall be written using the parentheses softkey, e.g. pressing **JK**( ), then stepping back into the parentheses for specifying the argument. The same applies to dyadic (like **C<sub>xy</sub>**) and triadic operations in analogy – their arguments shall be separated by blank spaces inserted via **R/S**.

Closing the *Equation Editor*, numeric exponents are automatically converted from e.g. **xy**<sup>23</sup> to **xy<sup>23</sup>**. For easier handling, this will be reverted when editing such an equation again.

## Layouting

After drawing a lot of fictional calculators just for fun, we gained our real-world layout experience with the *WP 34S*. Based on it and seeing a forthcoming better display (than the very limited one of the *HP-20b/30b*) allowing for softkeys at the horizon, I conducted a poll on the forum of the *Museum of HP Calculators* about general preferences for the placement of the four basic arithmetic operators on a then quite hypothetical portrait *RPN* pocket calculator in November 2012 (see <https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234505>). Then I published the concept and first layout for the *WP 43S* on said forum in return (<https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234685>).



The picture shows my last layout before said post, wishfully assuming the good old slanted keys of the Seventies.

Everything thereafter was and is just patient refinement and careful tuning of the basic idea. It even survived an hardware switch – we were waiting for *Eric Smith's* and *Richard Ottosen's* so-called *Reptiles* (see the *HHC* meetings until 2015) still when *Michael* mailed me the concept of his and *David's* *DM42* in March 2016. Actually, *DM42* development overtook the *WP 43S* – it was launched late in spring 2017 while *Pauli* and me were looking for willing software engineers and

actual support beyond friendly words still in vain. Despite its popularity in the community, qualified manpower for coding *WP 43S* was the bottleneck in our project since 2012.

## Menus

The community does not like deep *menus*: it prefers the *HP-32SII* to the *HP-32S*. On the other hand, it wants a large function set. So *menus* become inevitable but shall be designed carefully.

*Menu* size corresponds to keystroke efficiency; optimum is a *menu* encompassing three *views* containing up to 54 functions in total: the top *view*, one *view* going up via  $\blacktriangleleft$ , and one going down via  $\blacktriangleright$ . Larger *menus* lack efficiency, smaller *menus* lack functionality. Besides function

visibility, an operation presented in the unshifted row of the top *menu* view is more efficient than a shifted function presented on the keyboard – if used more than just once (cf. p. B-1).

Generally, I separated status setting from ‘acting’ operations in different *menu* views or rows at least.

## Number Range

A number range up to  $10^{99}$  is sufficient for almost all real-world problems – else common scientific calculators would feature a larger numeric range generally (cf. also pp. 146f). So we can conclude that the real number range supported (cf. p. 170) suffices by far for solving what has to be solved. Saving display space gave reason for RANGE.

For sake of consistency, maximum numbers of different *data types* should match. I.e. the maximum absolute value allowed for a *long integer* should be approximately equal to the respective values for a *real* and a *complex number*.

Note the number range determines the precision required for calculating accurately (see *Precision and Accuracy* on p. B-21).

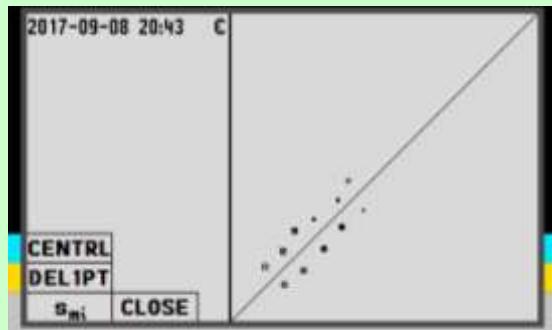
## Plotting?

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 third 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 earlier *RPN* calculators.

Plotting 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 ( $240 \times 240$  px for data,  $242 \times 240$  px incl. the vertical axis). Hence *softkeys* can be positioned on one side of the diagram (max. 3 x

2 labels) still. 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 was modified for clearing just the screen section required for the diagram. Four characters are provided in the small font for ‘drawing’ the vertical axis and the 45° line:



Though both axis and diagonal can also be created using AGRAPH and PIXEL.

Data points can then be plotted using POINT (containing 3x3 px) – positioning them properly in the diagram, however, will require some background calculations best performed by a program. 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, cf. GRAMOD on p. 109):

<b>Flag 34</b>	<b>Flag 35</b>	<b>How the AGRAPH Image is Displayed</b>
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 turned obsolete;)
- $s_{mi}$  for calculating the minimum experimental 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 in this almost quadratic drawing area then, using the commands AGRAPH, CENTRL, CLLCD, CLOSE, PIXEL, PLOT, POINT, and  $s_{mi}$ .

