



WP 43S REFERENCE MANUAL

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

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

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

WP 43S will stay constant, however. Stay informed by watching 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. 156 and bottom of p. 157 were kindly supplied by SwissMicros as well as the drawing on p. 213, the picture on p. 211 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 (<i>IOI</i>)	13
0 - 9	17
A	18
B	21
C	24
D	29
E	33
F	35
G	38
H	40
I	41
J	44
K	45
L	46
M	50
N	55
O	57
P	58
Q	61
R	61
S	69
T	76
U	78
V	79
W	80
X	81
Y	85

Z	85
A, α	85
β	87
Γ, γ	87
Δ, δ	87
ε	87
ζ	88
Π, π	88
Σ, σ	88
X	90
(, +, -, \times , /, ^	91
\rightarrow ,	92
%	95
The Rest	96
Names of System Variables and System Flags Provided	99
System Flags	104
Nonprogrammable Commands and Keys	108
Command Parameter Input and Closing It	109
Alphanumeric Input in X and Closing It	111
Section 2: Menus and Catalogs	115
One to Find and Rule Them All – the CATALOG	116
Accessing Cataloged Items Rapidly	119
Further Menus and Their Contents	121
Constants	130
Unit Conversions	140
Section 3: Calling and Executing Operations	149
Using XEQ for Executing Operations	149
Operations Requiring Trailing Parameters	150
Operations Changing Data Types	152

Appendix A: Hardware	155
Appendix B: Memory Management	160
Data Types	160
Statistical Summation Registers	163
SAVE and LOAD...	163
Range of Real Numbers	164
Limitations	168
What Happens when Changing Word Size?	170
Special Results	171
Program Step Size	176
Appendix C: Messages and Error Codes	177
Appendix D: Comparison to the Function Sets of HP-42S, HP-16C, HP-21S, and WP 34S	183
Corresponding Operations on <i>HP-42S</i>	183
Corresponding Operations on <i>HP-16C</i>	189
Corresponding Operations on <i>HP-21S</i>	191
Corresponding Operations on <i>WP 34S</i>	193
New Commands on your <i>WP 43S</i>	197
Reference Literature	202
Appendix E: Emulating a WP 43S on Your Computer	204
Appendix F: Flashing and Updating Your WP 43S	210
How to Flash Your <i>WP 43S</i>	210
How to Update Your <i>WP 43S</i>	212
Overlays	213
Appendix G: Troubleshooting Guide	215
Calculator Frozen	215
Fresh Battery Constantly Low	216
Keymap Trouble	216

Appendix H: Advanced Mathematical Functions and Tasks 218

Number Generating Functions	218
Statistical Distributions	220
More Statistical Formulas, also for Fitting	229
About the Curve Fitting Models Provided	235
About Error Propagation	242
Solving Differential Equations	243
Orthogonal Polynomials	246
Even More Mathematical Functions	251

Appendix I: Information for Advanced Users 257

Recursive Programming	257
Building <i>WP 43S</i> Almost from Scratch	258
Index of Everything Provided	260

Appendix J: Release Notes 272

WP 43S Quick Reference Guide 1

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

Background Considerations and Facts

1

Accessing Items	1
Alpha Register	3
Angles	4
Backward Compatibility	4
Calculation Internals	5
Character Sets	5
Complex Notation and Storage	7
Complex Numbers close to Infinity	7
Display Limits	9
Display Segmentation	15
Echo and Fallback	17
Equations	17
Layouting	18
Menus	19
Number Range	19
Plotting?	20
Precision and Accuracy	22
Prefixes	22
Sorting in Detail	22
Stack Size	28
Stack Lift Disabling Functions	29
Structured Programming	29
UNDO	30

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

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. specific 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 1.234×10^{-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*).

DT = data type (see *Appendix B*).

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

GP = general purpose.

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.

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

SRS = subroutine return stack (see *App. B* on pp. 160ff).

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

Some more abbreviations may be used and explained locally.

DRAFT

SECTION 1: INDEX OF ITEMS (IOI)

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 in column 2 the keystrokes necessary to call them. Most *items* shall be picked from *menus* (see pp. 115ff). For such *items*, we list the keys calling the respective *menu*, the *prefix* of the respective 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 since they are sorted alphabetically and will be explained in detail in a separate chapter below.

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 printed on the bezel or displayed in menus, featuring less or more characters than its name (see some unit conversions, for example). These labels are not required to be unique.¹

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



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 *HP* 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

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

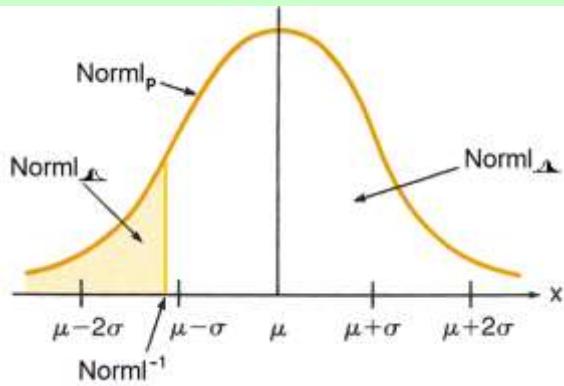
² Characters printed on grey background are inaccessible for users for the time being. The entire sorting table is printed in an appendix.

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**.³

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

Commands asking for confirmation are printed **red**. Those printed **red** or **orange** cannot be undone/reverted by UNDO.



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*.

³ 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.⁴

Operation or function **parameters** will be taken from the lowest stack register(s) unless mentioned explicitly in column 2 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.

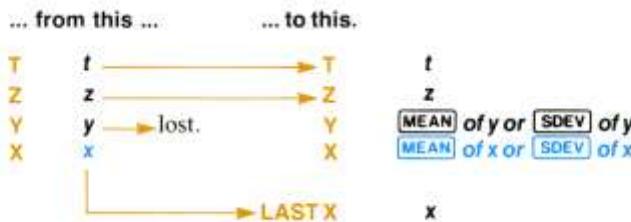
Three examples of the parameter notation used throughout the *IOI* are shown below. Assume **R12** contains **15.67** generally here, i.e. ***r12 = 15.67***.

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 RSD $\rightarrow 12$ rounds x to 15 significant digits.

⁴ On the *HP-42S*, however, also statistical functions returning two values do that – while the *Spices* (e.g. *HP-34C*) and *Voyagers* (e.g. *HP-15C*) push both results on the stack instead as you expect from *RPN* calculators. For your information, the picture below shows what the *HP-55*, *HP-19C/29C*, *HP-67/97*, *HP-41C*, and *HP-42S* do there:

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



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

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. 160ff. Many bit and integer functions make most sense operating on *short integers* (DT 10). Other functions typically work with more types of objects. Functions stating *DTs* 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 *DTs* differing from their input, the output types are listed as well.⁵

⁵ 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 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*.

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.⁶

Below, the functions checked already are highlighted green, those which don'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 *DTs*.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	 etc.	(1) {2}; {1} \rightarrow {2} Convert temperatures. See pp. 140ff.
$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$		
10^x		(1) {1, 2, 3, 8*, 9*, 10} Returns 10^x , the inverse of $\lg(x)$.
1COMPL	 	(0) Sets 1's complement mode for operations on <i>short integers</i> . Indicated in the <i>status bar</i> . See Section 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 .

⁶ Some reasoning why *automatic stack lift* is disabled for these four:

- a) CLX is for clearing \mathbf{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	(0) Sets 2's complement mode for operations on <i>short integers</i> . Indicated in the <i>status bar</i> . See Section 2 of the OM.
	g [INTS] ▲ 2COMPL	
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> .
ABS	f [CAT.] FCNS ABS	Points to $ x $ on p. 94. Maintained for backward compatibility only.
ACOS	f [CAT.] FCNS ACOS	Points to arccos on p. 19.
$ac \rightarrow m^2$	f [U→] f A: acre → m^2	(1) {2}; {1} → {2} Convert areas. See pp. 140ff.
$ac_{us} \rightarrow m^2$	f [U→] f A: acre _{us} → m^2	
ADV	f [ADV]	Menu. See p. 122.
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. 251 for more.
AGRAPH	g P.FN P.FN2 g AGRAPH s	(0) Alpha graphics. Displays a graphics image. Each character in the source s specifies an 8-dot-1-column pattern. The X - and Y -registers specify the pixel location of the bottom left point of this column (valid inputs are $1 \leq x \leq 400$ and $1 \leq y \leq 232$). 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. Cf. <i>HP-42S Owner's Manual</i> , pp. 135 – 140, and <i>HP-42S Programming Examples & Techniques</i> , pp. 214 – 223.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
ALL	g DISP ALL n	<p>(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 HP-42S almost.</p> <p>For $x \geq 10^{16}$ (or earlier for <i>complex</i> numbers), display will switch to SCI or ENG (see ALLENG) with the maximum number of necessary decimals displayable using the large font. 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. 62.</p>
AND	f BITS AND	<p>(2) {10}</p> <p>Works bitwise as in HP-16C (see the OM, Section 2).</p> <p>(2) {1, 2} → {1}</p> <p>Works like AND in HP-28S, i.e. x and y are interpreted before executing this operation. Zero is ‘false’; any other <i>real</i> number is ‘true’.</p>
ANGLES	f CAT. VARS ANGLES	Submenu of tagged angular variables defined at execution time. See pp. 116f.
arccos	TRI arccos	<p>(1) {3, 8*, 9*}; {1, 2} → {4};</p> <p>Returns the tagged angle $\text{arccos}(x)$.⁷</p>
arcosh	g EXP g arcosh	<p>(1) {2, 3, 8*, 9*}</p>
	TRI g arcosh	<p>Returns $\text{arcosh}(x)$.</p>

⁷ Precisely, ARCCOS returns the principal value of $\text{arccos}(x)$, i.e. a *real* part $\in [0, \pi]$ in 4^{R} , or $\in [0, 1]$ in 4π , or $\in [0^\circ, 180^\circ]$ in 4° or 4^{d} , or $\in [0^{\text{g}}, 200^{\text{g}}]$ in 4^{g} . Cf. ISO/IEC 9899.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
arcsin	TRI arcsin	(1) {3, 8*, 9*}; {1, 2} → {4}; Returns the tagged angle $\text{arcsin}(x)$. ⁸
arctan	TRI arctan	(1) {3, 8*, 9*}; {1, 2} → {4}; Returns the tagged angle $\text{arctan}(x)$. ⁹
arsinh	g EXP g arsinh	(1) {2, 3, 8*, 9*}
	TRI g arsinh	Returns $\text{arsinh}(x)$.
artanh	g EXP g artanh	(1) {2, 3, 8*, 9*}
	TRI g artanh	Returns $\text{artanh}(x)$.
ASIN	f CATALOG FCNS ASIN	Points to ARCSIN above. Maintained for backward compatibility only.
ASR	f BITS f ASR <i>n</i>	(1) {10}  Works like <i>n</i> (≤ 63) consecutive ASR commands in HP-16C, corresponding to a division of <i>x</i> by 2^n . ASR 0 executes as NOP, but loads L. See Section 2 of the OM.
ASSIGN	f ASN <i>item, location</i>	(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	f CATALOG FCNS ATAN	Points to ARCTAN above. Maintained for backward compatibility only.

⁸ Precisely, ARCSIN returns the principal value of $\text{arcsin}(x)$, i.e. a *real* part $\in [-\pi/2, \pi/2]$ in 4r , or $\in [-0.5, 0.5]$ in $\text{4}\pi$, or $\in [-90^\circ, 90^\circ]$ in 4° or $\text{4}''$, or $\in [-100^\circ, 100^\circ]$ in 4° . Cf. ISO/IEC 9899.

⁹ Precisely, ARCTAN returns the principal value of $\text{arctan}(x)$, i.e. a *real* part $\in [-\pi/2, \pi/2]$ in 4r , for example (cf. ASIN), if SPCRES is set. Else the result interval for ATAN becomes $(-\pi/2, \pi/2)$ in 4r , for example. Cf. ISO/IEC 9899.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
atm→Pa	f U→ F&p: atm→Pa	(1) {2}; {1} → {2}
au→m	f U→ x: au→m	Convert pressures and distances. See pp. 140ff.
A:	f U→ f A:	Submenu. See p. 140.
BACK	g P.FN P.FN2 g BACK n	<p>(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 F. See also SKIP.</p> <p>ATTENTION: If you edit a section of your routine crossed by one or more BACK, SKIP, or CASE jumps, this may well result in a need to manually maintain all those statements individually.</p>
bar→Pa	f U→ F&p: bar→Pa	(1) {2}; {1} → {2}
		Converts pressures. See pp. 140ff.
BATT?	g INFO f BATT?	(-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 ³	f U→ f v: f barrel → m ³	(1) {2}; {1} → {2}
		Converts volumes. See pp. 140ff.
BC?	f BITS g BC? n	(-1) {10}
		Tests if the specified bit in x is clear.
BEEP	f I/O f BEEP	(0) Sounds a sequence of four tones. See also TONE.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
BeginP		(0) Sets “Begin” mode in <i>TVM</i> : payments occur at the beginning of each period. Typical for savings plans and leasing. Compare ENDP.
BestF		<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 <i>BEST</i> in <i>HP-42S</i>).</p> <p>Relevant for L.R., CORR, COV, s_{XY}, \hat{x}, and \hat{y}. 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 n carries this information. Each fit model corresponds to a number as listed:</p> <ul style="list-style-type: none"> • LINF 1 • EXPF 2 • LOGF 4 • POWERF 8 • ROOTF 16 • HYPF 32 • PARABF 64 • CAUCHF 128 • GAUSSF 256 <p>Take the numbers of all models you can exclude and sum them up – the result is n.</p> <p>Example: Excluding the three 3-parameter models leads to $n = 64 + 128 + 256 = 448$. So call BESTF 448 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. 220ff for more.</p>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
BestF?	g INFO ▼ BestF?	(-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.
Binom _p	g PROB g Binom: Binom_p etc.	(1) {2} <i>Binomial distribution</i> with the number of successes g in X , the probability of a success p_o in I , and the sample size n in J . See p. 220 for more.
Binom ₋₁		Binom ⁻¹ returns the maximum number of successes m for a given probability p in X , p_o in I and n in J .
Binom:	g PROB g Binom:	Submenu. See p. 125.
BITS	f BITS	Menu. See p. 119.
B _n	g X.FN B_n	(1) {1, 2} B _n and B _n [*] return the Bernoulli number for an integer n > 0 given in X , working with different definitions (see both formulas on p. 218).
BS?	f BITS g BS? n	(0) {10} Tests if the specified bit in x is set.
Btu→J	f U→ E: Btu→J	(1) {2}; {1} → {2} Converts energies. See pp. 140ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
cal→J	f U E: cal→J	(1) {2}; {1} → {2} Converts energies. See pp. 140ff.
CASE	g P.FN P.FN2 g CASE s	(0) Works like SKIP below but takes the number of steps to skip from s. Example: Assume a program section: <pre> ... 100 CASE 12 101 GTO 01 102 GTO 02 103 GTO 07 104 GTO 05 105 LBL 01 ... 132 LBL 02 ... 153 LBL 05 ... 234 LBL 07 ... </pre> <p>In execution of this program, <i>r12</i> will be checked in step 100: if $r12 \leq 1$ then the program will proceed to step 101 and continue with a jump to step 105, for $r12 = 2$ the program will go to step 102, etc., resulting in a nice controlled dispatcher for $1 \leq r12 \leq 4$.</p> <p>ATTENTION: CASE might surprise you for $r12 > 4$ in the example above. Take care of the input you provide!</p> <p>If you edit a section of your routine crossed by one or more BACK, SKIP, or CASE jumps, this may well result in a need to manually maintain all those statements individually.</p>
CATALOG	f CATALOG	Catalog of everything. See pp. 116ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
CauchF	f [STAT] ▼ f CauchF	(0) Selects the <i>Cauchy</i> (a.k.a. <i>Lorentz</i>) peak fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 229ff for more.
Cauch_p	g [PROB] f Cauch: Cauch_p	(1) {2}
Cauch_x	etc.	<i>Cauchy-Lorentz</i> (a.k.a. <i>Lorentz</i> or <i>Breit-Wigner</i>) <i>distribution</i> with the location x_0 specified in I and the shape γ in J . See p. 223 for more.
Cauch₋₁		Cauch^{-1} returns x for a given probability p in X , with x_0 in I and γ in J .
Cauch:	g [PROB] f Cauch:	Submenu. See p. 125.
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 $\geq x$. Compare FLOOR.
CF	g [FLAGS] CF n g [MODE] CF n g [CLR] CF n	(0) Clears the <i>flag</i> specified, i.e. sets it to 0 .
CHARS	f [CATALOG] CHARS	Submenu of characters. See pp. 116ff.
CLALL	g [CLR] g CLall	(0) Clears all <i>registers</i> , <i>user flags</i> , variables, and programs in <i>RAM</i> . Modes will stay as they are. See also CLCVAR, CLFALL, CLPALL, CLREGS, CLSTK, and RESET.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
CLCVAR	g CLR CLCVAR	(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 0. , all integer ones to 0 , all <i>time</i> variables to 0:00:00 , all <i>date</i> variables to January 1st of year 0 , all <i>text strings</i> to zero length, and all the elements of all matrix variables used to 0 .
CLFALL	g FLAGS g CLFall g CLR f CLFall	(0) Clears all global and local <i>user flags</i> . Compare CF.
CLK	g CLK	<i>Menu</i> . See p. 115.
CLLCD	g CLR f CLLCD	(0) Clears the <i>LCD</i> in the rectangular window north and west of the point <i>x, y</i> . I.e. all pixels $\geq x$ and $\geq y$ are cleared.
CLMENU	g CLR CLMENU g P.FN P.FN2 ...	(0) Clears all <i>menu key</i> definitions for the programmable <i>menu</i> . See MENU.
CLP	g CLR CLP	(0) Clears the <i>current program</i> in <i>RAM</i> or <i>FM</i> . Freed memory is returned to the pool of free space.
CLPALL	g CLR f CLPall	(0) Clears <u>all</u> programs in <i>RAM</i> . Cf. CLP.
CLR	g CLR	<i>Menu</i> . See p. 122.
CLREGS	g CLR f CLREGS	(0) Clears all global and local <i>GP registers</i> allocated (see also LOCR), i.e. sets all these registers to 0 . The contents of the <i>stack</i> and L are kept.
CLSTK	g CLR f CLSTK 0 f FILL	Clears all <i>stack registers</i> currently allocated (i.e. either X ... T or X ... D). All other <i>register</i> contents are kept. Cf. CLREGS.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
CLX	g CLR CLX ⬅ (for closed x)	(1) Clears stack register X, disabling automatic stack lift. Cf. CLREGS and CLSTK.
CLΣ	g CLR CLΣ f STAT g CLΣ	(0) Clears the statistical summation registers and releases the memory allocated for them (see p. 163).
CNST	g P.FN f CNST n	(-1) {} → {2} Returns the constant stored at position n in CONST (see below and pp. 130ff). Allows for indirectly addressing these constants, also beyond ∞ .
COMB	g PROB C_{yx}	(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. Compare PERM. (2) {2, 3} See pp. 218f for the formula.
CONJ	g CPX conj	(1) {3, 9*} Returns the complex conjugate of x .
CONST	f CONST	Menu. Cf. CNST above and pp. 130ff.
CONVG?	g TEST g CONVG? r	(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: 0 = compare the numbers x and y relatively, 1 = compare them absolutely.</p> <p>c) The top bit tells how special numbers are treated: 0 = NaN and $\pm\infty$ are considered converged, 1 = they are not considered converged.</p> <p>Now, $r = a + 4b + 16c$.</p>
CORR	f STAT ▲ r	(-1) {} → {2} Returns the <i>coefficient of correlation</i> for the current statistical data and curve fit model. See pp. 229ff for more.
cos	TRI cos	(1) {2, 3, 8*, 9*}; {1, 4} → {2} Returns the cosine of the angle in X (see Section 2 of the OM for details).
cosh	g EXP g cosh TRI g cosh	(1) {2, 3, 8*, 9*} Returns the hyperbolic cosine of x .
COV	f STAT ▲ cov	(-1) {} → {2} Returns the population covariance for the two data sets entered via $\Sigma+$, depending on the curve fit model selected. See s_{XY} for the sample covariance and pp. 220ff for more.
CPX	g CPX	Menu. See p. 122.
CPXS	f CAT. VARS CPXS	Submenu of complex variables defined at execution time. See pp. 116f.
CPX?	g TEST ▲ CPX?	(0) Checks if x is <i>complex</i> . Returns true if X contains data of type 3 or 9 with nonzero imaginary part.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
CROSS	f MATX f cross	(2) {8} Requires two <i>real</i> 2D or 3D vectors in X and Y and returns their cross product. Crossing of 2D vectors works as for <i>complex</i> numbers.
	g CPX cross	(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	f U→ m: f carat → kg	(1) {2}; (1) → {2}
cwt→kg	f U→ m: cwt→kg	Convert masses. See pp. 140ff.
CX→RE	CC (works in run mode only)	(-1) {3} → {2}; (9) → {8}
	g CPX f CX→RE	Cuts a closed <i>complex</i> object <i>x</i> , putting either <ul style="list-style-type: none"> (for L) its <i>real</i> part in Y and its <i>imaginary</i> part in X or (for Ø) <i>magnitude</i> in Y and <i>phase</i> in X.
DATE	g CLK DATE	(-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	f CATALOG VARS f DATES	Submenu of <i>date</i> variables defined at execution time. See pp. 116f.
DATE→	g CLK DATE→	(-2) {2, 6} → {1} Assumes <i>x</i> containing a <i>date</i> in the format selected (or a <i>real</i> number in corresponding format) and pushes its three components as integers on the stack. Reversible by →DATE.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
DAY	g CLK f DAY	(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 day.
DBLR	g INTS g DBLR etc.	{10}
DBLx		Double word length commands for remainder, multiplication and division (see the HP-16C OH, Section 4, pp. 52ff).
DBL/		DBLR and DBL / accept a double size dividend in Y and Z (most significant bits in Y), the divisor in X as usual, and return the result in X . DBLx takes x and y as factors as usual but returns the product in X and Y (most significant bits in X).
dB→fr	f U→ ▲ dB → field ratio	(1) {2}; {1} → {2}
dB→pr	f U→ ▲ dB → power ratio	Convert ratios. See pp. 140ff.
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 <i>stack</i> [<i>denominator</i> (x), <i>numerator</i> (x), ...]. This works with maximum precision always, regardless of DENMAX, DENFIX, and DENANY. Example: If X contains 2.25 then DECOMP will return $x = 4$ and $y = 9$.
DEG	g MODE DEG	(0) Sets the <i>ADM</i> 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. 147.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
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. 94. Maintained for backward compatibility only.
DIGITS	f CAT. f DIGITS	Submenu of digits defined. See pp. 116ff.
DISP	g DISP	Menu. See p. 122.
DOT	g CPX dot	(2) {3} → {2} Returns $Re(x) \cdot Re(y) + Im(x) \cdot Im(y)$
	f MATX f dot	(2) {8} → {2}; {9} → {3} Requires two matrices in x and y and returns their dot (scalar) product. The dot product is defined as the sum of the products of the corresponding elements in both matrices. Note both matrices must be of the same size; else DOT will throw an error. See the OM, Sect. 2.
DROP	g DROP↓	Drops x ... from the stack. See Sect. 1 of the OM for details.
DROPy	g STK DROPy	Drops y
DSE	f LOOP DSE r	(0) {1, 2, 10} Given $cccc.c.fffii$ in the source, DSE decrements r by ii , skipping next program step if then $cccc \leq fff$ (cf. the rear side of your WP 43S). If r features no fractional part then fff is 0. If $ii = 0$, $cccc$ 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 $cccc > 0$.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
DSL		(0) {1, 2, 10} Works like DSE but skips if $cccccc < ffff$.
DSTACK		(0) Sets the maximum number of <i>stack registers</i> displayed. For an input of 1, only x will be shown directly above the <i>menu section</i> ; for 2, x and y will be displayed; maximum input is 4. Expanded views of e.g. matrices and multi-level returns like SUM will work as described in the OM regardless of the DSTACK parameter set. In any case, command input will be echoed directly below the <i>status bar</i> . 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.
DSZ		(0) {1, 10} Decrement r by 1 and skips the next step if $r = 0$ thereafter. Does not load L even for target address X . Cf. HP-16C.
D.MS	 (for closed input)	(0) Sets the <i>ADM</i> to <i>sexagesimal degrees</i> . Indicated in the <i>status bar</i> .
D.MS→		(1) {1, 2} → {4}
D.MS→D		Convert angles as described on pp. 147f.
D.MY		(0) Sets the format dd.mm.yyyy for <i>dates</i> .
D→D.MS		(1) {1, 2} → {4} Converts angles as described on p. 147.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
D→J		(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 converts it to a <i>Julian day number</i> ¹⁰ according to the J/G setting.
D→R		(1) {1, 2} → {4} Converts angles as described on p. 147.
EIGVAL		(-1) {8, 9} Evaluates the matrix x and pushes a diagonal matrix containing its eigenvalues on the stack.
EIGVEC		(-1) {8, 9} Evaluates the matrix x and pushes a matrix containing its eigenvectors on the stack.
END		(0) Last command in a program and terminal for searching local labels as described in the OM, Sect. 3. Works like RTN in all other aspects.
ENDP		(0) Sets “End” mode in TVM: payments occur at the end of each period. Typical for loans and investments. Cf. BEGINP.
ENG		(0) Sets engineer’s display format (see Section 2 of the OM).
ENORM		(1) {8, 9} → {2} Calculates the Euclidean norm of the matrix in \mathbf{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. 94.