## Precision and Accuracy

As mentioned above more than once, there are inevitable errors in each numeric calculation step, frequently caused by rounding to the internal finite precision the calculator features. Already a simple fraction like  $1/3$  stored as a real number deviates from the truth by more than  $3 \times 10^{-35}$ . During calculations, such errors accumulate (cf. footnote 66).

In real-world problems, usually the least accurate of all input (real) parameters determines the accuracy of the result. In the standard test mentioned in said footnote starting with  $9^\circ$ , you can nevertheless get 28-digits precision in the result since the input of  $9^\circ$  is exact (but note one digit precision is lost with each trigonometric function calculated here).

Internally, for instance, the *WP 34S* computes with 39 digits and rounds the results to 34 or 16 digits, respectively (seems *Free42* works alike since the standard test results match). Following these, the *WP 43S* works with 39 digits internally and rounds the results to 34 digits as well. *SLVQ* calculates using 72 digits internally. The statistical summation registers are 75 digits wide (used also for the initial steps of variance calculation). Cf. also *Angles* on pp. B-4f.

Luckily, real-world problems are usually far less precisely defined than the internal precision of the *WP 43S*. Compare also the set of physical and astronomical constants provided (cf. pp. 138ff).

## Prefixes

Prefixes and passed without any discussion for more than six years until 2019-06. Alternatively, prefixes and could have been chosen but their typography leaves less freedom for label placing.

## Sorting in Detail

There is no international standard for sorting characters; we had to invent our own order. Sorting of *items*, variable *names*, alphanumeric strings, *system flags*, etc. on the *WP 43S* works as listed below, top down and left to right.

Note that sorting is a two-step procedure: step 1 sorts the alphanumeric strings under consideration just according to column 1 of this table, comparing them; if two strings are rated equal in this aspect, step 2 takes the columns following into account.<sup>96</sup> The 4-digit number trailing each character in the table is its hexadecimal *Unicode*.<sup>97</sup>

	0020	2003	2004	...	2008	200a		2423
<b>0</b>	0030	220e	00b0	<b>0</b> 2070	<b>0</b> 2080			
<b>1</b>	0031	<b>1</b> 2027	00bc	00bd	<b>1</b> 2071	<b>1</b> 2081	<b>1</b>	2460
<b>2</b>	0032	00b2	2082	<b>2</b> 2461				
<b>3</b>	0033	00b3	2083	<b>3</b> 2462	221b			

<sup>96</sup> Applying this algorithm, a section of CATALOG'FCNS looks like e.g. **s**, **SAVE**, **SB**, **SCI**, **SCI0VR**, **scw→kg**, ..., **SLVQ**, **s<sub>m</sub>**, **SMODE?**, **s<sub>mw</sub>**, **SOLVE**, ...

Sorting is illustrated for the small font here. It holds also for the large font as far as characters are applicable.

<sup>97</sup> Characters printed on grey background are inaccessible for users; those printed on darker grey are not used at all so far.

4	0034	4	2074	4	2084	4	2463						
5	0035	5	2075	5	2085	5	2464						
6	0036	6	2076	6	2086	6	2465						
7	0037	7	2077	7	2087	7	2466						
8	0038	8	2078	8	2088	8	2467						
9	0039	9	2079	9	2089	9	2468						
10	2491	10	2469										
11	246a												
12	246b												
13	246c												
14	246d												
15	246e												
16	246f												
A	0041	a	0061	a	00aa	A	24b6	a	2090	a	249c		
		À	00c0	à	00e0	Á	00c1	á	00e1	Ã	00c2	â	00e2
		Ã	00c3	ã	00e3	Ä	00c4	ä	00e4	Å	00c5	å	00e5
		Æ	00c6	æ	00e6	Ā	0100	ā	0101	Ā	0102	ă	0103
										À	0104	à	0105
B	0042	b	0062	B	24b7	b	249d						
C	0043	c	0063	c	24b8	c	249e	ç	00c7	ç	00e7		
		Ć	0106	ć	0107	Č	010c	č	010d	Ć	2102	ć	2201
D	0044	d	0064	d	24b9	d	249f	đ	00d0	đ	00f0		
						Đ	010e	đ	010f	Đ	0110	đ	0111
E	0045	e	0065	E	24ba	e	2091	e	24a0	È	00c8	è	00e8
		É	00c9	é	00e9	Ê	00ca	ê	00ea	Ë	00cb	ë	00eb
		Ē	0112	ē	0113	Ě	0114	ě	0115	Ē	0116	è	0117
		Ę	0118	ę	0119	Ě	011a	ě	011b	Ę	2073		
F	0046	f	0066	f	24a1	F	24bb						

<b>G</b> 0047	<b>g</b> 0067	<b>g</b> 24a2	<b>g</b> 24bc	<b>g</b> 011e	<b>g</b> 011f	
<b>H</b> 0048	<b>h</b> 0068	<b>h</b> 210e	<b>h</b> 24a3	<b>h</b> 24bd	<b>h</b> 2095	
					<b>h</b> 0127	<b>h</b> 210f
<b>I</b> 0049	<b>i</b> 0069	<b>i</b> 24be	<b>i</b> 24a4	<b>i</b> 00cc	<b>i</b> 00ec	
	<b>í</b> 00cd	<b>í</b> 00ed	<b>í</b> 00ce	<b>í</b> 00ee	<b>í</b> 00cf	<b>í</b> 00ef
	<b>í</b> 012a	<b>í</b> 012b	<b>í</b> 012c	<b>í</b> 012d	<b>í</b> 012e	<b>í</b> 012f
					<b>i</b> 0130	<b>i</b> 0131
<b>J</b> 004a	<b>j</b> 006a	<b>j</b> 24bf	<b>j</b> 24a5			
<b>K</b> 004b	<b>k</b> 006b	<b>k</b> 24c0	<b>k</b> 24a6	<b>k</b> 2096		
<b>L</b> 004c	<b>l</b> 006c	<b>l</b> 24c1	<b>l</b> 24a7	<b>l</b> 2097	<b>l</b> 0139	<b>l</b> 013a
			<b>l</b> 013d	<b>l</b> 013e	<b>ł</b> 0141	<b>ł</b> 0142
<b>M</b> 004d	<b>m</b> 006d	<b>m</b> 24c2	<b>m</b> 24a8	<b>m</b> 2098		
<b>N</b> 004e	<b>n</b> 006e	<b>n</b> 24c3	<b>n</b> 24a9	<b>n</b> 2099	<b>ñ</b> 00d1	<b>ñ</b> 00f1
	<b>ń</b> 0143	<b>ń</b> 0144	<b>ń</b> 0147	<b>ń</b> 0148	<b>N</b> 2115	
<b>o</b> 004f	<b>o</b> 006f	<b>o</b> 00ba	<b>o</b> 00a9	<b>o</b> 24c4	<b>o</b> 24aa	<b>o</b> 2092
	<b>ò</b> 00d2	<b>ò</b> 00f2	<b>ó</b> 00d3	<b>ó</b> 00f3	<b>ò</b> 00d4	<b>ó</b> 00f4
	<b>ő</b> 00d5	<b>ő</b> 00f5	<b>ö</b> 00d6	<b>ö</b> 00f6	<b>ø</b> 00d8	<b>ø</b> 00f8
	<b>ó</b> 014c	<b>ó</b> 014d	<b>ó</b> 014e	<b>ó</b> 014f	<b>œ</b> 0152	<b>œ</b> 0153
<b>P</b> 0050	<b>p</b> 0070	<b>p</b> 24c5	<b>p</b> 24ab	<b>p</b> 209a		
<b>Q</b> 0051	<b>q</b> 0071	<b>q</b> 24c6	<b>q</b> 24ac	<b>Q</b> 211a		
<b>R</b> 0052	<b>r</b> 0072	<b>r</b> 24ad	<b>r</b> 24c7	<b>ŕ</b> 0154	<b>ŕ</b> 0155	
				<b>ŕ</b> 0158	<b>ŕ</b> 0159	<b>R</b> 211d
<b>S</b> 0053	<b>s</b> 0073	<b>s</b> 24c8	<b>s</b> 24ae	<b>s</b> 209b	<b>ś</b> 015a	<b>ś</b> 015b
	<b>ſ</b> 015e	<b>ſ</b> 015f	<b>ſ</b> 0160	<b>ſ</b> 0161	<b>þ</b> 00df	
<b>T</b> 0054	<b>t</b> 0074	<b>t</b> 24af	<b>t</b> 22a4	<b>t</b> 24c9	<b>t</b> 209c	
			<b>Ń</b> 0162	<b>ń</b> 0163	<b>Ń</b> 0164	<b>ń</b> 0165