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

Item	Keystrokes	Remarks (see pp. 13ff for general information)
ENTER↑	[ENTER↑]	(-1) Separates two entries in input. Copies x into Y, disabling <i>automatic stack lift</i> . See p. 112 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"> any character is entered in A/M, or any command is accepted for entry (be it via [ENTER↑], a function key, or [R/S] with a partial command line). Useful in routines, e.g. after PAUSE.
EQN	g [EQN]	Menu. See p. 123.
EQ.DEL	g [EQN] f DELETE	Deletes an equation.
EQ.EDI	g [EQN] EDIT	Opens the <i>Equation Editor</i> to edit an existing equation.
EQ.NEW	g [EQN] NEW	Opens the <i>Equation Editor</i> to enter a new equation.
erf	g [X.FN] erf etc.	(1) {2}; {1} → {2} Returns the error function or its complement. See pp. 251ff for more.
erfc		
ERR	g [P.FN] ERR n	(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. 176ff for the respective error codes. Compare MSG.
EVEN?	g [TEST] f EVEN?	(0) Checks if x is integer and even.
e ^x	e ^x	(1) {2, 3, 8*, 9*}; {1, 10} → {2} Returns e^x .
EXITALL	g [P.FN] P.FN2 EXITall	(0) Exits all menus.

See Section 4 of
the OM.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
EXP	g EXP	Menu. See p. 123.
ExpF	f STAT ▼ ExpF	(0) Selects the exponential curve fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 220ff for more.
Expon _p	g PROB	(1) {2}
Expon _▲	f Expon: Expon _p etc.	<i>Exponential distribution</i> with the rate λ in J . See pp. 220ff for more.
Expon _▲		
Expon ⁻¹		Expon ⁻¹ returns the survival time t_s for a given probability p in X , with λ in J .
Expon:	g PROB f Expon:	Submenu. See p. 125.
EXPT	f PARTS EXPT	(1) {1, 2} → {1} Returns the exponent h of the number $x = m \cdot 10^h$ displayed. Compare MANT.
$e^x - 1$	g EXP 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:	f U→ E:	Submenu. See p. 138.
FB	f BITS g FB n	(1) {10} Inverts ('flips') the specified bit in x .
FBR	g a.FN g FBR	(0) Font browser. Shows all characters implemented in the fonts designed for your <i>WP 43S</i> .
FCNS	f CATALOG FCNS	Submenu of all functions. See pp. 116ff.
FC?	g FLAGS FC? n	(0) Tests if the specified <i>flag</i> is clear.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
FC?C	g FLAGS f FC?C <i>n</i> etc.	(0) Tests if the specified flag is clear. Clears, flips, or sets this flag after testing, respectively.
FC?F		
FC?S		
FF	g FLAGS FF <i>n</i>	(0) Flips the flag specified.
FIB	g X.FN f FIB	(1) {1} Returns the Fibonacci number (4791 is maximum input), (1) {2, 3} Returns the extended Fibonacci number, see pp. 218f.
FILL	f FILL	Copies <i>x</i> to all stack registers.
FIN	g FIN	Menu. See p. 123.
FIX	g DISP FIX <i>n</i>	(0) Sets fixed point display format (see the OM, Sect. 2).
FLAGS	g FLAGS	Menu. See p. 123.
FLASH	f CATALOG PROGS FLASH	Submenu of global labels defined at execution time. See pp. 116f.
FLASH?	g INFO f FLASH?	(-1) {} → {1} Returns the number of free bytes in FM.
FLOOR	g INTS g FLOOR	(1) {8*}; {1, 2} → {1} Returns the greatest integer ≤ <i>x</i> . Cf. CEIL.
fm.→m	f U→ <i>x</i> : ▲ fathom → m	(1) {2, 11; {1}} → {2} Converts distances. See pp. 140ff.
FP	f PARTS FP	(1) {1, 2, 8*, 10} Returns the fractional part of <i>x</i> . Compare IP.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
FP?	[g] TEST [f] FP?	(0) Tests x for having a fractional part $\neq 0$.
$F_p(x)$	[g] PROB [F:] $F_p(x)$	(1) {2}
$F_{\Delta}(x)$	etc.	<i>Fisher's F distribution.</i> $F_e(x)$ equals $Q(F)$ on HP-21S. The degrees of freedom are specified in I and J . See pp. 220ff for more.
$F_{\Delta}(x)$		
$F^{-1}(p)$		
fr→dB	[f] U→ ▲ field ratio → dB	(1) {2}; {1} → {2} Converts ratios. See pp. 140ff.
FS?	[g] FLAGS FS? n	(0) Tests if the specified flag is set.
FS?C	[g] FLAGS	(0) Tests if the specified flag is set. Clears,
FS?F	[f] FS?C n	flips, or sets this flag after testing, respectively.
FS?S	etc.	
ft.→m	[f] U→ x: [f] ft.→m	(1) {2}; {1} → {2} Convert distances and volumes, respectively. See pp. 140ff.
ft _{us} →m	[f] U→ x: ▲ survey foot _{us} → m	
ft _{uk} →m ³	[f] U→ [f] V: [f] floz _{uk} → m ³	
ft _{us} →m ³	[f] U→ [f] V: [f] floz _{us} → m ³	
F:	[g] PROB F:	Submenu. See p. 125.
f'	[g] EQN f'	Submenus for calculating the 1 st or 2 nd derivative of a given equation. See the OM (Sect. 4) for more.
f''	[g] EQN f''	

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$f'(x)$	 $f'(x)$ <u>lbl</u>	$\{1, 2\} \rightarrow \{2\}$ $f(x)$ [$f'(x)$] returns the 1 st [2 nd] derivative of the function $f(x)$ at position x . This $f(x)$ must be specified in a routine starting with LBL <u>lbl</u> . On return, Y, Z, and T will be cleared and the position x will be in L. See Section 4 of the OM for more.
$f''(x)$	 $f''(x)$ <u>lbl</u>	ATTENTION: $f(x)$ and $f'(x)$ fill all stack registers with x before calling the routine specified.
F&p:	 F&p:	Submenu. See p. 140.
GAP	 GAP <u>n</u>	(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	 GaussF	(0) Selects the Gauss peak fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 229ff for more.
GCD	 GCD	(2) {1; 10} Returns the Greatest Common Divisor of x and y . ¹¹ This will always be positive.
g_d	 g_d etc.	(1) {2, 3}; {1} $\rightarrow \{2\}$
g_d^{-1}		Returns the Gudermannian function or its inverse. See p. 252 for details.

¹¹ $\text{GCD}(x, y) = \left| \left(x / \text{LCM}(x, y) \right) \times y \right|$. 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)
Geom_p	g PROB g Geom: Geom_p etc.	(1) {2} <i>Geometric distribution:</i> The CDF returns the probability for a first success after $m=x$ Bernoulli experiments. The probability p_o for a success in each such experiment must be specified in J . See pp. 220ff for more.
Geom_Δ		
Geom_Δ		
Geom⁻¹		Geom ⁻¹ returns the number of failures f before 1 st success for given probabilities p in X , p_o in J .
Geom:	g PROB f Geom:	Submenu. See p. 125.
gl_{uk}→m³	f U→ f v: gl_{uk}→m³ etc.	(1) {2}; {1} → {2}
gl_{us}→m³		Convert volumes. See pp. 140ff.
GRAD	g MODE GRAD	(0) Sets the <i>ADM</i> to <i>grades</i> (a.k.a. <i>gradian</i> or <i>gon</i>). Indicated in the <i>status bar</i> .
GRAD→	g L→ f GRAD→	(1) {1, 2} → {4} Converts angles as described on pp. 147f.
GTO	f GTO labl	(0) In <i>PEM</i> , inserts an unconditional branch to labl . Else positions the program pointer to labl .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
GTO.	f GTO . n	to step n (specify up to four digits until becoming unambiguous in used program memory).
	f GTO . labl	to the global label specified.
	f GTO . ▲	(0) Positions the program pointer ...
	f GTO . ▼	directly after <u>previous</u> END, going to the top of <i>current program</i> (see Sect. 3 of the OM).
	f GTO . .	directly after <u>next</u> END, going to the top of next program.
ha → m ²	f U- f A: ha → m ²	(1) {2}; {1} → {2} Converts areas. See pp. 140ff.
	g X.FND Orthog Hn	(2) {2}; {1} → {2}
H _n	... Orthog f Hnp	<i>Hermite polynomials</i> for probability (H _n) and physics (H _{np}). See p. 246 for details.
hp _E → W	f U- P: hp _E → W	(1) {2}; {1} → {2}
hp _M → W	etc.	
hp _{UK} → W	etc.	
Hyper _p	g PROB g Hyper: Hyper _p	(1) {2}
Hyper _▲	etc.	<i>Hypergeometric distribution</i> with the number of successes g in X , the probability of a success p₀ in I , the sample size n in J , and the batch size n₀ in K . See pp. 220ff for the formula.
Hyper _△		
Hyper ⁻¹		Hyper ⁻¹ returns the maximum number of successes m for a given probability p in X , p₀ in I , n in J , and n₀ in K .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
Hyper:		Submenu. See p. 125.
HypF		(0) Selects the hyperbolic fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 229ff for more.
IDIV		(2) {1, 10}; {2} → {1} Integer division, working like . See the OM, Sect. 2, for the DT of the quotient.
IDIVR	 	{1, 2, 10} Like IDIV but also returns the remainder in Y. See the OM, Section 2, for the resulting DTs of quotient and remainder.
iHg→Pa	 	(1) {2}; {1} → {2} Converts pressures. See pp. 140ff.
Im	 	(1) {2, 3} → {2}; {9} → {8}; Returns the imaginary part of x . Compare RE.
INC		(0) {1, 2, 10} Increments r by 1. Does not load L even for target address X.
INDEX	 	(1) Indexes a named matrix. You can also index a matrix by editing it (see M.EDIT or M.EDIN). After exiting the Matrix Editor, the matrix is no longer indexed. See also Matrix Utility Functions in the HP-42S Owner's Manual, pp. 223ff.
INFO		Menu. See p. 123.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
INPUT	g [P.FN] INPUT <i>r</i>	Works in programs only: Recalls the content of the source specified into X , displays the name of the source along with <i>r</i> , and halts program execution, allowing you to enter or calculate a value; pressing R/S then stores <i>x</i> 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 <i>name</i> undefined at execution time, INPUT automatically creates the variable with an initial value of zero.
INTS	g [INTS]	Menu. See p. 123.
INT?	g [TEST] f INT?	(0) Tests <i>x</i> for being an integer, i.e. having a fractional part equal to zero. Cf. FP?.
INVRT	f [CATALOG] FCNS INVRT	Works like $[M]^{-1}$ on p. 94. Maintained for backward compatibility only.
in. \rightarrow m	f [U \rightarrow] <i>x</i> : g in. \rightarrow m	(1) {2}; {1} \rightarrow {2} Converts distances. See pp. 140ff.
IP	f [PARTS] IP	(1) {1, 8*, 10}; {2} \rightarrow {1} Returns the integer part of <i>x</i> . Cf. FP.
ISE	f [LOOP] ISE <i>s</i>	(0) {1, 2, 10} Given ccccc.f ffi in the source <i>s</i> , ISE increments <i>s</i> by <i>ii</i> , skipping next program step if $cccc \geq ffff$ then (cf. the rear side of your WP 43S). If <i>s</i> has no fractional part then <i>ffff</i> = 0 and <i>ii</i> = 0. If <i>ii</i> = 0, ccccc will be incremented by 1. ISE does not load L even for target address X . Note that neither <i>ffff</i> nor <i>ii</i> can be negative, but ccccc can.

Item	Keystrokes	Remarks (see pp. 13ff for general information)												
ISG	f [LOOP] ISG s	(0) {1, 2, 10} Works like ISE but skips if ccccc > fff.												
ISM?	g [INFO] ISM?	(-1) {} → {1} Returns the <i>integer sign mode</i> set for <i>short integers</i> : <table style="margin-left: auto; margin-right: auto;"><tr><td>true</td><td>2</td><td>for 2's complement,</td></tr><tr><td>true</td><td>1</td><td>for 1's complement,</td></tr><tr><td>false</td><td>0</td><td>for unsigned, or</td></tr><tr><td>true</td><td>-1</td><td>for sign & mantissa mode.</td></tr></table>	true	2	for 2's complement,	true	1	for 1's complement,	false	0	for unsigned, or	true	-1	for sign & mantissa mode.
true	2	for 2's complement,												
true	1	for 1's complement,												
false	0	for unsigned, or												
true	-1	for sign & mantissa mode.												
ISZ	f [LOOP] ISZ s	(0) {1, 10} Increments s by 1, skipping next program step if s = 0 thereafter. ISZ does not load L even for target address X. Cf. HP-16C.												
I _{xyz}	g [X.FN] f I _{xyz}	(3) {1, 2} Returns the <i>regularized Beta function</i> . See p. 252.												
IΓ _p	g [X.FN] f IΓ _p	(2) {1, 2}												
IΓ _a	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. 123.												

Item	Keystrokes	Remarks (see pp. 13ff for general information)
J _y (x)	g X.FN g J _y (x)	(2) {2}; {1} → {2} J _y (x) returns the <i>Bessel function of first kind</i> and order y . See p. 253 for details.
J+	f MATX ▲ 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-	f MATX ▲ J-	
J/G	g CLK ▲ f J/G	(0) {2, 6} Sets the date the Gregorian calendar was introduced in the region you are interested in. See <i>Dates</i> in Section 2 of the OM.
J→Btu	f U→ E: J→Btu etc.	(1) {2}; {1} → {2}
J→cal		Convert energies. See pp. 140ff.
J→D	g CLK f J→D	(1) {1} → {6} Takes x as a <i>Julian day number</i> ¹² and converts it to a common date according to J/G (see above) and the date format selected.
J→Wh	f U→ E: J→Wh	(1) {2}; {1} → {2} Converts energies. See pp. 140ff.

¹² 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)
KEY		See KEYG and KEYX below.
KEYG	g P.FN P.FN2 KEYG <u>key#</u> , <u>labl</u>	Defines the label to be branched to (KEYG) or called (KEYX) when a particular softkey is pressed. KEYG and KEYX work in PEM only and will be translated to a program step
KEYX	g P.FN P.FN2 KEYX <u>key#</u> , <u>labl</u>	KEY <u>key#</u> GTO <u>labl</u> or KEY <u>key#</u> XEQ <u>labl</u> , respectively. Key numbers go from 1 to 18 with 1 corresponding to F1 , 9 to f F3 , and 14 to g F2 , for example.
KEY?	g TEST g KEY? r	(0) Tests if a key was pressed while a routine was running or paused. If <u>no</u> key was pressed in that interval, the next program step after KEY? will be executed; else it will be skipped and the code of said key will be stored in <u>r</u> . Key codes reflect the rows and columns on the keyboard (see the OM, Sect. 3; cf. GETKEY on HP-42S).
kg→ct	f U→ m: f kg → carat	(1) {2}; {1} → {2} Convert masses. See pp. 140ff.
kg→cwt	f U→ m: kg→cwt etc.	
kg→lb.		
kg→oz		
kg→scw	f U→ m: f kg → sh.cwt	
kg→sto	f U→ m: f kg → stone	
kg→s.t	f U→ m: ▲ kg → short ton	
kg→ton	f U→ m: ▲ kg→ton	
kg→trz	f U→ m: f kg → tr.oz	

Item	Keystrokes	Remarks (see pp. 13ff for general information)
KTYP?	g INFO KTYP? r	<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"> • 0 ... 9 if it corresponds to a digit 0 ... 9, • 10 if it corresponds to [,], or [+/-], • 11 if it corresponds to [f] or [g], • 13 if it corresponds to a softkey, and • 12 if it corresponds to any other key. <p>May help in user interaction with routines (see the OM, Section 3)..</p>
LASTx	RCL L	(-1) See Sect. 1 of the OM. Actually, this command will be recorded as RCL L in routines.
lbf→Nm	f U→ ▶ lbf-ft → Nm	(1) {2}; {1} → {2}
lbf→N	f U→ F&p: lbf→N	Convert torques and forces. See pp. 140ff.
LBL	g LBL <i>label</i>	(0) Identifies programs and routines for execution and branching. Read more about labels and specifying them in Section 3 of the OM.
LBL?	g TEST g LBL? <i>label</i>	(0) Tests for existence of the label specified, anywhere in program memory. See LBL for more.
lb.→kg	f U→ m: lb.→kg	(1) {2}; {1} → {2} Converts masses. See pp. 140ff.
LCM	g INTS f LCM	(2) {1; 10} Returns the Least Common Multiple of x and y. ¹³ This will always be positive. Cf. GCD.

¹³ $\text{LCM}(x, y) = \left| \left(\frac{x}{\text{GCD}(x, y)} \right) \times y \right|.$

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 _p	 	(1) {2}
LgNrm _μ	etc.	Log-normal distribution with $\mu = \ln \bar{x}_g$ specified in I and $\sigma = \ln \varepsilon$ in J. See \bar{x}_g and ε below and pp. 220ff for more.
LgNrm ⁻¹		LgNrm ⁻¹ returns x for a given probability p in X, with μ in I and σ in J.
LgNrm:	 	Submenu. See p. 125.
LinF	 	(0) Selects the linear fit model. Relevant for CORR, COV, L.R., s _{XY} , \hat{x} , and \hat{y} . See pp. 220ff 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 Y and the count of bit-shifts necessary to left justify the word in X. Example for word size 8: 1 0110 ₂ LJ returns x = 3 and y = 1011 0000 ₂
L _m	 	(2) {2}; {1} → {2}
L _{mα}	etc.	Laguerre polynomials and Laguerre's generalized polynomials. See pp. 235f 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. 87).
$\text{LN}\Gamma$		(1) {2, 3}; {1} → {2}
		Returns the natural logarithm of $\Gamma(x)$ (see p. 87). Allows also for calculating really great factorials (see an example in the <i>OM</i>).
$\text{LN}(1+x)$		(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> . LOAD calls LOADP , LOADR , LOADV , LOADΣ , LOADSS , recalls the lettered <i>registers</i> (incl. the <i>stack!</i>), and returns Backup restored . ¹⁴
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). ¹⁴
LOADR		(0) Recalls the numbered <i>GP registers</i> from backup (incl. the <i>local registers</i> allocated). Lettered <i>registers</i> will not be recalled. ¹⁴ The number of registers copied is the minimum of the registers held in the backup and allocated in <i>RAM</i> at execution time.
LOADSS		(0) Recovers the system state from backup, ¹⁴ meaning the entire calculator <i>configuration</i> (user assignments, system variables and <i>flags</i> as covered by RESET – see p. 64) plus all global user <i>flags</i> .

¹⁴ **LOAD** and **LOADP** are not programmable. See **SAVE** on p. 73 and App. B (pp. 166ff).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
LOADV	f [I/O] LOADV	(0) Recalls the user-defined variables from backup. ¹⁴
LOADΣ	f [I/O] LOADΣ	(0) Recovers the statistical summation <i>registers</i> from backup. Throws an error if there are none. ¹⁴
LocR	g [P.FN] g LocR n	(0) Allocates <i>n local registers</i> (≤ 99) and 16 <i>local flags</i> for the <i>current routine</i> . Cf. the OM, Sect. 3.
LocR?	g [INFO] f LocR?	(-1) {} → {1} Returns the number of <i>local registers</i> currently allocated.
LOG ₁₀	g lg	(1) {1, 2, 3, 8*, 9*, 10} ({1} → {2}) Returns the logarithm of <i>x</i> for base 10.
LOG ₂	g EXP lb x	(1) {1, 2, 3, 8*, 9*, 10} ({1} → {2}) Returns the logarithm of <i>x</i> for base 2.
LogF	f [STAT] ▼ LogF	(0) Selects the logarithmic curve fit model. Relevant for CORR, COV, L.R., s _{XY} , \hat{x} , and \hat{y} . See pp. 220ff for more.
Logis _p	g [PROB]	(1) {2} <i>Logistic distribution with μ given in I and s in J.</i> See pp. 220ff for details.
Logis _μ	f Logis: Logis _p etc.	
Logis _Δ		
Logis ⁻¹		
Logis:	g [PROB] f Logis:	Submenu. See p. 125.
LOG _{xy}	g EXP log _{xy}	(2) {1, 2, 3, 8*, 9*, 10}; ({1} → {2}) Returns the logarithm of <i>y</i> for the base <i>x</i> .
LOOP	f LOOP	Menu. See p. 124.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
ly→m	f [U→] x: ly→m	(1) {2}; {1} → {2} Converts distances. See pp. 140ff.
L.INTS	f [CATALOG] VARS L.INTS	Submenu of <i>long integer</i> variables defined at execution time. See pp. 116ff.
L.R.	f [STAT] ▲ L.R.	(-2) or (-3) {} → {2} Pushes the parameters a₂ (in Z), a₁ (in Y), and a₀ (in X) of the fit curve through the data points accumulated in the statistical summation <i>registers</i> on the stack, according to the curve fit model selected (see LINF, ORTHOF, EXPF, POWERF, LOGF, HYPF, ROOTF, PARABF, CAUCHF, GAUSSF). For a straight line, a₀ is its y-intercept and a₁ is its slope. See pp. 220ff for more.
m²→ac	f [U→] f A: m² → acre	(1) {2}; {1} → {2} Convert areas. See pp. 138ff.
m²→ac_{US}	f [U→] f A: m² → acre_{US}	
m²→ha	f [U→] f A: m² → ha	
m³→bbl	f [U→] f V: f m³ → barrel	(1) {2}; {1} → {2} Convert volumes. See pp. 138ff.
m³→fz_{UK}	f [U→] f V: f m³ → floz_{UK} etc.	
m³→fz_{US}	f [U→] f V: f m³ → qt. etc.	
m³→gl_{UK}	f [U→] f V: m³ → gl_{UK} etc.	
MANT	f [PARTS] MANT	(1) {2}; {1} → {2} Returns the mantissa m of the number $x = m \cdot 10^h$ displayed. Cf. EXPT.

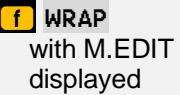
Item	Keystrokes	Remarks (see pp. 13ff for general information)
MASKL	f BITS MASKL <i>n</i>	(-1) {} → {10} Work like MASKL and MASKR on HP-16C, but with the mask length (or its address) following the command instead of being taken from X. Thus, the mask is pushed on the stack. MASKL 0 and MASKR 0 return 0.
MASKR	f BITS MASKR <i>n</i>	Example: For WSIZE 8, MASKL 3 will return a mask word 1110 0000 ₂ . 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. 116f.
MATR?	g TEST ▲ MATR?	(0) Checks if x is a real or complex matrix.
MATX	f MATX	Menu. See p. 124.
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 x and y.
MEM?	g INFO MEM?	(-1) {} → {1} Returns the number of free bytes 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. 116ff.
min	g X.FN g min	(2) {1, 2, 4, 5, 6, 7, 10} Returns the minimum of x and y.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
MIRROR	 	(1) {10} Reflects the bit pattern in x (e.g. 0001 0111 ₂ would become 1110 1000 ₂ for word size 8).
mi. \rightarrow m	 	(1) {2}; {1} \rightarrow {2}
mmH \rightarrow Pa	 	Convert distances and pressures. See pp. 138ff.
MOD	 	(2) {1, 2, 10} Returns $y \bmod x$ (modulo, see Section 2 of the OM for examples). Cf. RMD.
MODE		Menu. See p. 124.
MONTH	 	(1) {2, 6} \rightarrow {1} Assumes x containing a date in the format selected (or a real number in corresponding format) and extracts the month.
MSG	 	(0) {1, 2} Throws the (temporary) error message specified by the integer part of x . Cf. ERR. See App. C on pp. 176ff for the respective error codes.
MUL π	 	(0) Sets the ADM to multiples of π . Indicated in the status bar.
MUL π \rightarrow	 	(1) {1, 2} \rightarrow {4} Converts angles as described on pp. 147f.
MVAR	 	(0) Defines a menu variable. Such variables are required for VARMNU. Works in PEM only.
MyMenu	 	User menu. See the OM, Section 6.
My α	 	User menu in AIM.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
M.DELR	f DELR with M.EDIT displayed	(0) {8, 9} Deletes the current row of elements (where the cursor is in). Will not work if the matrix has only one row.
M.DIM	f MATX f DIM name	(0) {1, 2} Creates a new named matrix or re-dimensions an existing matrix to IP(y) rows and IP(x) columns. See DIM in the <i>HP-42S Owner's Manual</i> , p. 217.
M.DIM?	g INFO g DIM?	(8, 9) → {1}
	f MATX ▲ f DIM?	Returns the dimensions of the matrix <i>x</i> (rows to Y , columns to X). Note the matrix is saved in L . Former <i>y</i> goes into Z , former <i>z</i> into T , etc.
M.DY	g CLK ▲ M.DY	(0) Selects the format mm/dd/yyyy for dates.
M.EDI	f MATX EDIT	(2) {8, 9} Opens <i>x</i> using the <i>Matrix Editor</i> (like MATRIX EDIT in HP-42S). ¹⁵ 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 HP-42S). ¹⁵ See Section 2 of the OM.
M.EDIT		Submenu for matrix editing, called by M.EDI and M.EDIN. See p. 124.
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 HP-42S). Cf. M.PUT.

¹⁵ 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. *xz*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.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 <i>Section 2</i> of the OM; see also GROW in the <i>HP-42S Owner's Manual</i> , p. 213.). Cf. M.WRAP.
M.INSR	f INSR	(0) Inserts a new row of elements containing zero, left of the current cursor position in the matrix.
M.LU	f MATX ▲ f M.LU	(1) WP 34S: Takes a <i>descriptor</i> of a square matrix in X . Transforms (<i>X</i>) into its LU decomposition in-situ. The value in X is replaced by a <i>descriptor</i> that defines the pivots that were required to calculate the decomposition. The most significant digit is the pivot for the first diagonal entry, the next most significant for the second and so forth.
M.NEW	f MATX NEW	(2) {1, 2} → {8} Creates a new matrix (like NEW in HP-42S). Its number of rows shall be supplied in Y and its number of columns in X . M.NEW returns a matrix <i>x</i> with all its elements set to zero.
M.OLD	OLD with M.EDIT displayed	(0) Recalls the old element content (like OLD in HP-42S). See the OM, Sect. 2.
M.PUT	f MATX g PUTM	(0) {8, 9} Puts the matrix <i>x</i> as is into the indexed matrix (like PUTM in HP-42S). Cf. M.GET.
M.R>R	f MATX ▲ f R>R	(0) {8, 9} Swaps row <i>x</i> and row <i>y</i> of the indexed matrix (like R>>R in HP-42S).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
M.SIMQ		_submenu of <u>MATX</u> , called by SIM_EQ.
M.SQR?		(0) Returns true if x is a square matrix.
M.WRAP		(0) Controls the index pointers (see <i>Section 2</i> of the OM). Cf. M.GROW.
m:		<u>Submenu</u> . See p. 138.
m→au		(1) {2}; {1} → {2} Convert distances or heights. See pp. 138ff.
m→fm.		
m→ft.		
m→ft _{us}		
m→in.		
m→ly		
m→mi.		
m→nmi.		
m→pc		
m→pt.		
m→yd.		
NAND		(2) Works in analogy to AND. See p. 19.
NaN?		(0) Returns true if x is Not a Number.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
NBin _p	g [PROB] g NBin: NBin _p etc.	(1) {2} Negative binomial distribution with the total number of failures f in X, the probability of a success p_0 in I, and the number of draws n in J. See pp. 220ff for more information.
NBin ₋₁		
NBin ₋₁		
NBin ⁻¹		
NBin:	g [PROB] g NBin:	Submenu. See p. 125.
NEIGHB	g [INFO] f NEIGHB	(2) {1} Returns ... <ul style="list-style-type: none">• $x + 1$ for $x < y$;• x for $x = y$;• $x - 1$ for $x > y$. (2) {2} Returns the nearest machine-representable number to x in the direction towards y in the mode set. For <ul style="list-style-type: none">• ... $x < y$, it is the machine successor of x ;• ... $x = y$, it is y ;• ... $x > y$, it is the machine predecessor of x. NEIGHB may be useful investigating numeric stability (see NEIGHBOR in the HP-71 Math Pac).
NEXTP		(1) {1, 10}; {2} → {1} Returns the next prime number greater than IP(x) . See also PRIME? on p. 60.
nmi.→m	f U→ x: f nmi.→m	(1) {2}; {1} → {2}
Nm→lbft	f U→ Nm → lbf·ft	Convert distances and torques. See pp. 138ff.
NOP	g [P.FN] P.FN2 f NOP	'Empty' step (for historical reasons only).
NOR	f BITS f NOR	(2) Works in analogy to AND. See p. 19.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
Norml _p	g PROB Norml: Norml _p etc.	(1) {2} Normal distribution with an arbitrary mean μ given in I and a standard deviation σ in J. See Sect. 2 of the OM for an application example and pp. 220ff for more.
Norml _a		Norml ⁻¹ returns x for a given probability p in X, with μ in I and σ in J.
Norml ⁻¹		
Norml:		Submenu. See p. 125.
NOT	f BITS NOT	(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	g Σ n	(-1) {} → {1} Recalls the number of accumulated data points.
N→lbf	f U F&p; N→lbf	(1) {2}; {1} → {2} Converts forces. See pp. 138ff.
ODD?	g TEST f ODD?	(0) Checks if x is integer and odd.
OFF	g OFF	(0) Turns your WP 43S off. In PEM, inserts a step to turn it off under program control.
OR	f BITS OR	(2) Works in analogy to AND. See p. 19.
OrthoF	f STAT g OrthoF	(0) Selects the linear orthogonal fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 229ff for more.
ORTHOG		
oz→kg	f U m: oz→kg	(1) {2}; {1} → {2} Converts masses. See pp. 138ff.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
ParabF	f [STAT] ▼ f ParabF	(0) Selects the parabolic fit model. Relevant for COV, CORR, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 229ff for more.
PARTS	f PARTS	Menu. See p. 124.
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	
Pa→bar	f U→ F&p: Pa→bar	
Pa→iHg	f U→ F&p: f Pa → in.Hg	(1) {2}; {1} → {2}
Pa→mmH	f U→ F&p: f Pa → mmHg	Convert pressures. See pp. 140ff.
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. 140ff.
PERM	g PROB P _{yx}	(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. 218f for the formula.
PGMINT	f ADV f PGMINT lab!	Specifies the address of the expression to be integrated or solved, respectively. See Section 4 of the OM.
PGMSLV	f ADV f PGMSLV lab!	

Item	Keystrokes	Remarks (see pp. 13ff for general information)
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. 18 for more.
PLOT	f STAT g PLOT	(0) Plots the n data points given by the $nx2$ matrix x . See p. B-20 for more.
P_n	g X.FN Orthog P_n	(1) {2}; {1} → {2} <i>Legendre polynomials.</i> See pp. 235f for more.
POINT	g P.FN P.FN2 g POINT	(1) {1, 2} Turns on a square point (3x3 px ■) on the screen. The position of its center is given by the integer parts of the numbers in X and Y . See AGRAPH on p. 18 for more.
Poiss_p	g PROB	(1) {2}; {1} → {2}
Poiss_λ	g Poiss: Poiss _p etc.	<i>Poisson distribution</i> with the number of successes g in X and the <i>Poisson parameter λ</i> in I . See pp. 220ff for details.
Poiss_Δ		Poiss ⁻¹ returns the maximum number of successes m for a given probability p in X and λ in I .
Poiss:	g PROB g Poiss:	<i>Submenu.</i> See p. 125.
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_{XY} , \hat{x} , and \hat{y} (see pp. 220ff for more).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
PRCL	g P.FN f PRCL	(0) Copies the <i>current program</i> (from <i>FM</i> or <i>RAM</i>) and appends it to <i>RAM</i> , where it can then be edited (see the OM). PRCL allows for duplicating programs in <i>RAM</i> . Will only work with enough space at destination. Recall a library routine from <i>FM</i> , edit it, and PSTO – this way you can modify this part of the <i>FM</i> library (see PSTO).
PRIME?	g TEST f PRIME?	(0) {1, 2, 10} Checks if $ IP(x) $ is a prime. Returns true for prime and false for composite. For $x > 3.3 \times 10^{24}$, true means ‘probably prime’ with infinitesimal probability for being composite (see p. 169 for more).
PRINT	g PRINT	Menus. See p. 125.
PROB	g PROB	
PROG		Submenu of global labels defined when calling XEQ etc. See Section 3 of the OM.
PROGS	f CAT. PROGS	Submenu of global labels defined at execution time. See pp. 116f.
pr→dB	f U→ ▲ power ratio → dB	(1) {2}; {1} → {2}
psi→Pa	f U→ F&p: psi→Pa	Convert ratios or pressures. See pp. 138ff.
PSTO	g P.FN f PSTO	(0) Copies the <i>current program</i> (see the OM) from <i>RAM</i> and appends it to the <i>FM</i> library. Cf. PRCL. This program must include at least one LBL statement with a global label (preferably at its beginning). If a program with the same label already exists in the library it will be deleted first. Global labels may be browsed in CATALOG PROGS and called by XEQ .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
pt.→m	f [U→] x: g point → m	(1) {2}; {1} → {2} Converts print heights. See pp. 138ff.
PUTK	g [P.FN] f PUTK r	(0) Assumes a key code in the address specified. Stops program execution, takes said code and puts it in the keyboard buffer resulting in immediate execution of the corresponding call. R/S is required to resume program execution then. May help in user interaction with routines (see the OM, Section 3).
P.FN	g [P.FN]	Menu. See p. 125.
P.FN2	g [P.FN] P.FN2	Submenu. See p. 125.
P:	f [U→] P:	Submenu. See p. 138.
qt.→m³	f [U→] f x: qt.→m³	(1) {2}; {1} → {2} Converts volumes. See pp. 140ff.
RAD	g MODE RAD	(0) Sets the <i>ADM</i> to <i>radians</i> . Indicated in the status bar.
RAD→	g [L→] f RAD→	(1) {1, 2} → {4} Converts angles as described on pp. 147f.
RAM	f [CAT.] PROGS RAM	Submenu of global labels defined at execu- tion time. See pp. 116f.
RANGE	g [DISP] ▲ f RANGE	(0) {1, 2} Limits the range of displayable real numbers to $\pm 10^{\pm n}$ with $n = \text{IP}(x)$. $99 \leq n \leq 6145$. For greater input, n will be set to 6145 (startup default); for less it will be set to 99. RANGE allows for saving screen space for reasonable data.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
RANGE?	g DISP ▲ f RANGE?	(-1) {} → {1}
	g INFO ▲ RANGE?	Returns the current range setting.
RANI#	g PROB ▲ RANI#	(-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#	g PROB ▲ RAN#	(-1) {} → {2} Returns a random number between 0 and 1 like RAN does in HP-42S. See also SEED.
RBR	f RBR	Calls the <i>register browser</i> – 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	RCL <i>r</i>	(-1) Recalls the content of a <i>register</i> or variable.
RCLCFG	RCL f Config <i>r</i>	(0) Recalls a calculator <i>configuration</i> stored by STO CFG (see the OM, Sect. 2 and 6). Cf. also RESET on p. 64.
RCLEL	f MATX g RCLEL	(-1) {} → {2, 3}
	RCL g ...EL	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	f MATX ▲ 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.
	[RCL] g ...IJ	
RCLS	[RCL] f Stack r	Recalls 4 or 8 values from a set of registers starting at address r, and pushes them on the stack. This is the converse command of STOS.
RCL+	[RCL] + r	(1) Recalls a content of a register or variable, executes the operation specified, and puts the result on the stack like a monadic function. ¹⁶
RCL-	[RCL] - r	
RCLx	[RCL] x r	
RCL /	[RCL] / r	
RCL↑	[RCL] f Max r	(1) {1, 2, 4, 5, 6, 10}
	[RCL] ▲ r	Replaces x with the maximum of r and x. ¹⁶
RCL↓	[RCL] f Min r	(1) {1, 2, 4, 5, 6, 10}
	[RCL] ▼ r	Replaces x with the minimum of r and x. ¹⁶
RDP	g DISP f RDP n	(1) {2, 3, 5, 8*, 9*} Rounds x to n decimal places ($0 \leq n \leq 99$, think of FIX format), taking the RM setting into account. See RM and compare RSD.
Re	g CPX Re	(1) {2, 3} → {2}; {9} → {8}
	f PARTS g Re	Returns the real part of x. Cf. IM.
REALS	f CAT. VARS REALS	Submenu of real/variables defined at execution time. See pp. 116f.

¹⁶ 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)
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 <i>register names</i> defined. See pp. 116f.
RESET	g CLR g RESET	Executes CLALL and resets your WP 43S to <i>startup default configuration</i> , i.e. 2COMPL, ALL 0, DEG, DENMAX 0, DSTACK 4, GAP 3, J/G 1752-01-01, LinF, LocR 0, RM 0, RSD 34, TDISP -1, WSIZE 64, and Y.MD. In this course, RESET also sets RANGE to 6145 and resets the random number generator. It sets the <i>flags</i> ASLIFT, DECIM., DENANY, MULTX, PROPFR, and TDM24; it clears all other <i>system flags</i> (see pp. 104ff). See said commands and <i>system flags</i> 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">• (for L) the <i>real</i> part from Y and <i>imaginary</i> part from X, or• (for ⊕) <i>magnitude</i> from Y and <i>phase</i> from X.
	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	   RJ	<p>(10)</p> <p>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.</p> <p>Example: 10 1100₂ RJ results in $y = 1011_2$ and $x = 2$.</p>
RL	  RL 	<p>(1) {10} </p> <p>Works like n consecutive RLs on HP-16C, similar to RLn there ($0 \leq n \leq 63$). RL 0 executes as NOP, but loads L. See the OM, Sect. 2, for more.</p>
RLC	  RLC 	<p>(1) {10} </p> <p>Works like n consecutive RLCs on HP-16C, similar to RLCn there ($0 \leq n \leq 64$). RLC 0 acts as NOP, but loads L. See the OM, Sect. 2, for more.</p>
RM	  RM 	<p>(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 <u>not</u> alter the display nor change the behavior of ROUND. The following 7 modes are supported:</p> <ul style="list-style-type: none"> 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 ('business-man's rounding' ¹⁷). 2: round half down: $\frac{1}{2}\downarrow$ 0.5 rounds down. 3: round up: $\leftarrow\rightarrow$ rounds away from 0. 4: round down: $\rightarrow\leftarrow$ rounds towards 0.