<b>U</b> 0055	<b>u</b> 0075	<b>u</b> 24ca	<b>u</b> 24b0	<b>u</b> 1d64	<b>Ü</b> 00d9	<b>ü</b> 00f9
	<b>ú</b> 00da	<b>ú</b> 00fa	<b>ú</b> 00db	<b>ú</b> 00fb	<b>Ü</b> 00dc	<b>ü</b> 00fc
	<b>Ü</b> 0168	<b>ü</b> 0169	<b>Ü</b> 016a	<b>ü</b> 016b	<b>Ü</b> 016c	<b>ü</b> 016d
			<b>Ü</b> 016e	<b>ü</b> 016f	<b>Ü</b> 0172	<b>ü</b> 0173
<b>v</b> 0056	<b>v</b> 0076	<b>v</b> 24cb	<b>v</b> 24b1			
<b>w</b> 0057	<b>w</b> 0077	<b>w</b> 24cc	<b>w</b> 24b2	<b>ŵ</b> 0174	<b>ŵ</b> 0175	
<b>x</b> 0058	<b>x</b> 0078	<b>x</b> 1d61	<b>x</b> 24cd	<b>x</b> 24b3	<b>x</b> 2093	
			<b>ẍ</b> 0379	<b>ẍ</b> 0378	<b>ẍ</b> 037f	<b>ẅ</b> 221c
<b>ÿ</b> 0059	<b>y</b> 0079	<b>y</b> 24ce	<b>y</b> 24b4	<b>ÿ</b> 00dd	<b>ý</b> 00fd	
	<b>ÿ</b> 0176	<b>ÿ</b> 0177	<b>ÿ</b> 0178	<b>ÿ</b> 00ff	<b>ý</b> 0233	<b>ý</b> 0232
<b>ż</b> 005a	<b>z</b> 007a	<b>z</b> 24cf	<b>z</b> 24b5	<b>ż</b> 0179	<b>ż</b> 017a	<b>ż</b> 017b
			<b>ż</b> 017c	<b>ż</b> 017d	<b>ż</b> 017e	<b>ż</b> 2124
<b>Α</b> 0391	<b>α</b> 03b1	<b>α</b> 2065	<b>ά</b> 03ac			
<b>Β</b> 0392	<b>β</b> 03b2					
<b>Γ</b> 0393	<b>γ</b> 03b3					
<b>Δ</b> 0394	<b>δ</b> 03b4	<b>δ</b> 2066				
<b>Ε</b> 0395	<b>ε</b> 03b5	<b>έ</b> 03ad				
<b>Ζ</b> 0396	<b>ζ</b> 03b6					
<b>Η</b> 0397	<b>η</b> 03b7	<b>ή</b> 03ae				
<b>Θ</b> 0398	<b>θ</b> 03b8					
<b>Ι</b> 0399	<b>ι</b> 03b9	<b>ί</b> 03af	<b>ϊ</b> 03aa	<b>ϊ</b> 03ca	<b>՚</b> 0390	
<b>Κ</b> 039a	<b>κ</b> 03ba					
<b>Λ</b> 039b	<b>λ</b> 03bb					
<b>Μ</b> 039c	<b>μ</b> 03bc	<b>μ</b> 00b5	<b>μ</b> 2067			
<b>Ν</b> 039d	<b>ν</b> 03bd					
<b>Ξ</b> 039e	<b>ξ</b> 03be					
<b>Ο</b> 039f	<b>ο</b> 03bf	<b>ό</b> 03cc				
<b>Π</b> 03a0	<b>π</b> 220f	<b>π</b> 03c0				

P	03a1	ρ	03c1						
Σ	03a3	σ	03c3	ζ	03c2				
Τ	03a4	τ	03c4						
Υ	03a5	υ	03c5	ύ	03cd	ΰ	03ab	ΰ	03cb
Φ	03a6	φ	03c6						
Χ	03a7	χ	03c7						
Ψ	03a8	ψ	03c8						
Ω	03a9	ω	03c9	ώ	03ce				
(	0028	)	0029						
[	005b	Γ	23a1		23a2	Λ	23a3		
		]	005d	Ι	23a4		23a5	Ι	23a6
{	007b	}	007d						
▲	2430	▲	2431	▲	2432	▲	2433		
+	002b	+	207a	+	208a	±	00b1		
-	002d	-	207b	-1	2072	-	208b	≠	2213
×	00d7	·	00b7	•	2219	◦	2218	*	002a
/	002f	\	005c						
^	005e								
,	002c	;	2429						
.	002e	:	2428	...	2026				
!	0021	!	00a1						
?	003f	?	00bf						
:	003a	:	2236	÷	00f7				
;	003b								
'	0027	'	2018	'	2019	,	201a	'	201b
"	0022	"	201c	"	201d	,,	201e	"	201f
@	0040							«	00ab
-	005f	⌘	2427					»	00bb