¹⁷ 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)
		5: ceiling: $\lceil x \rceil$ rounds towards $+\infty$. 6: floor: $\lfloor x \rfloor$ rounds towards $-\infty$. The abbreviations printed on grey background are used in STATUS for indicating the respective rounding modes (see Section 5 of the OM).
RMD	f RMD	(2) {1, 2, 10}
	g INTS f RMD	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.
RM?	g INFO RM?	(-1) {} → {1}
		Returns the floating point rounding mode set. See RM for more.
RNORM	f MATX ▲ f 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	f STAT ▼ f RootF	(0) Selects the root fit model. Relevant for CORR, COV, L.R., s_{XY} , \hat{x} , and \hat{y} . See pp. 229ff for more.
ROUND	g DISP ROUND	(1) {2, 3, 4, 5, 6, 8*, 9*} Rounds x using the current display format like RND on HP-42S.
ROUNDI	g DISP ROUNDI	(1) {8*}; {2} → {1} Rounds x to next integer. $\frac{1}{2}$ rounds to 1. Cf. IP.
RR	f BITS ▲ RR n	(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.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
RRC	 	(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		(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	 	(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		(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{t} is lit. If pressed in run mode with no routine executing, RTN 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{t} is lit.
RTN+1	 	(0) Works like RTN, but moves the program pointer two 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. Example: For $x = 34.567$, R-CLR will clear R34 through R89.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
		<p>ATTENTION: For $nn = 0$, clearing will cover the maximum available:</p> <ul style="list-style-type: none"> • For $sss \in [0; 99]$, it will stop at R99. • For $sss \in [100; 111]$, it will stop at K. • For $sss \geq 112$, it will stop at the highest currently allocated local register.
R-COPY	 	<p>(0) {2}</p> <p>Interprets x in the form $sss.nnnnn$. Takes nn registers starting with address sss and copies their contents to ddd etc.</p> <p>Example: For $x = 7.0304567$, $r07$, $r08$, $r09$ will be copied into R45, R46, R47, respectively.</p> <p>For $x < 0$, R-COPY will take nn registers from FM instead, starting with register number sss. Destination will be in RAM always.</p> <p>ATTENTION: For $nn = 0$, copying will cover the maximum available as explained with R-CLR. Then x must be negative.</p>
R-SORT	 	<p>(0) {2}</p> <p>Interprets x in the form $sss.nn$. Sorts the contents of nn registers starting with address sss.</p> <p>Example: Assume $x = 49.036\ 9$, $r49 = 1.2$, $r50 = -3.4$, and $r51 = 0$; then R-SORT will return $r49 = -3.4$, $r50 = 0$, and $r51 = 1.2$.</p> <p>ATTENTION: For $nn = 0$, sorting will cover the maximum available as explained with R-CLR.</p>
R-SWAP	 	<p>(0) {2}</p> <p>Works like R-COPY but <u>swaps</u> the contents of source and destination registers.</p>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
R→D	[g] [L↔] [g] R→D	(1) {1, 2} → {4} Converts angles as described on pp. 147f.
R↑	[f] [R↑]	Rotates the stack contents one level up or down,
R↓	[R↓]	respectively. See Section 1 of the OM for details.
s	[f] [STAT] s	(-2) {} → {2} Takes the statistical sums accumulated, calculates the <i>sample standard deviations</i> s_y and s_x and pushes them on the stack. See Sect. 2 of the OM for the output format and pp. 220ff for the formula.
SAVE	[g] [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 .
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. 138ff.
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 dividing (multiplying) x by 10^n . Works for decimal numbers in a way like SL and SR work for binary integers.
SDR		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
SEED	g PROB ▲ SEED	(0) {2} Stores a seed for random number generation. For $x \leq 0$, the seed is taken from the real-time clock.
SEND	f I/O f SEND	(0) Sends all RAM data to the device connected via serial I/O. See RECV and the OM, Sect. 3 for more.
SETCHN	g DISP ▲ CHINA	(0) Sets regional format preferences. ¹⁸
SETDAT	g CLK ▲ SETDAT	(0) Sets the date for the real-time clock (the emulator takes this information from the PC clock).
SETEUR	g DISP ▲ EUROPE	
SETIND	g DISP ▲ INDIA	
SETJPN	g DISP ▲ JAPAN	
SETSIG	g MODE f SETSIG	(0) {1} Sets the number of significant digits (0 ... 34) for rounding after each operation. SETSIG 0 sets maximum precision.
SETTIM	g CLK ▲ SETTIM	(0) Sets the time for the real-time clock (the simulator takes this information from the PC clock).
SETUK	g DISP ▲ UK	
SETUSA	etc.	(0) Set regional format preferences. ¹⁸
SF	g FLAGS SF <i>n</i>	(0) Sets the flag specified.
	g MODE SF <i>n</i>	

¹⁸ See Section 2 of the OM about localization of numeric output.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
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 $\text{signum}(x)$. (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 $(MATA) \cdot \overrightarrow{MATX} = \overrightarrow{MATB}$. 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	TRI f sinc	(1) {2, 3, 8*, 9*} ... for $x \neq 0$ and 1 for $x = 0$. Tagged input of <i>data type</i> 4 will be converted in <i>radians</i> before computing. Untagged input is taken as <i>radians</i> . Returns $\frac{\sin(x)}{x}$...
sincπ	TRI f sincπ	(1) {2, 3, 8*, 9*} Returns $\frac{\sin(\pi x)}{\pi x}$...

Item	Keystrokes	Remarks (see pp. 13ff for general information)
sinh	g EXP g sinh	(1) {2, 3, 8*, 9*}
	TRI g sinh	Returns the hyperbolic sine of x .
SKIP	g P.FN P.FN2 g SKIP n	(0) Skips n program steps forwards ($0 \leq n \leq 255$). So e.g. SKIP 2 skips over the next two steps, going e.g. from step 123 to step 126. If SKIP attempts to cross an END, an error is thrown. ATTENTION: If you edit a section of your routine crossed by one or more BACK, SKIP, or CASE jumps, this may well result in a need to manually maintain all those statements individually .
SL	f BITS SL n	(1) {10}  Works like n (≤ 63) consecutive SLs on HP-16C. SL 0 executes as NOP, but loads L. See Sect. 2 of the OM for more.
SLVQ	f ADV SLVQ	{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">• If $r := b^2 - 4ac \geq 0$, SLVQ returns $-\frac{b \pm \sqrt{r}}{2a}$ in Y and X. In a routine, the step after SLVQ will be executed.• 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. 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	f [STAT] 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_{mw}	f [STAT] f 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	g [SNAP]	(0) Stores a snapshot of the screen (a.k.a. screenshot) in a BMP file on the calculator's USB flash disk in the /SCREENS directory. The file name comprises date and time of storage. The file can be dealt with on your computer.
SOLVE	f [ADV] SOLVE var	{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 X and Y when calling SOLVE. It returns var_{root} in X, the second last var -value tested in Y, then $f(var_{root})$ in Z, and 0 in T. Additionally, SOLVE acts as 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. ATTENTION: SOLVE fills all <i>stack registers</i> with x before calling the routine specified.
Solver	g [EQN] Solver	Submenu for solving a given equation. See the OM, Sect. 4, for more.
SPEC?	g [TEST] ▲ f SPEC?	(0) True if x is 'special' (i.e. $±\infty$ or NaN).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
SR	f BITS ▲ SR n	(1) {10} 0 → [] → C Works like n (≤ 63) consecutive SRs on HP-16C. SR 0 executes as NOP, but loads L. See Section 2 of the OM for more.
SSIZE?	g INFO SSIZE?	(-1) {} → {1} Returns the number of <i>stack registers</i> currently allocated, 4 or 8.
STAT	f STAT	Menu. See p. 126.
STATUS	f STATUS g FLAGS STATUS	Flag browser. See Section 5 of the OM.
STK	g STK	Menu. See p. 126.
STO	STO r	(0) Stores x into destination.
STOCFG	STO f Config r	(0) Stores the current calculator <i>configuration</i> in a variable or <i>register</i> for later use as described in Sect. 2 and 6 of the OM. RCLCFG recalls such data. Cf. also RESET on p. 64.
STOEL	f MATX g STOEL STO g ...EL	(1) {1, 2, 3} Stores a copy of x into the indexed matrix at the current element, a_{ij} . Cf. RCLEL.
STOIJ	f MATX ▲ STOIJ STO g ...IJ	(1) {1} Sets the index pointers to IP(x) (column number) and IP(y) (row number). Cf. RCLIJ.
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 <i>stack</i> in a set of 4 or 8 <i>registers</i> , starting at the destination address specified. See RCLS.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
STO+	STO + r	(0) Executes the specified operation on r and stores the result (e.g. $r - x$) at the address specified. ¹⁹
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. 138ff.
STO↑	STO f Max r	(0) {1, 2, 4, 5, 6, 10}
	STO ▲ r	
STO↓	STO f Min r	
	STO ▼ r	
STRI?	g TEST g STRI?	(0) True if x is a <i>text string</i> (like STR? in HP-42S).
STRING	f CATALOG VARS STRING	Submenu of <i>text string</i> variables defined at execution time. See pp. 116f.
SUM	f STAT SUM	(-2) {} → {2} Recalls the linear sums Σy and Σx . Useful in basic 2D vector algebra. Output is labeled in analogy to s .
s_w	f STAT f s_w	(-1) {} → {2} Calculates the <i>standard deviation</i> for weighted data (where the weight y of each data point x was entered via [Σ+]). See pp. 220ff for the formula.

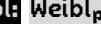
¹⁹ Only legal operations according to the matrices in Section 2 of the OM will work. See also the examples given there.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
s_{xy}	f STAT ▲ s_{xy}	(-1) { } → {2} Calculates the <i>sample covariance</i> for the two data sets entered via SUM , depending on the curve fit model selected. See pp. 220ff for the formula and COV for the <i>population covariance</i> .
SYSTEM	g MODE g SYSTEM	(0) Returns to DMCP. Works on the calculator only (not on the simulator). See App F.
SYS.FL	f CATALOG SYS.FL	Submenu of system flags. See p. 116.
s(a)	f STAT ▲ f s(a)	(-2) { } → {2} Pushes the standard errors s(a₁) (in Y), and s(a₀) (in X) 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. 235f for more.
S.INTS	f CAT. VARS S.INTS	Submenu of short integer variables defined at execution time. See pp. 116f.
s.t→kg	f U→ m: ▲ short ton → t	(1) {2}; {1} → {2} Convert masses and times. See pp. 138ff.
s→year	f U→ f s→year	
tan	TRI tan	(1) {2, 3, 8*, 9*}; {1, 4} → {2} Returns the tangent of the angle in X. Returns “Not a Number” for $x = \pm 90^\circ$ or equivalents if SPCRES is set.
tanh	g EXP g tanh	(1) {2, 3, 8*, 9*}
	TRI g tanh	Returns the hyperbolic tangent of x.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
TDISP	g CLK ▲ TDISP <i>n</i>	(0) Sets time display format. TDISP 0 and 1 allow for displaying just <i>hours</i> and <i>minutes</i> , TDISP 2 for <i>seconds</i> , too, and <i>n</i> ≥ 3 also for <i>n</i> – 2 digits showing decimal fractions of <i>seconds</i> . TDISP –1 allows for displaying all digits.
TEST	g TEST	Menu. See p. 126.
TICKS	g P.FN TICKS	(–1) {} → {1} Returns the number of ticks from the real-time clock at execution time. 1 tick = 0.1 s. Counting starts when the calculator is turned on.
TIME	g CLK TIME	(–1) {} → {5} Recalls the time from the real-time clock at execution (see Sect. 2 of the OM for the output format).
TIMER	f TIMER	Starts the timer application based on the real-time clock and following the timer of HP-55. See the OM, Sect. 5 for a detailed description.
TIMES	f CAT. VARS f TIMES	Submenu of <i>time</i> variables defined at execution time. See pp. 116f.
T_n	g X.FN Orthog T_n	(2) {2}; {1} → {2} <i>Chebyshev polynomials of first kind.</i> See pp. 235f for details.
TONE	f I/O f TONE <i>n</i>	(0) Sounds a tone according to <i>n</i> (= 1 ... 9).
ton→kg	f U→ m: ▲ ton→kg	(1) {2}; {1} → {2} Converts masses. See pp. 138ff.
TOP?	g TEST g TOP?	(0) Returns ... <ul style="list-style-type: none"> • false if called with the program pointer being in a subroutine; • true if called in the top routine (i.e. if the program-running flag is set and the SRS pointer is clear).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
tor→Pa	f [U→] F&p: f torr → Pa	(1) {2}; {1} → {2} Converts pressures. See pp. 138ff.
t_p(x)	g PROB t: t_p(x) etc.	(1) {2}
t_Δ(x)		<i>Student's t distribution.</i> The degrees of freedom are stored in J. t _e (x) equals Q(t) on HP-21S. See Section 2 of the OM for an application example and pp. 220ff for more mathematical details.
t_Δ(x)		
t⁻¹(p)		
TRANS	f [CAT.] FCNS TRANS	Works like [M] ^T on p. 94. Maintained for backward compatibility only.
TRI	TRI	Menu. See p. 126.
trz→kg	f [U→] m: f tr.oz → kg	(1) {2}; {1} → {2} Converts masses. See pp. 138ff.
TVM	g FIN TVM	Submenu for the application <i>Time Value of Money</i> . See Section 5 of the OM.
t:	g PROB t:	Submenu. See p. 125.
t↔	g STK t↔ r	Swaps <i>t</i> and <i>r</i> , in analogy to <i>x↔</i>
ULP?	g INFO f ULP?	(1) {1, 2} Returns 1 times the smallest power of ten which can be added to <i>x</i> or subtracted from <i>x</i> to actually change the (internal) value of <i>x</i> in your WP 43S in the mode set. Thus, 1 is returned for integers. Indicated in STATUS.
U_n	g X.FN Orthog U_n	(2) {2}; {1} → {2} <i>Chebyshev polynomials of second kind.</i> See pp. 235f for details.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
UNDO	f	Recalls the stack, the summation registers, and the system flags as they were before the last operation executed.
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 complex 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 Section 2 of the OM.
	g INTS ...	
U→	f U→	Menu. See p. 126.
VAR		Submenu of variables defined at execution time when calling STO, RCL, etc.
VARMNU	g P.FN f VARMNU labl	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. 116f.
VERS?	g INFO g VERS?	(0) Shows your firmware version and build number until next keystroke (see the OM, Sect. 2).

Item	Keystrokes	Remarks (see pp. 13ff for general information)
VIEW		(0) Shows <i>r</i> until the next key is pressed. Example: If a variable called Test12 contains -123.45 , VIEW @ Test12 ENTER↑ will display Test12 = -123.45
V:		Submenu. See p. 138.
V ₄		(2) {8} → {4} Returns the angle between two 2D or 3D vectors $\vartheta = \arccos\left(\frac{\vec{v}_1 \cdot \vec{v}_2}{ \vec{v}_1 \vec{v}_2 }\right)$
WDAY		(1) {2, 6} → {1} Assumes <i>x</i> 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). ²⁰
Weibl _p	  Weibl:  etc.	(1) {2} Weibull distribution with its shape parameter b in I and its characteristic lifetime T in J . See pp. 220ff for details.
Weibl _▲		Weibl ⁻¹ returns the survival time t_s for a given probability p in X , with b in I and T in J .
Weibl ⁻¹		
Weibl:		Submenu. See p. 125.
WHO?		(0) Displays credits to the brave men who made this project work (temporary message).
Wh→J		(1) {2}; {1} → {2} Converts energies. See pp. 138ff.

²⁰ 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)
W_m	 etc.	(1) {2, 3}; {1} → {2}
W_p		W_p returns the principal branch of <i>Lambert's W</i> for given $x \geq -1/e$. W_m returns its negative branch (works for $x \in \mathbb{R}$ only). W^{-1} returns x for a given W_p (≥ -1). See pp. 251ff for more.
W^{-1}		
WSIZE	 	(0) Sets the <i>word size</i> (a.k.a. bit width) for <i>DT 10</i> . Works almost like on <i>HP-16C</i> , but with the parameter $1 \leq n \leq 64$ trailing the command instead of taken from X . WSIZE 0 sets the <i>word size</i> to maximum, i.e. 64 bits. The <i>word size</i> set is indicated in the <i>status bar</i> . WSIZE works as it does in <i>WP 34S</i> : <u>Reducing word size</u> truncates the values in L and the stack allocated. All other memory content stays as is (see <i>App. B</i> on pp. 160ff). <u>Increasing the word size</u> will add empty bits to L and each <i>stack register</i> allocated. All other memory content stays as is.
WSIZE?	 	(-1) {} → {1} Recalls the <i>word size</i> set. This size is also indicated in the <i>status bar</i> .
$W \rightarrow hp_E$	 	etc. (1) {2}; {1} → {2} Convert powers. See pp. 138ff.
$W \rightarrow hp_M$		
$W \rightarrow hp_{UK}$		
\hat{x}	 	(1) {2}; {1} → {2} Returns a forecast \hat{x} for a given y (in X) according to the curve fit model chosen. See L.R. for more.
\bar{x}	 	(-2) {} → {2} Calculates the <i>arithmetic means</i> of the y - and x -data accumulated and pushes them on the <i>stack</i> . See also s , s_m , and σ .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
x^2		(1) [1, 2, 3, 8*, 9*, 10]
x^3		Return the square or cube of x , respectively.
XEQ		(0) Executes the function or routine with the label specified. – In PEM, XEQ inserts a call to the subroutine with the label specified.
\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. 229ff for the formula. Output format will be similar to the one of \bar{x} . See also ε , ε_m , and ε_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 type = 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.
x_{\max}		(-2) {} → {2} Returns the maximum (or minimum) of the y- and x-data accumulated and pushes them on the stack.
x_{\min}		ATTENTION: These extrema will not be updated after Σ- (see there).
XNOR		(2) Work in analogy to AND. See p. 19.
XOR		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
xOUT	[XEQ] way	Cleans and reverts the settings of xIN, taking care of a proper return including the correct setting of I and the stack. Typically, way = xOUT NORMAL . Generally, xOUT shall be the last command of an XROM routine.
\bar{x}_{RMS}	f [STAT] ▲ g \bar{x}_{RMS}	(-2) {} → {2} Calculates the quadratic means of the y- and x-data accumulated and pushes them on the stack. Quadratic means are often used in electrical engineering for alternating currents.
\bar{x}_w	f [STAT] f \bar{x}_w	(-1) {} → {2} Returns the arithmetic mean for weighted data (where the weight y of each data point x was entered via $\Sigma+$). See pp. 229ff for the formula. See also s_w and s_{mw} .
$\sqrt[x]{y}$	g EXP $\sqrt[x]{y}$	(2) Returns the x^{th} root of y . For integers or reals < 0 , it will return complex numbers if CPXRES is set.
X.FN	g X.FN	Menu. See p. 127.
x!	f x!	(1) {1, 10} Returns the factorial $n!$. Note this is only defined for positive integers. $20!$ is the biggest factorial $< 2^{64}$. $450!$ is maximum allowed for long integers.
		(1) {2, 3} Returns $\Gamma(x + 1)$. $2\ 123.549\ 956\ 662\ 463\ 236\ 31$ is the maximum x for reals.
x:	f U→ x:	Submenu. See p. 138.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$x \rightarrow DATE$	g CLK $x \rightarrow DATE$	(1) {2} → {6} Interprets the <i>real</i> number 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.
$x \leftrightarrow r$	g STK $x \leftrightarrow r$	Swaps x and r , in analogy to $x \leftarrow y$. Will be listed like $x \leftrightarrow J$, $x \leftrightarrow .12$, $x \leftrightarrow \rightarrow 12$, etc. in programs.
$x \leftrightarrow y$	$x \leftrightarrow y$	Swaps the contents of stack registers X and Y .
$x = ?$	g TEST $x = ? r$	(0) etc.
$x \neq ?$		Compare x with r . See $x < ?$ for more.
$x \approx ?$	g TEST Δ $x \approx ? r$	(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.
$x < ?$	g TEST $x < ? r$	(0) {1, 2, 4, 5, 6, 7, 10}
$x \leq ?$	etc.	Compare x with r . <i>Text strings</i> are compared according to their sorting order.
$x \geq ?$		Example: TEST $x < ? K$ compares x with k , and will be listed as $x < ? K$ in a routine. It will return true if $x < k$ at execution time. See examples in Sect. 1 of the OM for more.
$x = +0?$	g TEST Δ $x = +0?$	(0) {2, 3, 10} etc.
$x = -0?$		These two 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 .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
\hat{y}		(1) {2}; {1} → {2} Returns a forecast y (in X) for a given x according to the curve fit model chosen. See L.R. for more.
$yd \rightarrow m$		(1) {2}; {1} → {2} Converts distances. See pp. 138ff.
YEAR		(1) {2, 6} → {1} Assumes x containing a date in the format selected (or a real number in corresponding format) and extracts the year.
year→s		(1) {2}; {1} → {2} Converts times. See pp. 138ff.
y^x		(2) Raises y to the power of x for compatible objects. ²¹
Y.MD		(0) Sets the format yyyy-mm-dd for dates.
$y \leftrightarrow r$		Swaps y and r , in analogy to $x \leftrightarrow$.
$z \leftrightarrow$		Swaps z and r , in analogy to $x \leftrightarrow$.
αINTL		Submenu. See pp. 116ff.
		Menu in AIM (see p. 128).
αLENG?		(-1) {} → {1} Returns the number of characters found in the text string r , similar to ALENG in HP-42S. ²²

²¹ See the tables in the OM, Section 2 for details and compatibility.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
αMATH	 	Submenu. See pp. 116ff. Menu in AIM. See p. 129.
$\alpha\text{POS?}$	 	(-1) {} → {1} Looks in the <i>text string r</i> for the target given in X . If a match is found, αPOS returns the position number where the target was found (counting the left-most character as position 0). If a match is not found, αPOS returns -1. ²²
	 	The target may be an individual character code or an <i>text string</i> . αPOS saves a copy of the target in L . It works similar to POSA in <i>HP-42S</i> .
αRL	 	(0) Rotates the <i>text string r</i> by x characters like AROT in <i>HP-42S</i> , but with $x \geq 0$. $\alpha\text{RL } 0$ executes as NOP, but loads L . ²²
αRR	 	(0) Works like αRL but rotates to the right.
αSL	 	(0) Shifts the x leftmost characters out of the <i>text string r</i> , like ASHF in <i>HP-42S</i> . This allows for deleting the first x characters in the string. $\alpha\text{SL } 0$ executes as NOP, but loads L . ²²
αSR	 	(0) Works like αSL but for the x rightmost characters out of <i>r</i> , deleting the last x characters in the string. ²²
αFN		Menu. See p. 129.
$\text{A...}\Omega$	 	Submenu of Greek letters, see pp. 116ff.
$\alpha\cdot$	 	Submenu. See pp. 116ff. Menu in AIM. See p. 129.

²² This command will throw an error if there is no *text string* or an empty string in *r* at execution time.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\alpha \rightarrow x$	g a.FN $\alpha \rightarrow x$ r	(-1) {} → {10 ₁₆ }
		Pushes the character code of the leftmost character in the <i>text string r</i> on the <i>stack</i> and removes this character from the string, similar to ATOX in HP-42S. ²²
$\beta(x,y)$	g X.FN ▲ $\beta(x,y)$	(2) {2, 3}; {1} → {2}
		Returns <i>Euler's Beta function</i> $B(x, y) = \frac{\Gamma(x) \Gamma(y)}{\Gamma(x+y)}$ with $Re(x) > 0$ and $Re(y) > 0$. Called β here to avoid ambiguity. See $\Gamma(x)$ below.
Γ_{xy}	g X.FN ▲ Γ_{xy}	(2) {2}; {1} → {2}
γ_{xy}	g X.FN ▲ γ_{xy}	Returns the <i>upper / lower incomplete Gamma function</i> . See pp. 251ff for more.
$\Gamma(x)$	g PROB ▲ $\Gamma(x)$	(1) {2, 3}; {1} → {2}
		Returns $\Gamma(x)$. Note x! calls $\Gamma(x + 1)$. See also $\text{LN}\Gamma$.
δx	f CATALOG PROGS δx	Predefined global label for $f'(x)$ and $f''(x)$ – see Section 4 of the OM.
$\Delta\%$	f △%	(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.
ε	f STAT g ε	(-2) {} → {2}
		Calculates the <i>scattering factors</i> ε_y and ε_x for <i>log-normally</i> distributed sample data and pushes them on the <i>stack</i> . ε 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. 220ff for more information.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
ε_m		(-2) { } → {2} Works like ε 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>).
ε_p		(-2) { } → {2} Works like ε but returns the <i>scattering factors</i> of the two populations.
$\zeta(x)$		(1) {2, 3} Returns <i>Riemann's Zeta</i> . See p. 255 for more.
π		(-1) { } → {2} Recalls π .
Π_n		Computes a product using the routine specified. See Section 4 of the OM for more. ATTENTION: Π_n fills all <i>stack registers</i> with x before calling the routine specified.
Σ		<i>Menu</i> . See p. 130.
σ		(-2) { } → {2} Works like s but returns the <i>standard deviations</i> of the two populations instead. See pp. 220ff.
Σ^1/x		(-1) { } → {2}
Σ^1/x^2		etc.
Σ^1/y		
Σ^1/y^2		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\Sigma \ln^2 x$	 etc.	ATTENTION: 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.
$\Sigma \ln^2 y$		
$\Sigma \ln x$		
$\Sigma \ln xy$		For space reasons, two sums are abbreviated: $\Sigma \ln xy$ denotes $\sum \ln(x) \ln(y)$ and
$\Sigma \ln y/x$		$\Sigma \ln y/x$ denotes $\sum \frac{\ln(y)}{x}$.
Σ_n		Computes a sum using the routine specified. See Section 4 of the OM for more. ATTENTION: Σ fills all stack registers with x before calling the routine specified.
s_w		(-1) {} → {2} Works like s_w but returns the standard deviation of the population instead. See pp. 220ff.
Σx	 etc.	
Σx^2		
$\Sigma x^2 y$		(-1) {} → {2}
$\Sigma x^2/y$		Recall the corresponding statistical sums, necessary for statistical analyses and regressions (see Σ^1/x above for more).
Σx^3	 etc.	
Σx^4		
$\Sigma x \ln y$		
$\Sigma x y$		
$\Sigma x/y$		

Item	Keystrokes	Remarks (see pp. 13ff for general information)
Σy	g [Σ] Σy etc.	
Σy^2		
$\Sigma y \ln x$	g [Σ] g Σylnx	
$\Sigma+^{23}$	f STAT Σ+	<p>[8] → {2}</p> <p>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.</p>
		<p>[1, 2]</p> <p>Adds one 2D data point to the statistical sums.</p>
$\Sigma-^{23}$	f STAT f Σ-	<p>[1, 2]</p> <p>Subtracts one data point from the statistical sums. Note $\Sigma-$ updates neither x_{\max} nor x_{\min}.</p>
$\chi^2_p(x)$	g [PROB] χ²:	
$\chi^2_\Delta(x)$	$\chi^2_p(x)$ etc.	<p>(1) {2}</p> <p><i>Chi-square distribution</i> (with its degrees of freedom given in I). $\chi^2_\Delta(x)$ equals $Q(\chi^2)$ on HP-21S. See Section 2 of the OM for an application example and pp. 220ff for more.</p>
$(\chi^2)^{-1}$		
$\chi^2:$	g [PROB] χ²:	<i>_submenu</i> . See p. 125.

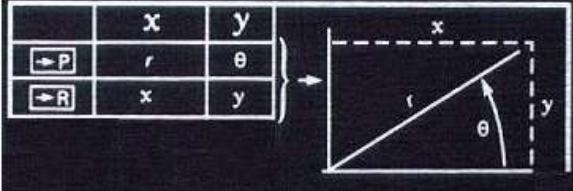
²³ $\Sigma+$ and $\Sigma-$ return *temporary information* as shown in Section 2 of the OM and disable *automatic stack lift*. Both commands may also be used for 2D vector adding and subtracting (see SUM and the corresponding example in Section 2 of the OM).

Item	Keystrokes	Remarks (see pp. 13ff for general information)									
$(-1)^x$	 	(1) {1, 2, 3, 8*, 9*, 10} If x is non-integer, returns $\cos(\pi x)$.									
$[M]^T$	 	(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.									
$[M]^{-1}$	 	(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. ²⁴									
$+/-$		(1) ‘Unary minus’, returns $x \times (-1)$. ²⁴									
$\pm\infty?$	 	(0) {2} → {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 $x = +\infty$,</td></tr><tr><td>true</td><td style="text-align: right;">-1</td><td>for $x = -\infty$, 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. ²⁴									
\times		(2) Returns $y \times x$ for compatible numeric objects. ²⁴									

²⁴ See the tables in the OM, Section 2 for details and compatibility.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
$\times\text{MOD}$		(3) {1, 2, 10} Returns $(z \times y) \bmod x$ for $x > 1, y > 0, z > 0$. ²⁵
/		(2) Like \times , but returns $y \times x^{-1}$. Returns $y \text{ div } x$ if both y and x are of DT1 or 10; cf. IDIV.
$\wedge\text{MOD}$		(3) {1, 2, 10} Returns $(z^y) \bmod x$ for $x > 1, y > 0, z > 0$. ²⁵
\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 <i>date</i> in x . Thus inverts DATE \rightarrow .
$\rightarrow\text{DEG}$		(1) {1, 2, 4} \rightarrow {4} Convert angles as described on pp. 147f.
$\rightarrow\text{D.MS}$		
$\rightarrow\text{GRAD}$		
$\rightarrow\text{HR}$		(1) {5} \rightarrow {2} Operates on <i>times</i> like $\rightarrow\text{REAL}$ below. Maintained for backward compatibility only.
$\rightarrow\text{H.MS}$		(1) {1, 2, 5} \rightarrow {5} Converts x to a sexagesimal <i>time</i> – cf. p. 111.
$\rightarrow\text{INT}$		(1) {1, 2, 10} \rightarrow {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 .

²⁵ See MOD.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
→MULπ	g L→ →MULπ	(1) {1, 2, 4} → {4} Converts angles as described on pp. 147f.
→POL	g →P	{2}; {1} → {2} Assumes X and Y containing 2D Cartesian coordinates of a point or components of a vector (x, y). Converts them to the respective polar coordinates or components (r, θ). Reverted by →REC, see there.  For switching the display format of complex numbers, see POLAR.
→RAD	g L→ →RAD	(1) {1, 2, 4} → {4} Converts angles as described on pp. 147f
→REAL	f .d (for closed input)	(1) {1, 2, 4, 5, 6, 10} → {2} Converts x to a real number. Any object (e.g. a time) tagged sexagesimal will be converted in a decimal number. For dates, the date format chosen is taken into account. Numbers shown as fractions will be displayed as decimal numbers (cf. FRACT and PROPFR). To return the real part of a complex number, take RE. To cut a complex number into its parts, use CC or CX→RE.
→REC	f R←	{2}; {1, 4} → {2} Assumes X and Y containing 2D polar coordinates of a point or components of a vector (r, θ). Converts them to the respective Cartesian coordinates or components (x, y). Inverted by →POL. – For switching the format of complex numbers, see POLAR.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
X		<p>Shuffles the contents of the <i>stack registers X, Y, Z, and T</i> at execution time.</p> <p>Examples:</p> <ul style="list-style-type: none"> xxyz works like $\text{ENTER} \uparrow$ (but does <u>not</u> disable <i>automatic stack lift!</i>), xyzt works like $x \leftarrow y$, yztx works like $R \downarrow$ in a 4-level <i>stack</i>, txyz works like $R \uparrow$ in a 4-level <i>stack</i>, <p>but also yytt or zzzx is possible.</p> <p>ATTENTION: This is a very powerful command although it does not look it. Note it will affect the <u>bottom four stack registers</u> only; there is no connection to A ... D, <u>regardless of stack size</u>. Playing with X, you may lose some <i>stack</i> contents and make a mess of the <i>stack</i> easily.</p>
$ M $		<p>(1) {8} \rightarrow {2}; {9} \rightarrow {3}</p> <p>Requires a square matrix in X and returns its determinant. The original matrix is stored in L.</p>
$ x $		<p>(1) {1, 2, 4, 10}</p> <p>Returns the absolute (unsigned) value of x.</p>
		<p>(1) {8*}</p> <p>Returns a matrix with the absolute values of all input matrix elements. Cf. ENORM.</p>
		<p>(1) {3} \rightarrow {2}</p> <p>Returns the <i>magnitude</i> $\sqrt{\text{Re}(x)^2 + \text{Im}(x)^2}$ in X.</p>
		<p>(1) {9*} \rightarrow {8}</p> <p>Returns a <i>real</i> matrix with the <i>magnitudes</i> of all input matrix elements. Cf. ENORM.</p>

Item	Keystrokes	Remarks (see pp. 13ff for general information)
	 	(2) {2, 3}; {1} → {2} Returns $\left(\frac{1}{x} + \frac{1}{y}\right)^{-1}$; useful in electrical engineering especially. Returns 0. for $x \times y = 0$.
%	 %	(1) {2}; {1} → {2} Returns $\frac{xy}{100}$, leaving y unchanged.
%MRR	 %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 $x = \text{FV}$ = future value after z periods, $y = \text{PV}$ = present value. For $z = 1$, Δ% returns the same result easier.
%T	 %T	(1) {2}; {1} → {2} Returns $100 \cdot x/y$, interpreted as % of total. Leaves y unchanged.
%Σ	 %Σ	(1) {2}; {1} → {2} Returns $100 \cdot x / \sum x$.
%+MG	 %+MG	(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: $P_{sale} = y / \left(1 - \frac{x}{100}\right)$ You may use %+MG for calculating net amounts as well; just enter a negative percentage in x .

Item	Keystrokes	Remarks (see pp. 13ff for general information)
\sqrt{x}	g x g EXP ✓x	(1) {1, 2, 3, 8*, 9*, 10}; ({1} → {2}, {1, 2} → {3}) Returns the square root of x . Square roots of non-square <i>long integers</i> will return <i>reals</i> . Square roots of negative <i>long integers</i> or <i>reals</i> will return <i>complex numbers</i> if CPXRES is set.
\int	f ADV ∫fdx ∫ var (listed in programs as ∫fd 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 . ²⁶ ATTENTION: \int fills all <i>stack registers</i> with x before calling the routine specified in PGMINT.
	g EQN ∫f ∫	Integrates the current equation. ²⁶
∫f	g EQN ∫f	Submenus. See pp. 122f.
∫fdx	f ADV ∫fdx	
¶	g 4 or g CPX f ¶ or f PARTS f ¶	(1) {2} → {4} Returns 180° (or equivalent) for $x < 0$ and 0 else. (1) {3} → {4} Returns the <i>phase</i> or argument $\arg(x) = \arctan\left(\frac{\text{Im}(x)}{\text{Re}(x)}\right)$. Cf. $ x $. (1) {9*} → {8} Returns a matrix with the <i>phases</i> of all input matrix elements. Cf. $ x $.

²⁶ See Section 4 of the OM for more.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
→	g L→	Menu of angular conversions. See p. 130.
ADV	g PRINT ADV	(0) Prints the current contents of the print buffer and a linefeed. ATTENTION: The printer will actually print only when a linefeed is sent to it.
CHAR	g PRINT f CHAR n	(0) Sends a single character (with the code specified) to the printer. Character codes $n > 127$ can only be specified indirectly. MODE setting will be honored. See ADV.
DLAY	g PRINT g DLAY n	(0) Sets a delay of n ticks (see TICKS) to be used with each linefeed on the printer.
LCD	g PRINT LCD	(0) Sends the contents of the entire LCD to the printer, so you get a hardcopy of the screen.
MODE	g PRINT g MODE n	(0) Sets print mode. Legal print modes are: 0: Use the printer font and character set wherever possible (default). All characters feature the same width (5 columns + 2 columns spacing). 1: Use the variable pitch display font, resulting in some jitter on the printout but packing more characters in a row. 2: Use the small display font, which allows for packing even more info in a row. 3: Send the output to the serial 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	g PRINT PROG	(0) Prints the listing of the <i>current program</i> (see Sect. 3 of the OM), one row per step.

Item	Keystrokes	Remarks (see pp. 13ff for general information)
Σ		(0) Prints the summation <i>registers</i> . Each <i>register</i> prints in one row starting with its label.
$\#$		(0) Sends a single <i>byte</i> , without translation, to the printer (e.g. a control code). $n > 127$ can only be specified indirectly. setting will not be honored. See .
$\#$		For inserting an integer $0 \leq n \leq 255$ in a single program step. Kept for backward compatibility to <i>WP 34S</i> only.
$\#B$		(1) {10} Counts the bits set in x (like on <i>HP-16C</i>).

Names of System 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. The contents of *items* printed on light blue can be stored by STOCFG and recalled by RCLCFG.

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).
ADM		Reserved integer variable for the <i>ADM</i> : 0: DEG, 1: D.MS, 2: RAD, 3: MULπ, 4: GRAD.
ALLENG	[A]	
ALPHA		System flags – see next chapter.
ALP.IN		
ASLIFT		
AUTOFF		
AUTXEQ		
B		Reserved variables for <i>registers B</i> and C .
C		
CARRY	[C]	
CPXj		System flags – see next chapter.
CPXRES	[I] (imaginary)	
D		Reserved variable for <i>register D</i> .
DECIM.		System flags – see next chapter.
DENANY		
DENFIX		
DENMAX		Reserved integer variable for the maximum denominator, filled by the command DENMAX.

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
DMY		System flags – see next chapter.
FRACT		
FV	g FIN TVM FV	Reserved real variable for the future value of your investment or loan in <i>TVM</i> . ²⁷
GRAMOD		Reserved integer variable determining how the AGRAPH image is displayed: ²⁸ 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		System flag – see next chapter.
I		Reserved variable for register I .
IGN1ER		System flags – see next chapter.
INTING		
ISM		Reserved integer variable for the <i>integer sign mode</i> : -1: sign and mantissa mode, 0: unsigned, 1: 1's complement, 2: 2's complement.
i%/a	g FIN TVM i%/a	Reserved real variable for the annual interest rate of your investment or loan in <i>TVM</i> . ²⁷
J		
K		
L		Reserved variables for registers J , K , and L .

²⁷ See Section 5 of the OM.

²⁸ Working like flags 34 and 35 in HP-42S.

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
LEAD.0	[L]	System flags – see next chapter.
LOWBAT		
Mat_A	f [MATX] SIM EQ Mat A	
Mat_B	f [MATX] SIM EQ Mat B	Reserved variables for solving systems of linear equations (see Sect. 2 of the OM).
Mat_X	f [MATX] SIM EQ Mat X	
MDY		System flags – see next chapter.
MULTx		
NPER	g [FIN] TVM n _{PER}	Reserved variable for the <u>total</u> number of <ul style="list-style-type: none"> payment periods for your loan or compounding periods for your investment.
NUM.IN		System flags – see next chapter.
OVERFL	B (big)	
PER/a	g [FIN] TVM f per/a	Reserved variable for the <u>annual</u> number of <ul style="list-style-type: none"> payments for your loan or compounding periods of your investment.
PMT	g [FIN] TVM PMT	Reserved variable for the payment per period for your investment or loan in TVM. ²⁷
POLAR	[X]	
PRINT		System flags – see next chapter.
PROPFTR		
PRTACT		
PV	g [FIN] TVM PV	Reserved variable for the present value of your investment or loan in TVM. ²⁷

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
QUIET		System flag – see next chapter.
REALDF		Reserved integer variable for real number display format: 0: ALL, 1: FIX, 2: SCI, 3: ENG.
REGS		Reserved variable for the 100×1 matrix of registers – if required.
RUNIO		
RUNTIM		
SLOW		System flags – see next chapter.
SOLVING		
SPCRES	D (danger)	
SSIZE8		
ST.A		
ST.B		
ST.C		
ST.D		
ST.T		
ST.X		
ST.Y		
ST.Z		
TDM24		
TRACE	T	System flags – see next chapter.
USB		
USER	f USER	System flag toggled by USER – see next chapter

Name	Keystrokes if applicable	Remarks (see pp. 13ff for general information)
VMDISP		
YMD		System flags – see next chapter.
αCAP		
↑Lim	f [ADV] ∫fdx ↑Lim etc.	Reserved <i>real</i> variables for the upper and lower limit of integration (s. the OM, Sect. 4).
#DEC		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	SBI	Flag ²⁹	Remarks
Time display	Time string	TDM24 (set)	<p>Set for international 24h time display. Expands the date display in the <i>status bar</i> to four digits for the year.</p> <p>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.</p>

²⁹ The flags printed on green may be written by the user, those printed on yellow are written by the system exclusively. *Startup default* status is given in parentheses for the flags set at startup – all others are clear at startup. Grey, orange, and light blue colors are used as in previous chapter.

Purpose	SBI	Flag ²⁹	Remarks
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.
Complex results	C / R	CPXRES	Set for allowing complex results also for real 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 imaginary number <i>i</i> , clear for <i>i</i> .
Polar notation	E / @	POLAR	Set for polar display of complex numbers, clear for rectangular.
Fraction display	—	FRACT	Set if fraction display is chosen, clear for decimal display (a b/c sets FRACT, .d clears it). See next entries.
Fraction kind 1	—	PROPFRACTION	Set for <i>proper fractions</i> , clear for <i>improper fractions</i> (a b/c 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>). ³⁰ For given DENMAX, this is the most precise way of displaying a decimal number as a fraction. See also next entry.

³⁰ E.g. for DENMAX = 5 and DENANY set, denominators 1, 2, 3, 4, and 5 are allowed.

Purpose	SBI	Flag ²⁹	Remarks
Fraction kind 3	see OM	DENFIX ³¹	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. ³²
Carry	c or C	CARRY	Reflects the status of the carry bit.
Overflow	o or 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. ³³
Alpha	A or α	ALPHA	Set for AIM, else clear.
Upper case	A / α	αCAP (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.
USB power	⎓ _B	USB	Set if your WP 43S is connected to a USB power source
Low battery	🔋	LOWBAT	Set if the battery voltage is low (see Section 2 of the OM).

³¹ DENFIX is evaluated only if DENANY is clear.

³² 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).

³³ Works like flag 3 in HP-16C.

Purpose	SBI	Flag ²⁹	Remarks
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).
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 (may also be by RCLCFG).
Beeper	—	QUIET	Set for disabled beeper.
Radix mark	—	DECIM. (set)	Set for a decimal point, clear for a comma.
Multiplication symbol	—	MULT \times (set)	Set for multiplication symbol \times , clear for \cdot .
Overflow from ALL	—	ALLENG	Set if numbers exceeding the range displayable in ALL or FIX will be shown in ENGineer's format, clear for SCientific.
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 Section 2 of the OM; see also GROW in the HP-42S Owner's Manual, p. 213.
Automatic OFF	—	AUTOFF (set)	Set if automatic shutdown is enabled. Else your WP 43S will remain ON until you will turn it off manually or battery voltage will drop below the limit of 2.0V (see Section 2 of the OM).
Automatic execution	—	AUTXEQ	Like flag 11 of HP-42S.

Purpose	SBI	Flag ²⁹	Remarks
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).
Enable automatic stack lift	—	ASLIFT (set)	Cleared by ENTER, CLX, Σ+, and Σ-, 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.

Summarizing, all *catalog* and *menu* calls themselves as well as the operations called by **EXIT**, **P/R**, **Α**, **Σ**, **Δ**, **▲**, **▼**, and **◀** 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. 115ff) for more about this topic.