~	007e								
→	2192	→	21c0						
←	2190	↖	21cd						
↑	2191	↑	21e7	▲	21c9	↑	242b		
↓	2193	↓	21e9	▼	21cb				
↗	21c4								
↔	2195								
☰	21cc								
¬	00ac								
Λ	2227	¥	2228	¥	22bb	฿	22bc	₪	22bd
&	0026								
	007c		2223		2224		2225		2226
«	226a	<	003c	≤	2264	≡	2261	:=	2254
		≈	2248	≡	2258	△	2259	≠	2260
								≥	2265
								>	003e
								»	226b
%	0025	\$	0024	€	20ac	¢	00a2	£	00a3
✓	221a	✗	221d						
∞	221e	♾	209e	♾	209f				
ʃ	222b	ʃ	222c	ʃ	222d	ƒ	222e	ƒ	222f
◎	2299	◎	229a	◎	2068			ֆ	2230
⊕	2295	⊕	2069						
↳	221f	↳	22a5						
↶	2220	↶	2221	↶	2222				
↑	2308	↑	2309						
↓	230a	↓	230b						
⌚	2399	⌚	231b	⌚	231a	⌚	242a	⌚	242c
#	0023							⌚	242f
UK	242d	US	242e						

▼ 2200	♂ 2202	Ξ 2203	‡ 2204	∅ 2205	▲ 2206	▼ 2207
	€ 2208	ƒ 2209	Ξ 220b	‡ 220c	₪ 2229	₪ 222a
↳ 2421	/ 2422	/ 2425	/ 2426			
✓ 2713						

## Stack Size

At a very early stage of this project (2013), *stack size* was discussed. 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). In this context 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. At the bottom line, eight *stack registers* turn out being sufficient for solving any real-world mathematical, scientific, or engineering problem (cf. *Section 1* of the OM as well as field experience with *WP 34S* and *WP 31S* since 2011).

After all, we decided sticking to *RPN* as implemented on the *WP 34S* and *WP 31S*. It covers everything needed most easily. For special action support, the commands STOS and RCLS are provided.

## 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, Σ+, and Σ– disable *automatic stack lift*, all other functions enable it. But compare INPUT on p. 45.

## Structured Programming

In 2013, I suggested the following control structures:

- IF ... THEN ... ELSE ... END,
- FOR ... FROM ... TO ... END,
- REPEAT ... UNTIL, and
- 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; and
- WHILE ... DO BEGIN ... END;

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

## UNDO

In 2013, UNDO was planned as it works in *HP-48*, recalling just the *stack* as it was before executing last command. The *WP 31S*, on the other hand, features an UNDO recalling the entire calculator status as it was before executing last command. It turned out that such a complete UNDO was viable on the *WP 43S* as well, so we implemented it for better user experience.

Please find here two surfaces printed to scale: at right the original keys and keyplate of your *WP 43S* and below its virtual keyboard in alpha input mode (*AIM*). Furthermore there are two pictures overleaf, one taken from the back of an *HP-16C* and the other inspired by a print on an *HP-11C*.

Choose your favorite, cut it out, and use it with your *WP 43S*. If you bought your *WP 43S* complete and flashed, all except the *HP-16C* related picture shall be found printed on it.



	C	G
$[+]$	*	*
$[-]$	--	x
$\times$	x	x
$\div$	x	x
$\sqrt{x}$	x	--
$\text{CHS}$	--	x
$\text{DBL } X$	--	o
$\text{DBL } \div$	x	o
$\text{SL}$	*	--
$\text{SR}$	*	--
$\text{ASR}$	*	--
$\text{RL}$	*	--
$\text{RR}$	*	--
$\text{RLC}$	*	--
$\text{RRC}$	*	--

Annotations:

- $\div$ :  $\text{RMD} \neq 0 \rightarrow C$
- $\sqrt{x}$ :  $\text{RMD} \neq 0 \rightarrow C$
- $\text{DBL } \div$ :  $(Y \& Z) \div X \rightarrow X$ ;  $\text{RMD} \neq 0 \rightarrow C$