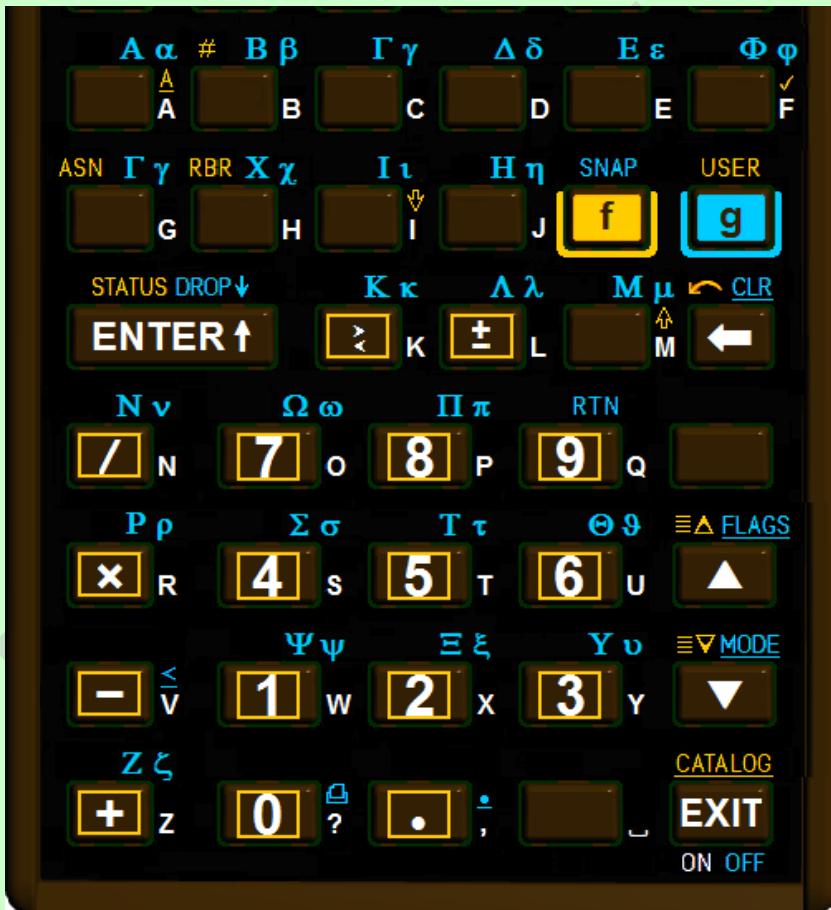
The *browsers RBR* and *STATUS* as well as the *application TIMER* use some keys for particular control purposes (e.g. **STO**, **RCL**, **.**, and numeric keys – see the OM, Section 5).

Command Parameter Input and Closing It

The following table shows what will happen when particular keys are pressed while command parameter input is not finished yet (see pp. 111ff 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
[A] ... [D], [I] ... [L, T], [X] ... [Z]	addressing	Enters the address of a <i>global GP register</i> or <i>user flag</i> .
[A] ... [Z] [g] [A] ... [g] [O] [f] [0] ... [f] [9]	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 ▼ and ▲ to switch cases for letters. See the virtual keyboard on p. 110 (cf. the OM, Sect. 2).
[ENTER↑]	arbitrary parameter input pending	If there is no input yet, assumes the default, if applicable. Closes pending input, interprets it as a <i>register</i> or <i>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.
[←]	arbitrary parameter input pending	Deletes the rightmost character keyed in. If there is nothing left, cancels the pending command, returning to the status of your WP 43S as it was before that input was started.
[0] ... [9]	addressing or specifying	Enters a numeric parameter, an address, or a local label. See Sections 1 and 3 of the OM for valid number ranges.
[.]	addressing	Header for <i>local registers</i> or <i>user flags</i> .

Keystrokes	Situation	Meaning
EXIT	arbitrary parameter input pending	If there is an open <i>menu</i> , closes it. Else cancels pending command input, returning to the status of your <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 open, so you can use all accessible characters for alphabetic searching (see pp. 119f). Note there is an 'alpha helper' printed on the rear 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. 109f 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, α	Appends the corresponding Latin or Greek letter to the <i>text string</i> x . Use \blacktriangledown and \blacktriangleup to switch cases. See the picture on previous page and cf. Section 2 of the OM.
[g] [A] ... [g] [O]		
[f] [#]	A, α	Appends # to the <i>text string</i> x .
[f] [#] base	\neg (A, α)	Closes input of a short integer
[f] [d..ms]		sexagesimal angle
[f] [.d]		date
[f] [h..ms]		sexagesimal time in X. ³⁴
[f] [x^2] (✓)	A, α	Appends a checkmark ✓ to the <i>text string</i> x .
[CC]	\neg (A, α)	Closes input of the first part (i.e. <i>real</i> part or <i>magnitude</i>) of a <i>complex</i> number in X and waits for input of its second part (i.e. <i>imaginary</i> part or <i>phase</i> , see the Key Response Table and Section 2 of the OM).
[f] [R↓] (↓)		A, α

³⁴ 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
ENTER↑	arbitrary input pending If there was input expected but not entered, cancels entry. Else closes input (in X) and checks the following conditions top-down: <ul style="list-style-type: none"> If this input is <u>alphanumeric</u> (i.e. if it contains at least one non-numeric character except .), takes it as a <i>text string</i>. Else (i.e. if this input is purely numeric) if it contains one CC, takes it as a <i>complex number</i>. Else if it contains two ., takes it as a <i>fraction</i>. Else if it contains one . or one E, takes it as a <i>real number</i>. Else (i.e. if it contains neither a CC nor a . nor an E), tests it for #: <ul style="list-style-type: none"> If it contains one # and a valid base trailing it then takes it as a <i>short integer</i>; else looks up if <u>previous entry</u> was a <i>short integer</i>: if true then takes the new input as another <i>short integer</i> of the same base; else takes the new input as a <i>long integer</i>. Then checks the new input (according to the condition met) as outlined in footnote 34 and interprets it. Finally, unless an error had to be thrown, copies x into Y .
f x>y	A, α Appends x to the <i>text string</i> x .
+/-	$\neg(A, \alpha)$ Changes the sign of the number entered. Affects the exponent if pressed after E .
f +/-	A, α Appends ± to the <i>text string</i> x .
E	$\neg(A, \alpha)$ Closes input of the mantissa and waits for input of the exponent (see Section 1 of the OM).
f E (↑)	A, α 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, α	Appends to the <i>text string</i> x .
	A, α	If $MULTx$ is set then appends , else to the <i>string</i> x .
	A, α	Appends to the <i>text string</i> x .
	A, α	Appends to the <i>text string</i> x .
 A ... F	\neg (A, α)	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, α)	Standard numeric input, appending the corresponding digit to x . Note you can enter ... <ul style="list-style-type: none"> • up to 16 digits plus a sign in the mantissa and up to three digits plus a sign in the exponent for a <i>real number</i> or any part of a <i>complex number</i>, • an arbitrary number of digits plus a sign for a <i>long integer</i>, • up to 64 bits for a <i>short integer</i>, or • up to 16 digits for the nominator and up to 4 digits for the denominator of a fraction.
	A, α	Appends to the <i>text string</i> x .
	A, α	Appends the respective digit to the <i>text string</i> x .
	A, α	Appends to the <i>text string</i> x .
	α	Turns to upper case for the following letters.
	A	Turns to lower case for the following letters.

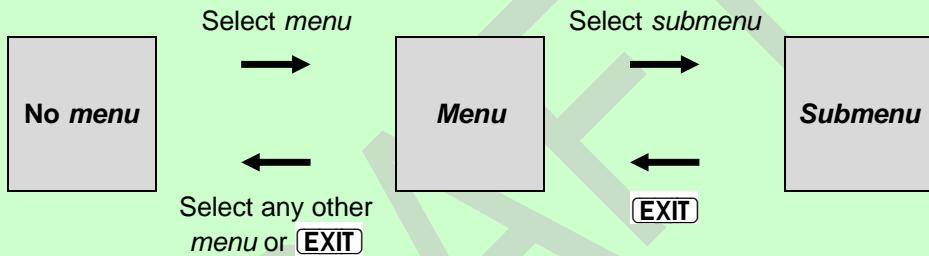
Keystrokes in mode(s)		Meaning
.	\neg (A, α)	Inserts a radix mark as selected.
		Separates <i>degrees</i> from <i>minutes</i> , <i>seconds</i> , and <i>hundredths of seconds</i> in angular input, so input format is dddd.mmsshh d.ms for sexagesimal angles (cf. p. 111 and Section 2 of the OM).
		Separates <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.mmssffff h.ms for sexagesimal <i>times</i> (cf. p. 111 and Section 2 of the OM).
Second .	\neg (A, α)	A 2 nd . in input indicates a fraction. See the OM, Sect. 2 for examples. The 2 nd . just separates the nominator and the denominator in input. Note you cannot enter E after you entered . twice – but you may delete the 2 nd dot while editing the input.
.	A, α	Appends , to the <i>text string x</i> .
f .	A, α	Appends . to the <i>text string x</i> .
R/S	! , program waits for input	Closes input and starts its checks and interpretation like ENTER! above. Resumes program execution.
	A, α	Appends a blank space to the <i>text string x</i> .
EXIT	arbitrary input pending	If there is an open <i>menu</i> , closes it. Else closes pending numeric or alphanumeric input and releases it for interpretation.

There are many more characters you can enter via the three alpha menus or **CATALOG CHARS**. See p. 116 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'αINTL,
- 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. 119f).

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. 119f.

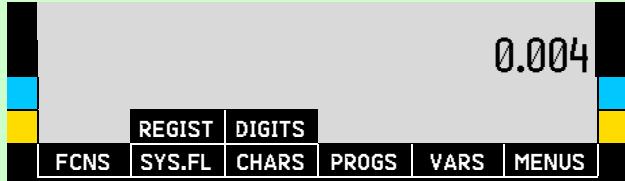
The contents of the various branches of **CATALOG** are presented below. Note they are printed in reverse order compared to the display of your *WP 43S*, taking care of your top-down reading habits:

							Remarks
CATALOG:	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 740 functions provided
	2COMPL	$\sqrt[3]{x}$	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:	ALLENG	ALPHA	ALP.IN	...			system flags (cf. pp. 99ff)
	...						
CHARS:	αINTL	$\alpha\ldots\Omega$		αMATH	$\text{My}\alpha$	$\alpha\bullet$	character branches
$\alpha\text{INTL}:$	A	\tilde{A}	\ddot{A}	\hat{A}	$\tilde{\hat{A}}$...	international Latin letters, see p. 128
$\alpha\text{MATH}:$	<	\leq	=	...			mathematical operators and symbols, see p. 129
$\alpha\ldots\Omega:$	A	B	Γ	Δ	...		Greek letters, see p. 129
$\alpha\bullet:$!	;	...				punctuation marks, see p. 130
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	n _{PER} ...	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. 121ff 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

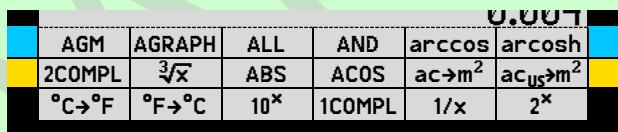
Three branches of **CATALOG** are expandable (**MENUS** and the submenus of **PROGS** and **VARS**) since you may create *items* of these kinds (cf. the OM, Sect. 6); the other ones are fixed size (**FCNS**, **REGIST**, **SYS.FL**, **DIGITS**, and **CHARS**) since all functions, *register names*, *system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



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:



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.³⁵ **EXIT** will just leave the open branch without doing anything.

³⁵ 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 740 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 *I/O* on pp. 13ff. Find the other predefined *items* provided (*system flags* and variable *names*) on pp. 99ff.

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 \blacktriangle and \blacktriangledown as explained in previous chapter. In CONST and major parts of CATALOG (FCNS, SYS.FL, MENUS, REGIST, DIGITS, CHARS α 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; ³⁶ AIM is on. <table border="1" style="width: 100%; border-collapse: collapse;"><tr><td style="text-align: center; padding: 2px;">AGM</td><td style="text-align: center; padding: 2px;">AGRAPH</td><td style="text-align: center; padding: 2px;">ALL</td><td style="text-align: center; padding: 2px;">AND</td><td style="text-align: center; padding: 2px;">arccos</td><td style="text-align: center; padding: 2px;">arcosh</td></tr><tr><td style="text-align: center; padding: 2px;">2COMPL</td><td style="text-align: center; padding: 2px;">$\sqrt[3]{x}$</td><td style="text-align: center; padding: 2px;">ABS</td><td style="text-align: center; padding: 2px;">ACOS</td><td style="text-align: center; padding: 2px;">$ac \rightarrow m^2$</td><td style="text-align: center; padding: 2px;">$ac_{us} \rightarrow m^2$</td></tr><tr><td style="text-align: center; padding: 2px;">$^{\circ}C \rightarrow ^{\circ}F$</td><td style="text-align: center; padding: 2px;">$^{\circ}F \rightarrow ^{\circ}C$</td><td style="text-align: center; padding: 2px;">10^x</td><td style="text-align: center; padding: 2px;">1COMPL</td><td style="text-align: center; padding: 2px;">$1/x$</td><td style="text-align: center; padding: 2px;">2^x</td></tr></table>	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
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														

³⁶ ... 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. F)																		
Return	Your WP 43S displays a view starting with the first <i>item</i> starting with this character ³⁷ e.g.																		
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">fm.\rightarrowm</td><td style="padding: 2px;">FP</td><td style="padding: 2px;">F_p(x)</td><td style="padding: 2px;">FP?</td><td style="padding: 2px;">fr\rightarrowdB</td><td style="padding: 2px;">FS?</td></tr> <tr> <td style="background-color: #d9e1f2; padding: 2px;">FF</td><td style="background-color: #d9e1f2; padding: 2px;">FIB</td><td style="background-color: #d9e1f2; padding: 2px;">FILL</td><td style="background-color: #d9e1f2; padding: 2px;">FIX</td><td style="background-color: #d9e1f2; padding: 2px;">FLASH?</td><td style="background-color: #d9e1f2; padding: 2px;">FLOOR</td></tr> <tr> <td style="background-color: #d9e1f2; padding: 2px;">FB</td><td style="background-color: #d9e1f2; padding: 2px;">FBR</td><td style="background-color: #d9e1f2; padding: 2px;">FC?</td><td style="background-color: #d9e1f2; padding: 2px;">FC?C</td><td style="background-color: #d9e1f2; padding: 2px;">FC?F</td><td style="background-color: #d9e1f2; padding: 2px;">FC?S</td></tr> </table>	fm.\rightarrowm	FP	F_p(x)	FP?	fr\rightarrowdB	FS?	FF	FIB	FILL	FIX	FLASH?	FLOOR	FB	FBR	FC?	FC?C	FC?F	FC?S
fm.\rightarrowm	FP	F_p(x)	FP?	fr\rightarrowdB	FS?														
FF	FIB	FILL	FIX	FLASH?	FLOOR														
FB	FBR	FC?	FC?C	FC?F	FC?S														
3 User input	Second character of the <i>item</i> desired (e.g. S)																		
Return	Your WP 43S displays a view starting with the first <i>item</i> starting with the string you specified e.g.																		
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">f''(x)</td><td style="padding: 2px;">GAP</td><td style="padding: 2px;">GaussF</td><td style="padding: 2px;">GCD</td><td style="padding: 2px;">g_d</td><td style="padding: 2px;">g_d$^{-1}$</td></tr> <tr> <td style="background-color: #d9e1f2; padding: 2px;">fz_{UK}\rightarrowm³</td><td style="background-color: #d9e1f2; padding: 2px;">fz_{US}\rightarrowm³</td><td style="background-color: #d9e1f2; padding: 2px;">F_Δ(x)</td><td style="background-color: #d9e1f2; padding: 2px;">F_Δ'(x)</td><td style="background-color: #d9e1f2; padding: 2px;">F$^{-1}$(p)</td><td style="background-color: #d9e1f2; padding: 2px;">f'(x)</td></tr> <tr> <td style="background-color: #d9e1f2; padding: 2px;">FS?</td><td style="background-color: #d9e1f2; padding: 2px;">FS?C</td><td style="background-color: #d9e1f2; padding: 2px;">FS?F</td><td style="background-color: #d9e1f2; padding: 2px;">FS?S</td><td style="background-color: #d9e1f2; padding: 2px;">ft.\rightarrowm</td><td style="background-color: #d9e1f2; padding: 2px;">ft_{US}\rightarrowm</td></tr> </table>	f''(x)	GAP	GaussF	GCD	g_d	g_d$^{-1}$	fz_{UK}\rightarrowm³	fz_{US}\rightarrowm³	F_Δ(x)	F_Δ'(x)	F$^{-1}$(p)	f'(x)	FS?	FS?C	FS?F	FS?S	ft.\rightarrowm	ft_{US}\rightarrowm
f''(x)	GAP	GaussF	GCD	g_d	g_d$^{-1}$														
fz_{UK}\rightarrowm³	fz_{US}\rightarrowm³	F_Δ(x)	F_Δ'(x)	F$^{-1}$(p)	f'(x)														
FS?	FS?C	FS?F	FS?S	ft.\rightarrowm	ft_{US}\rightarrowm														
4 User input	Press the corresponding softkey e.g. for FS?S																		

³⁷ 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 α (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 740);
- 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&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
ADV	SOLVE	SLVQ	$f'(x)$	Π_n	Σ_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	
BITS	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	
CLK	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	
	CLΣ	CLP	CF	CLMENU	CLCVAR	CLX	
CLR	CLREGS	CLPall	CLFall		CLLCD	CLSTK	almost as in HP-42S
	CLall					RESET	
CONST							catalog of constants, see pp. 130ff
CPX	dot	cross	UNITV	Re	conj	Re \Rightarrow Im	special complex functions
	CX \rightarrow RE	RE \rightarrow CX	sign	Im	x	\neq	
DISP	FIX	SCI	ENG	ALL	ROUND1	ROUND	display rounding and shifts, formats and settings, mostly for reals
	SDL	SDR			RDP	RSD	
	CHINA	EUROPE	INDIA	JAPAN	UK	USA	
	GAP		RANGE	RANGE?		DSTACK	

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

Menu							Remarks
LOOP	DSE	DSZ	DSL	ISE	ISZ	ISG	
	DEC					INC	
MATX	NEW	[M] ⁻¹	M	[M] ^T	SIM EQ	EDIT	matrix operations (the items sorted almost as in HP-42S)
	dot	CROSS	UNITV	DIM	INDEX	EDITN	
	ENORM		STOEL	RCLEL	PUTM	GETM	
	I+	I-	STOIJ	RCLIJ	J-	J+	
	RSUM	RNORM	M.LU	DIM?		R>R	
	EIGVAL					EIGVEC	
M.EDIT	←	↑	OLD	GOTO	↓	→	Matrix Editor as in HP-42S
	INSR		DELR		WRAP	GROW	
M.SIMQ	Mat A	Mat B				Mat X	solver for systems of linear equations
MODE	SF	DEG	RAD	GRAD	MULπ	CF	mode settings; SYSTEM is not available on the simulator
			RM		SETSIG	DENMAX	
	SYSTEM						
MyMenu							will show up out of AIM ³⁸
Myα							will show up in AIM ³⁸
PARTS	IP	FP	MANT	EXPT	sign	DECOMP	some overlaps with HP-42S CONVERT
					x	%	
					Re	Im	
PRINT	☒x	☒r	☒Σ	☒ADV	☒LCD	☒PROG	the PRINT commands of the HP-42S
	☒STK	☒REGS	☒USER	☒TAB	☒#	☒CHAR	
				☒WIDTH	☒DISPLAY	☒MODE	

³⁸ ... as long as no other menu is called (see Section 6 of the OM).

Menu							Remarks
<u>PROB</u>	Norml:	t:	C _{yx}	P _{yx}	F:	χ^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	Γ(x)	
Binom:	Binom _p		Binom _△	Binom _△		Binom ⁻¹	
Cauch:	Cauch _p		Cauch _△	Cauch _△		Cauch ⁻¹	
Expon:	Expon _p		Expon _△	Expon _△		Expon ⁻¹	
F:	F _p (x)		F _△ (x)	F _△ (x)		F ⁻¹ (p)	
Geom:	Geom _p		Geom _△	Geom _△		Geom ⁻¹	
Hyper:	Hyper _p		Hyper _△	Hyper _△		Hyper ⁻¹	
LgNrm:	LgNrm _p		LgNrm _△	LgNrm _△		LgNrm ⁻¹	
Logis:	Logis _p		Logis _△	Logis _△		Logis ⁻¹	
NBin:	NBin _p		NBin _△	NBin _△		NBin ⁻¹	
Norml:	Norml _p		Norml _△	Norml _△		Norml ⁻¹	
Poiss:	Poiss _p		Poiss _△	Poiss _△		Poiss ⁻¹	
t:	t _p (x)		t _△ (x)	t _△ (x)		t ⁻¹ (p)	
Weibl:	Weibl _p		Weibl _△	Weibl _△		Weibl ⁻¹	
χ^2 :	χ^2_p (x)		χ^2_{Δ} (x)	χ^2_{Δ} (x)		$(\chi^2)^{-1}$	
<u>P.FN</u>	INPUT	END	ERR	TICKS	PAUSE	P.FN2	additional programming functions (avoided a multi-view menu here).
	PSTO	PRCL	VARMNU	MVAR	CNST	PUTK	
	R-CLR	R-COPY	R-SORT	R-SWAP	LocR	PopLR	
P.FN2	MENU	KEYG	KEYX	CLMENU	EXITall	RTN+1	
	SDL	SDR	MSG	NOP			
	BACK	CASE	SKIP	AGRAPH	PIXEL	POINT	

Menu							Remarks	
STAT	$\Sigma+$	\bar{x}	s	s	s_m	SUM	for sample statistics.	
	$\Sigma-$	\bar{x}_w	s_w	s_w	s_{mw}			
	$CL\Sigma$	\bar{x}_G	ε	ε_p	ε_m	PLOT	for curve fitting and 2d sample statistics.	
	L.R.	r	s_{xy}	Cov	\hat{x}	\hat{y}		
	s(a)	\bar{x}_H					for choosing the fit model(s)	
		\bar{x}_{RMS}	x_{max}	x_{min}		OrthoF		
LinF	ExpF	LogF	PowerF			BestF	for choosing the fit model(s)	
	GaussF	CauchF	ParabF	HypF	RootF			
STK	$x \gtrless$	$y \gtrless$	$z \gtrless$	$t \gtrless$	\gtrless	DROPy	stack related operations.	
TEST	$x < ?$	$x \leq ?$	$x = ?$	$x \neq ?$	$x \geq ?$	$x > ?$	binary tests.	
	INT?	EVEN?	ODD?	PRIME?	LEAP?	FP?		
	ENTRY?	KEY?	LBL?	STRI?	CONVG?	TOP?		
	$x = +0?$	$x = -0?$	$x \approx ?$	MATR?	CPX?	REAL?		
	SPEC?	NaN?		M.SQR?				
TRI	sin	arcsin	cos	arccos	tan	arctan	trigonometric & hyperbolic functions (cf. EXP).	
	sinc	sincπ						
	sinh	arsinh	cosh	arcosh	tanh	artanh		
U→	E:	P:	year → s	F&p:	m:	x:	unit conversions (see pp. 140ff).	
	$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$	s → year		v:	A:		
	power ratio → dB	dB → power ratio	Nm → lbf·ft	→ Nm	field ratio → dB	dB → field ratio		
	A:	$\text{acre} \rightarrow \text{m}^2$	$\text{ha} \rightarrow \text{m}^2$	$\text{m}^2 \rightarrow \text{ha}$	$\text{acre}_{\text{US}} \rightarrow \text{m}^2$	$\text{m}^2 \rightarrow \text{acre}_{\text{US}}$	units of area	
	E:	cal → J	J → cal	Btu → J	J → Btu	Wh → J	J → Wh	units of energy

Menu							Remarks
F&p:	lbf → N	N → lbf	bar → Pa	Pa → bar	psi → Pa	Pa → psi	units of force and pressure
	in.Hg → Pa	Pa → in.Hg	torr → Pa	Pa → torr	atm → Pa	Pa → atm	
			mmHg → Pa	Pa → mmHg			
M:	lb. → kg	kg → lb.	cwt → kg	kg → cwt	oz → kg	kg → oz	units of mass
	stone → kg	kg → stone	short cwt → kg	kg → sh.cwt	tr.oz → kg	kg → tr.oz	
	ton → kg	kg → ton	short ton → kg	kg → short ton	carat → kg	kg → carat	
P:	hp_E → W	W → hp_E	hp_UK → W	W → hp_UK	hp_M → W	W → hp_M	units of power
V:	gl_UK → m³	m³ → gl_UK	qt. → m³	m³ → qt.	gl_US → m³	m³ → gl_US	units of volume
	floz_UK → m³	m³ → floz_UK	barrel → m³	m³ → barrel	floz_US → m³	m³ → floz_US	
X:	au → m	m → au	ly → m	m → ly	pc → m	m → pc	units of length
	mi. → m	m → mi.	nmi. → m	m → nmi.	ft. → m	m → ft.	
	in. → m	m → in.			yd. → m	m → yd.	
	fathom → m	m → fathom	point → m	m → point	survey foot_US → m	m → survey foot_US	
X.FN	AGM	B_n	B_n*	erf	erfc	Orthog	advanced mathematical functions like Beta, Bessel, etc.
	FIB	g_d	g_d⁻¹	I_xy_z	IΓ_p	IΓ_q	
	J_y(x)	lnβ	lnΓ	max	min	NEXTP	
	W_m	W_p	W⁻¹	β(x,y)	γ_xy	Γ_xy	
	ζ(x)	(-1)^x					
Orthog	H_n	L_m	L_max	P_n	T_n	U_n	orthogonal polynomials
	H_np						

Menu	<input type="checkbox"/>	Remarks					
<u>αINTL</u>	A a	À à	Á á	Â â	Ã ã	Ä ä	[α] catalog of all Latin letters provided. ³⁹ 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.
	Å å	Æ æ	Ā ā	Ă ā	Ą ą	Ɓ ɓ	
	C c	Ç ç	Ć č	Č č	D d	Ɗ ڏ	
	ڏ ڏ	ڇ ڏ	E e	ڦ ڏ	É é	ڦ ڏ	
	ڦ ڏ	ڦ ڏ	ڦ ڏ	ڦ ڏ	ڦ ڏ	ڦ ڏ	
	F f	G g	ږ ڗ	H h	I i	ڙ ڙ	
	ି ି	ି ି	ି ି	ି ି	ି ି	ି ି	
	ି ି	J j	K k	L l	ି ି	ି ି	
	ି ି	M m	N n	ି ି	ି ି	ି ି	
	O o	ୟ ଯ	ୟ ଯ	ୟ ଯ	ୟ ଯ	ୟ ଯ	
	ୟ ଯ	ୟ ଯ	ୟ ଯ	ୟ ଯ	ୟ ଯ	ୟ ଯ	
	R r	ର ର	ର ର	S s	ର ର	ର ର	
	ର ର	ର ର	T t	ର ର	ର ର	ର ର	
	ର ର	ର ର	ର ର	ର ର	ର ର	ର ର	
	ର ର	ର ର	ର ର	ର ର	ର ର	ର ର	
	X x	Y y	ଯ ଯ	ଯ ଯ	ଯ ଯ	ଯ ଯ	
	ଯ ଯ	ଯ ଯ	ଯ ଯ				

³⁹ See https://de.wikipedia.org/wiki/Liste_lateinischer_Alphabete#Erweiterungen.

Menu							Remarks
αMATH	<	≤	=	≈	≥	>	[α] for comparison symbols, parentheses & brackets, as well as more mathematical and related symbols. You can reach every character by 3 keystrokes maximum.
	{	[()]	}	
	× / · ⁴⁰	÷ / :	∫	∞	∞	∞	
	¬	¬	∨	≠		&	
	✗	✗	⊥	✗	✓	✗	
	✗	✗	✗	✗	✗	✗	
	:=	≈	≡	E	C	R	
	⊗	⊗	⊕				
α.FN	x→α	αRL	αRR	αSL	αSR	α→x	dedicated functions for alphanumeric strings, plus font browser
					αLENG?	αPOS?	
	FBR						
Α...Ω	Α α	Β β	Γ γ	Δ δ	Ε ε	Ζ ζ	[α] Greek letters. The keyboard grants direct access to 24 of them. ⁴¹ 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	
	ς	τ τ	υ υ	φ φ	χ χ	ψ ψ	
	Ω ω	ά	έ	ή	ί	ύ	
	Ϊ ī	ó	ú	ÿ ü	ö	ó	

⁴⁰ With startup default settings, the multiplication dot is found here and the multiplication cross is called via  in A/M. If MULTx is clear, however, this dot is called via  in A/M and the multiplication cross via [αMATH](#). The symbols : and ÷ will swap, too.

⁴¹ The Greek alphabet (sic!) goes **alpha**, **beta**, **gamma**, **delta**, **e-psilon**, **zeta**, **ēta**, **theta**, **iota**, **kappa**, **lambda**, **my**, **ny**, **xi**, **o-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 obvious reasons – these suffice: there is no goose named Seitsemän appearing in that novel).

Menu							Remarks
$\alpha\bullet$!	;	:	'	"	✓	[α] for punctuation marks, currency symbols, arrows, and further special characters.
	ı	ż	ſ	ø	~	њ	
	\$	€	%	&	£	¥	
	←	↑	↓	↓	→	↑	
	«	»	»	⌚	•	*	
	⌚	⌚	⌚	⌚	⌚	⌚	
	„	”	…	—			
Σ	n	Σx	Σx^2	Σxy	Σy^2	Σy	all the sums necessary for the statistics in <u>STAT</u> .
		$\Sigma \ln x$	$\Sigma \ln^2 x$	$\Sigma \ln xy$	$\Sigma \ln^2 y$	$\Sigma \ln y$	
	$\Sigma x^2 y$	$\Sigma x \ln y$		$\Sigma \ln y/x$		$\Sigma y \ln x$	
	$\Sigma x^2/y$	Σ^1/x	Σ^1/x^2	$\Sigma x/y$	Σ^1/y^2	Σ^1/y	
	Σx^3	Σx^4					
$\leftarrow\rightarrow$	\rightarrow DEG	\rightarrow RAD	\rightarrow GRAD		\rightarrow D.MS	\rightarrow MUL π	angular conversions, cf. pp. 147f.
	DEG \rightarrow	RAD \rightarrow	GRAD \rightarrow		D.MS \rightarrow	MUL π \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_{α}	F_{δ}	
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 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.⁴² 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
a (0) ⁴³	365,242 5 d <i>(per definition)</i>	Gregorian year
a₀	$5,291\,772\,109\,03 \cdot 10^{-11}$ m <i>($1,5 \cdot 10^{-10}$)</i>	Bohr radius $a_0 = \alpha / 4\pi R_\infty$
a_{Moon}	$3,844 \cdot 10^8$ m <i>($1 \cdot 10^{-3}$)</i>	Semi-major axis of the Moon's orbit around the earth $\approx 1,3$ light seconds.

⁴² 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.

⁴³ 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_{\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> $\approx 499 \text{ light seconds} \approx 8 \text{ light minutes}$.
c	$2,997\,924\,58 \cdot 10^8 \text{ m/s}$ <i>(exact)</i>	Speed of light in vacuum $\approx 300\,000 \frac{\text{km}}{\text{s}} = 300 \frac{\text{km}}{\text{ms}} =$ $300 \frac{\text{m}}{\mu\text{s}} = 30 \frac{\text{cm}}{\text{ns}} = 0,3 \frac{\text{mm}}{\text{ps}}$ etc.
c_1 (5)	$3,741\,771\,85 \dots 10^{-16} \text{ W m}^2$ <i>(exact)</i>	First radiation constant $c_1 = 2\pi h c^2$
c_2	$0,014\,387\,768\,77 \dots \text{ m} \cdot \text{K}$ <i>(exact)</i>	Second radiation constant $c_2 = hc/k$
e	$1,602\,176\,634 \cdot 10^{-19} \text{ A s}$ <i>(exact)</i>	Elementary charge $e = \frac{2}{K_J R_K} = \Phi_0 G_0$
e_E	$2,718\,281\,828\,459\,045\,2\dots$	<i>Euler's e.</i>
F	$96\,485,332\,12 \dots \text{ A s/mol}$ <i>(exact)</i>	<i>Faraday constant</i> $F = e N_A$
F_α (10)	$2,502\,907\,875\,095\,892\,8\dots$	<i>Feigenbaum's α and δ</i>
F_δ	$4,669\,201\,609\,102\,990\,6\dots$	
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 γ from other authors. See \mathbf{GM}_{\oplus} below for a more precise value.
G_0	$7,748\,091\,729 \dots 10^{-5} / \Omega$ <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>

Name	Numeric value and <i>rel. SD</i>	Remarks
g_e (15)	-2,002 319 304 362 56 <i>(1,7·10⁻¹³)</i>	Landé's electron g-factor
GM_{\oplus}	3,986 004 418 · 10 ¹⁴ m ³ /s ² <i>(2,0·10⁻⁹)</i>	Newtonian constant of gravitation times the Earth's mass with its atmosphere included (according to WGS84 ⁴⁴)
g_{\oplus}	9,806 65 m/s ² (<i>per def.</i>)	Standard earth acceleration (note the actual values vary from 9,780 4 m/s ² at equator to 9,832 2 m/s ² at the poles)
h	6,626 070 15 · 10 ⁻³⁴ J s <i>(exact)</i>	Planck constant
\hbar	1,054 571 817 ... · 10 ⁻³⁴ J s <i>(exact)</i>	Reduced Planck constant $\hbar = h/2\pi$
k (20)	1,380 649 · 10 ⁻²³ J/K <i>(exact)</i>	Boltzmann constant $k = R/N_A$
K_J	4,835 978 48 ... · 10 ¹⁴ Hz/V <i>(exact)</i>	Josephson constant $K_j = 2e/h$
l_{PL}	1,616 255 · 10 ⁻³⁵ m <i>(1,1·10⁻⁵)</i>	Planck length $l_{PL} = t_{PL}c$
m_e	9,109 383 701 5 · 10 ⁻³¹ kg <i>(3,0·10⁻¹⁰)</i>	Electron mass $\triangleq 511,00$ keV
M_{Moon}	7,349 · 10 ²² kg <i>(5·10⁻⁴)</i>	Mass of the Moon
m_n (25)	1,674 927 498 04 · 10 ⁻²⁷ kg <i>(5,7·10⁻¹⁰)</i>	Neutron mass $\triangleq 939,57$ MeV

⁴⁴ See http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html

Name	Numeric value and <i>rel. SD</i>	Remarks
m_n/m_p	1,001 378 419 31 <i>(4,9·10⁻¹⁰)</i>	Neutron to proton mass ratio
m_p	1,672 621 923 69·10 ⁻²⁷ kg <i>(3,1·10⁻¹⁰)</i>	Proton mass $\doteq 938,27$ MeV
m_{PL}	2,176 435·10 ⁻⁸ kg <i>(1,1·10⁻⁵)</i>	Planck mass $m_{PL} = \sqrt{\hbar c/G} \approx 22$ µg
m_p/m_e	1 836,152 673 43 <i>(6,0·10⁻¹¹)</i>	Proton to electron mass ratio
m_u (30)	1,660 539 066 60·10 ⁻²⁷ kg <i>(3,0·10⁻¹⁰)</i>	Atomic mass constant $\approx 10^{-3}$ kg/ N_A
$m_u c^2$	1,492 418 085 60·10 ⁻¹⁰ J <i>(3,0·10⁻¹⁰)</i>	Energy equivalent of the atomic mass constant $\approx 931,49$ MeV
m_μ	1,883 531 627·10 ⁻²⁸ kg <i>(2,2·10⁻⁸)</i>	Muon mass $\doteq 105,66$ MeV
M_\odot	1,989 1·10 ³⁰ kg <i>(5·10⁻⁵)</i>	Mass of the Sun
M_\oplus	5,973 6·10 ²⁴ kg <i>(5·10⁻⁵)</i>	Mass of the Earth. See GM_\oplus above for a more precise value.
N_A (35)	6,022 140 76·10²³ / mol <i>(exact)</i>	Avogadro's number
NaN	<i>Not a Number</i>	See p. 168 and the corresponding entry in Section 5 of the OM.
P_0	101 325 Pa <i>(per def.)</i>	Standard atmospheric pressure
R	8,314 462 618... J/mol K <i>(exact)</i>	Molar gas constant

Name	Numeric value and <i>rel. SD</i>	Remarks
r_e	$2,817\,940\,326\,2 \cdot 10^{-15} \text{ m}$ $(4,5 \cdot 10^{-10})$	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_{Moon}	$1,737\,530 \cdot 10^6 \text{ m}$ $(5 \cdot 10^{-7})$	Mean radius of the Moon
R_∞	$10\,973\,731,568\,160 / \text{m}$ $(1,9 \cdot 10^{-12})$	Rydberg constant $R_\infty = \frac{\alpha^2 m_e c}{2 h}$
R_\odot	$6,96 \cdot 10^8 \text{ m}$ $(5 \cdot 10^{-3})$	Mean radius of the sun
R_\oplus	$6,371\,010 \cdot 10^6 \text{ m}$ $(5 \cdot 10^{-7})$	Mean radius of the Earth
a (45)	$6,378\,137\,0 \cdot 10^6 \text{ m}$ (<i>p. def.</i>)	Semi-major axis
b	$6,356\,752\,314\,2 \cdot 10^6 \text{ m}$ $(1,6 \cdot 10^{-11})$	Semi-minor axis
e^2	$6,694\,379\,990\,14 \cdot 10^{-3}$ $(1,5 \cdot 10^{-12})$	First eccentricity squared
e'^2	$6,739\,496\,742\,28 \cdot 10^{-3}$ $(1,5 \cdot 10^{-12})$	Second eccentricity squared
f^{-1}	$298,257\,223\,563$ (<i>per def.</i>)	Flattening parameter
T_0 (50)	$273,15 \text{ K}$ (<i>per definition</i>)	= 0°C , standard temperature
T_p	$1,416\,785 \cdot 10^{32} \text{ K}$ $(1,1 \cdot 10^{-5})$	Planck temperature $T_P = \frac{c^2}{k} \sqrt{\frac{\hbar c}{G}} = \frac{M_P c^2}{k} = \frac{E_P}{k}$

Name	Numeric value and <i>rel. SD</i>	Remarks
t_{PL}	$5,391\,245 \cdot 10^{-44} \text{ s}$ <i>(1,1·10⁻⁵)</i>	Planck time $t_{PL} = l_{PL}/c$
V_m	$0,022\,413\,969\,5\dots \text{ m}^3/\text{mol}$ <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	$376,730\,313\,668 \Omega$ <i>(1,5·10⁻¹⁰)</i>	Characteristic impedance of vacuum
α (55)	$7,297\,352\,569\,3 \cdot 10^{-3}$ <i>(1,5·10⁻¹⁰)</i>	Fine-structure constant $\alpha = \frac{e^2}{2\varepsilon_0 h c} \approx \frac{1}{137}$
γ	$6,674\,30 \cdot 10^{-11} \text{ m}^3/\text{kg s}^2$ <i>(2,2·10⁻⁵)</i>	Newtonian constant of gravitation; also known as G from other authors. See GM_\oplus below for a more precise value.
γ_{EM}	$0,577\,215\,664\,901\,532\,9\dots$	Euler-Mascheroni constant
γ_p	$2,675\,221\,874\,4 \cdot 10^8 \text{ Hz/T}$ <i>(4,2·10⁻¹⁰)</i>	Proton gyromagnetic ratio $\gamma_p = 4\pi \mu_p / h$
$\Delta\nu_{Cs}$	9 192 631 770 Hz <i>(exact)</i>	Hyperfine transition frequency of ^{133}Cs
ε_0 (60)	$8,854\,187\,812\,8 \cdot 10^{-12} \frac{\text{As}}{\text{Vm}}$ <i>(1,5·10⁻¹⁰)</i>	Vacuum electric permittivity $\varepsilon_0 = 1/\mu_0 c^2$ (note the so-called Coulomb's constant is just $1/4\pi\varepsilon_0$)

Name	Numeric value and <i>rel. SD</i>	Remarks
λ_c	$2,426\,310\,238\,67 \cdot 10^{-12} \text{ m}$ $(3,0 \cdot 10^{-10})$	
λ_{cn}	$1,319\,590\,905\,81 \cdot 10^{-15} \text{ m}$ $(5,7 \cdot 10^{-10})$	Compton wavelengths of the electron $\lambda_c = h/m_e c$, neutron $\lambda_{cn} = h/m_n c$, and proton $\lambda_{cp} = h/m_p c$, respectively
λ_{cp}	$1,321\,409\,855\,39 \cdot 10^{-15} \text{ m}$ $(3,1 \cdot 10^{-10})$	
μ_0	$1,256\,637\,062\,12 \cdot 10^{-6} \frac{\text{Vs}}{\text{Am}}$ $(1,5 \cdot 10^{-10})$	Vacuum magnetic permeability
μ_B (65)	$9,274\,010\,078\,3 \cdot 10^{-24} \frac{\text{J}}{\text{T}}$ $(3,0 \cdot 10^{-10})$	Bohr magneton $\mu_B = e\hbar/2m_e$
μ_e	$-9,284\,764\,704\,3 \cdot 10^{-24} \frac{\text{J}}{\text{T}}$ $(3,0 \cdot 10^{-10})$	Electron magnetic moment
μ_e/μ_B	$-1,001\,159\,652\,181\,28$ $(1,7 \cdot 10^{-13})$	Ratio of electron magnetic moment to Bohr's magneton
μ_n	$-9,662\,365\,1 \cdot 10^{-27} \frac{\text{J}}{\text{T}}$ $(2,4 \cdot 10^{-7})$	Neutron magnetic moment
μ_p	$1,410\,606\,797\,36 \cdot 10^{-26} \frac{\text{J}}{\text{T}}$ $(4,2 \cdot 10^{-10})$	Proton magnetic moment
μ_u (70)	$5,050\,783\,746\,1 \cdot 10^{-27} \frac{\text{J}}{\text{T}}$ $(3,1 \cdot 10^{-10})$	Nuclear magneton $\mu_u = e\hbar/2m_p$
μ_μ	$-4,490\,448\,30 \cdot 10^{-26} \frac{\text{J}}{\text{T}}$ $(2,2 \cdot 10^{-8})$	Muon magnetic moment

Name	Numeric value and <i>rel. SD</i>	Remarks
σ_B	$5,670\,374\,41\dots \cdot 10^{-8} \frac{W}{m^2 K^4}$ <i>(exact)</i>	Stefan-Boltzmann constant $\sigma_B = \frac{2\pi^5 k^4}{15 h^3 c^2}$
Φ	1,618 033 988 749 894 8...	Golden ratio $\Phi = \frac{1}{2}(1 + \sqrt{5})$
Φ_0	$2,067\,833\,848\dots \cdot 10^{-15} V\ s$ <i>(exact)</i>	Magnetic flux quantum $\Phi_0 = \frac{\hbar}{2e}$
ω (75)	$7,292\,115 \cdot 10^{-5} \text{ rad/s}$ <i>($2 \cdot 10^{-8}$)</i>	Angular velocity of the Earth according to WGS84 (see footnote 44 on p. 133).
$-\infty$	$-\infty$	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.
∞	∞	

Note these constants will appear in routines with a heading # character (e.g. # c) as shown in the OM, Section 3.

A few more constants with unknown accuracy, not stored in CONST:

Numeric value	Remarks
$1,12 \cdot 10^4 \text{ m/s}$	Escape speed from Earth
$1,217 \text{ kg/m}^3$	Air density at standard conditions
331 m/s	Speed of sound at standard conditions
$28,97 \cdot 10^{-3} \text{ kg/mol}$	Molar mass of air
1370 W/m^2	Solar constant (average incident power at the mean distance of Earth to the Sun outside the atmosphere)
$1,62 \text{ m/s}^2$	Gravity acceleration on the Moon

Some more values found in nature are known with up to one or two digits precision only – they may be helpful for quick estimates nevertheless:

Radius of an atomic nucleus	$\sim 10^{-15}$ m
Radius of an atom ⁴⁵	$\sim 10^{-10}$ m
Radius of our home galaxy	$\sim 10^5$ l.y. $\approx 10^{21}$ m
Radius of the observable universe ⁴⁶	$\approx 45 \times 10^9$ l.y. $\approx 4.3 \times 10^{26}$ m
Number of stars in our home galaxy ⁴⁷	$\sim 2 \times 10^{11}$
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 ⁴⁸	$\sim 10^{57}$
Amount of atoms in observable universe	$\sim 10^{80}$
Baryonic mass in observable universe	$\sim 10^{53}$ kg

⁴⁵ 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!

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).

⁴⁶ For more information, please see https://en.wikipedia.org/wiki/Observable_universe (or https://de.w.../Beobachtbares_Universum or https://fr.w.../Univers_observable or 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 explaining the difference between the radius printed above and the naïve assumption of 13.8×10^9 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.

⁴⁷ 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? (Calculate yourself, then feel free looking up footnote 52 for an estimate.)

⁴⁸ You can take this number for the amount of atoms in our solar system as well – the stuff beyond the sun is neglectable here.

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.

Unit Conversions

Your *WP 43S* features 14 angular conversions provided in $\underline{4\rightarrow}$ (cf. p. 130) and 94 unit conversions in $\underline{U\rightarrow}$. The structure of $\underline{U\rightarrow}$ 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. 126f for more details of the structure.

The seven basic *SI* units are: *kelvin, meter, kilogram, second, ampere, mol*, and *candela* (the latter measuring *luminous intensity*⁴⁹). Beyond these and products or powers of them, knowledge of some *SI derived units* carrying special names are helpful. Each conversion contained in $\underline{U\rightarrow}$ either begins or ends in one of these *SI* units:

⁴⁹ Translator's note for German readers: This means *Lichtstärke*. On next page, *luminous flux* translates to *Lichtfluss* or *Lichtstrom* and *luminance* to *Leuchtdichte*.

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>	$1 \text{ W} = 1 \text{ V A} = 1 \text{ J/s}$
Electric potential	<i>volt</i>	$1 \text{ V} = 1 \text{ W/A}$
Charge	<i>coulomb</i>	$1 \text{ C} = 1 \text{ A s}$
Capacitance	<i>farad</i>	$1 \text{ F} = 1 \text{ C/V} = 1 \text{ As/V}$
Conductance	<i>siemens</i>	$1 \text{ S} = 1 \text{ A/V}$
Resistance	<i>ohm</i>	$1 \Omega = 1 \text{ V/A}$
Magnetic flux	<i>weber</i>	$1 \text{ Wb} = 1 \text{ V s}$
Magnetic flux density	<i>tesla</i>	$1 \text{ T} = 1 \text{ Wb/m}^2 = 1 \text{ Vs/m}^2$
Inductance	<i>henry</i>	$1 \text{ H} = 1 \text{ Wb/A} = 1 \text{ Vs/A}$
Frequency	<i>hertz</i>	$1 \text{ Hz} = 1/\text{s}$
Absorbed dose	<i>gray</i>	$1 \text{ Gy} = 1 \text{ J/kg}$
Plane angle	<i>radian</i>	$1 \text{ rad} = 1 \text{ m/m}$
Solid angle	<i>steradian</i>	$1 \text{ sr} = 1 \text{ m}^2/\text{m}^2$
Luminous flux	<i>lumen</i>	$1 \text{ lm} = 1 \text{ cd sr}$
Luminance, illuminance	<i>lux</i>	$1 \text{ lx} = 1 \text{ cd/m}^2$

For talking about inputs and results, knowing also the symbols and names of SI prefixes is beneficial. They cover 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}
μ	micro-	10^{-6}
n	nano-	10^{-9}
p	pico-	10^{-12}
f	femto-	10^{-15}
a	atto-	10^{-18}

All the conversions featured in $\text{U}\rightarrow$ and $\text{A}\rightarrow$ 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 for better visibility.

Softkey	Calculation	Remarks	Branch
$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	$\times 1,8 + 32$	$\theta [^{\circ}\text{C}] = (T [\text{K}] + T_0)$ and $t [^{\circ}\text{F}] = (t [^{\circ}\text{R}] + T_0 \times 1,8)$	$\text{U}\rightarrow$
$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$	$- 32) / 1,8$		
$\text{acre} \rightarrow \text{m}^2$	$\times 4\,046,86$	These <i>acres</i> are based on ' <i>international feet</i> ', see below	$\text{U}\rightarrow$ f A:
$\text{acre}_{\text{US}} \rightarrow \text{m}^2$	$\times 4\,046,87$	These <i>acres</i> are based on ' <i>U.S. survey feet</i> ', see below	
$\text{atm} \rightarrow \text{Pa}$	$\times 1,013\,25 \times 10^5$	Atmospheres	$\text{U}\rightarrow$ F&p:
$\text{au} \rightarrow \text{m}$	$\times 1,495\,98 \times 10^{11}$	Astronomic units	$\text{U}\rightarrow$ x:
$\text{barrel} \rightarrow \text{m}^3$	$\times 0,158\,987$	(U.S.) barrels of oil, abbr. bbl	$\text{U}\rightarrow$ f V:
$\text{bar} \rightarrow \text{Pa}$	$\times 100\,000$	$1 \text{ mbar} = 1 \text{ hPa}$	$\text{U}\rightarrow$ F&p:
$\text{Btu} \rightarrow \text{J}$	$\times 1\,055,06$	British thermal units	
$\text{cal} \rightarrow \text{J}$	$\times 4,186\,8$	Calories	$\text{U}\rightarrow$ E:

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

Softkey	Calculation	Remarks	Branch
kg → carat	/ 0,000 2		
kg → cwt	/ 50,802 4		
kg → oz	/ 0,028 349 5		
kg → lb.	/ 0,453 592		
kg → sh.cwt	/ 45,359 2	1 t [(metric) ton] := 1000 kg	
kg → short ton	/ 907,185		
kg → stone	/ 6,350 29		
kg → ton	/ 1 016,05		
kg → tr.oz	/ 0,031 103 5		
lbf → N	× 4,448 22	Pounds force	
lbf·ft → Nm	× 1,355 82	Pounds force times feet	
lb. → kg	× 0,453 592	Pounds; 1 lb := 16 ounces	
ly → m	× 9,460 73×10 ¹⁵	Light years	
m ² → acre	/ 4 046,86	Square meters;	
m ² → acre _{us}	/ 4 046,87	1 a [are] := 100 m ² , 1 ha [hectare] := 10 000 m ² ,	
m ² → ha	/ 10 000	1 km ² = 100 ha = 10 ⁶ m ²	
m ³ → barrel	/ 0,158 987		
m ³ → floz _{UK}	/ 2,841 31×10 ⁻⁵		
m ³ → floz _{US}	/ 2,957 35×10 ⁻⁵	Cubic meters;	
m ³ → gl _{UK}	/ 4,546 09×10 ⁻³	1 liter := 1 dm ³ = 10 ⁻³ m ³ , 1 ml [milliliter] = 1 cm ³	
m ³ → gl _{US}	/ 3,785 42×10 ⁻³		
m ³ → qt.	/ 1,136 52×10 ⁻³		
mi. → m	× 1 609,344	1 mile := 1 760 yards	
mmHg → Pa	× 133,322	See Pa below.	

Softkey	Calculation	Remarks	Branch
$m \rightarrow au$	/ $1,495\ 98 \times 10^{11}$		
$m \rightarrow \text{fathom}$	/ 1,828 8		
$m \rightarrow \text{ft.}$	/ 0,304 8		
$m \rightarrow \text{in.}$	/ 0,025 4		
$m \rightarrow ly$	/ $9,460\ 73 \times 10^{15}$		
$m \rightarrow \text{mi.}$	/ 1 609,344	<i>Meters</i>	[U→] [x:]
$m \rightarrow \text{nmi.}$	/ 1 852		
$m \rightarrow pc$	/ $3,085\ 68 \times 10^{16}$		
$m \rightarrow \text{point}$	/ $352,778 \times 10^{-6}$		
$m \rightarrow \text{survey foot}_{\text{US}}$	/ 0,304 801		
$m \rightarrow \text{yd.}$	/ 0,914 4		
$\text{nmi.} \rightarrow m$	x 1 852		<i>Nautical miles</i>
$\text{Nm} \rightarrow \text{lbf}\cdot\text{ft}$	/ 1,355 82		[U→] [▼]
$N \rightarrow \text{lbf}$	/ 4,448 22		[U→] F&p:
$oz \rightarrow kg$	x 0,028 349 5		[U→] [m:]
$Pa \rightarrow atm$	/ $1,013\ 25 \times 10^5$	<i>Pascals;</i> 1 hPa = 1 mbar	
$Pa \rightarrow \text{bar}$	/ 100 000		
$Pa \rightarrow \text{in.Hg}$	/ 3 386,39	For all real world purposes (i.e. within experimental errors), <i>torr</i> are equivalent to <i>millimeters of Hg</i> . Possible differences are merely calculatory. Please check also the document linked in footnote 50 on next page.	[U→] F&p:
$Pa \rightarrow \text{mmHg}$	/ 133,322		
$Pa \rightarrow \text{psi}$	/ 6 894,76		
$Pa \rightarrow \text{torr}$	/ 133,322		
$pc \rightarrow m$	x $3,085\ 68 \times 10^{16}$	<i>Parsecs</i>	[U→] [x:]
$\text{point} \rightarrow m$	x $352,778 \times 10^{-6}$	1 (<i>typographical point</i>) := 1/72 inch	

Softkey	Calculation	Remarks	Branch
power ratio → dB	$10 \lg(p_1/p_2)$		
psi → Pa	× 6 894,76	Pounds per square inch	F&p:
qt. → m³	× 1,136 52 × 10⁻³	1 (Imperial) quart := 40 (Imp.) fluid ounces	
short cwt → kg	× 45,359 2	1 short hundredweight := 100 lbs	
short ton → kg	× 907,185	1 short ton := 2000 lbs	m:
stone → kg	× 6,350 29		
survey foot _{US} → m	× 0,304 801	1 U.S. survey foot := $\frac{1200}{3937}$ m	x:
s → year	/ 31 556 952		
ton → kg	× 1 016,05	1 Imperial ton := 200 cwt	m:
torr → Pa	× 133,322	See Pa above.	F&p:
tr.oz → kg	× 0,031 103 5	Troy ounces	m:
Wh → J	× 3 600	Watt-hours	E:
W → hp _E	/ 746	Watts	
W → hp _M	/ 735,499		P:
W → hp _{UK}	/ 745,700		
yd. → m	× 0,914 4	1 yard := 3 feet	x:
year → s	× 31 556 952	= $365,242\ 5 \times 24 \times 60^2$	

Two more, really simple conversions (hence not included in U→):⁵⁰

- $1 \text{ barn} = 1 \text{ b} = 10^{-28} \text{ m}^2$ for cross sections in nuclear and particle physics (note the name),

⁵⁰ Find even more special units (including many outdated and weird ones) in this 90-page guide of NIST: <https://physics.nist.gov/cuu/pdf/sp811.pdf>. It also tells comprehensively how to deal with them.

- $1 \text{ tex} = 10^{-6} \text{ kg/m}$ for yarn density in the textile industry.

...	Remarks
DEG→	Takes an integer or <i>real</i> ⁵¹ x as an angular input in <i>decimal</i> or <i>sexagesimal degrees</i> , resp., and converts it to the current <i>ADM</i> .
D.MS→D	Takes an integer or <i>real</i> ⁵¹ x as an angular input in <i>sexagesimal degrees</i> (formatted dddd.dd.mmmsshh) and converts it to an <i>angle in decimal degrees</i> (corresponding to the old command H.MS→H).
D→R	Takes an integer or <i>real</i> ⁵¹ x as an angular input in ... <i>radians</i> . ⁵²
D→D.MS	... <i>sexagesimal degrees</i> (corresponding to the old command H→H.MS).
GRAD→	... <i>grades/gon</i> ...
MULπ→	Takes an integer or <i>real</i> ⁵¹ x as an angular input in ... <i>multiples of π</i> ... to the current <i>ADM</i> .
RAD→	... <i>radians</i> . ⁵² ...
R→D	Takes an integer or <i>real</i> ⁵¹ x as an angular input in <i>radians</i> and converts it to <i>decimal degrees</i> (equaling the old command R→D). ⁵²

⁵¹ If x is neither integer nor *real* (i.e. neither DT 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.

⁵² Note that *real/angles given in radians* cannot represent full circles (or multiples of them, as well as simple fractions of π like $\pi/2$, $\pi/3$, $\pi/4$, etc.) exactly but with an accuracy of 34 digits 'only'. If you want to avoid rounding errors caused by that but must keep π in play, *multiples of π* may be a better choice for the *angular display mode* here.

Note that large numeric inputs in trigonometric functions are reduced to values between $-\pi$ and $+\pi$ before calculating (as mentioned in *Section 2* of the OM). Such reductions may easily introduce inaccuracies when *radians* are used – all other angular units are uncritical in this aspect.

 ...	Remarks
→DEG	Takes an integer (<i>DT 1</i>) or <i>real</i> (<i>DT 2</i>) x as an angular input in the current <i>ADM</i> and converts it to <i>decimal degrees, sexagesimal degrees, grades/gon, multiples of π, or radians</i> , respectively. ⁵²
→D.MS	
→GRAD	If x is a tagged <i>real</i> (<i>DT 4</i>), on the other hand, this information is used in conversion (e.g. if $x = 1.5\pi$ then →GRAD will return 300° regardless of current <i>ADM</i>).
→MULπ	
→RAD	If x is neither of <i>DT 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.⁵³

⁵³ Solution of footnote 47: In this very simple model, an average volume of 150 (*l.y.*)³ is available for each star in our galaxy, i.e. a sphere with a diameter of 6 *l.y.*. For estimating the consequences of a collision of galaxies, take into account that star density decreases significantly from the galactic center outwards.

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 **α** . 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**. 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 STO CFG STOS STO+ STO- STOx STO/ STO [†] STO [‡] t [‡] VIEW x [‡] x=? x≠? x≈? x<? x≤? x≥? x>? y [‡] z [‡] αENG? αPOS? αRL αRR αSL α→x [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 Π _n Σ _n		Program label
GTO.	Number of program step	Label

Operations requiring one trailing parameter	Numeric parameter	Alpha par.
INDEX MVAR M.DIM M.EDIN 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 st to 4 th 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 *DT* of the contents 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 <i>DT</i>	Operation(s)	Output <i>DT</i>	Output <i>registers</i>
1	$1/x$ $\sqrt[3]{x}$ AGM ALL cos ENG erf erfc e x FIX f' f'' gd gd $^{-1}$ J $_y(x)$ LN LN β LNL \log_{10} \log_2 \log_{xy} MANT POISS... SCI sin tan W $_m$ W $_p$ W $^{-1}$ \sqrt{y} $\beta(x,y)$ Γ_{xy} γ_{xy} $\Gamma(x)$ Δ% →REAL % %MRR %T %Σ %+MG \sqrt{x} as well as all unit conversions and all orthogonal polynomials	2 ⁵⁴	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

⁵⁴ The functions printed on yellow background will return *long integers* (*DT* 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
3	→INT	10	X
	ABS CROSS IM RE x 4	2	X
	CX→RE	2	X, Y
4	SLVQ	2 or 3	X, Y, Z
	cos sin tan	2	X
	→HR	2	X
5	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 ₄	4	X
9	M.DIM?	1	X, Y
	ENORM	2	X
	DET DOT M	3	X
	ABS IM RE ROUND I x	8	X
10	SIGN	1	X
	e ^x LN LOG _x y →REAL	2	X
	x→α	7	X

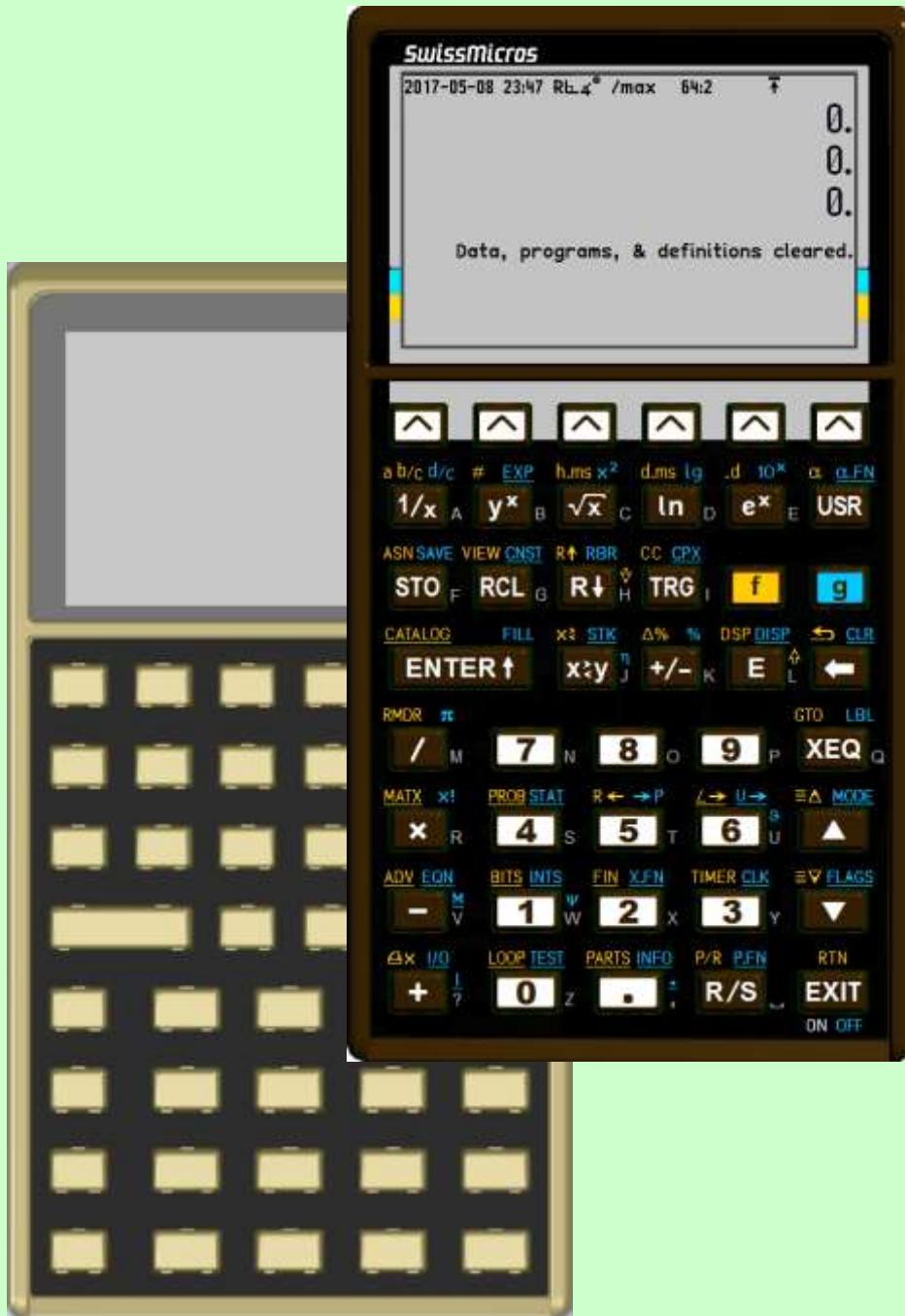
APPENDIX A: HARDWARE

- Dimensions: wedge-shaped 77 mm × 144 mm × 13 mm or 8 mm (see a picture on next page).
- Mass: 180 g incl. battery.
- LCD: monochrome high contrast (14:1) transflective memory display with an active area of 58.8 mm × 35.3 mm, 400 × 240 quadratic pixels, and dot pitch 147 µm.
- Processor: low power ARM Cortex-M4F (*STMicroelectronics STM32L476*) incl. real-time clock running at 25 MHz on battery power or 80 MHz when power is supplied through USB (see below; look up <https://www.st.com/en/evaluation-tools/32l476gdiscovery.html> for the development board used by SwissMicros).
- Memory: 1 MB *FM*, 128 kB *RAM* (see App. B on pp. 160ff), 32 MB additional external *FM* on a QSPI chip; user *RAM* is some 75 kB, user *FM* is 32 MB.
- Power supply: 3 V by 1 CR2032 lithium coin cell (battery life up to 3 years); optionally powered through USB port; typical average currents drawn: power on & busy: 4.2 mA; idle: 0.1 mA; power off: 3 µA.
- Buzzer: piezo-electric with 4 kHz resonance frequency, tunable from 1 Hz up to 20 kHz in steps of 1 Hz.
- Connectivity: infrared port compatible with *HP-82240A/B* printers,
micro-USB-B port connecting to a host computer as external mass storage device.
- Self-test: initiated by **xxx**
- Keyboard overlays: Three short slots on either side of the keyboard are provided in the calculator frame for easily fixing your overlay sheets with your personal layouts printed on. See App. F for more (on p. 213). Look up Section 6 of the OM for inspiration.



Seven pictures of the hardware are displayed here and on the 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 key-
board side of the *PCB*
carrying the switching
domes, carefully re-
lease the *LCD* connec-
tion **A**, then unfasten
the two Philips bolts **B**
pictured left and below.
All 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 *DTs* 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*,⁵⁵
- two *DTs* for two kinds of *menus*,
- a *DT* of variable length for storing *configurations* (see RESET, 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 *DT*:

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 name				0	0	data information ⁵⁶				<i>DT</i> (as specified below) – 1					

⁵⁵ Up to 256 *flags* would be possible in total. We need far less *system flags* so far.

⁵⁶ E.g. base of a *short integer*, angular unit for an *angle*.

DT number and meaning		Size [bytes]
1	\mathbb{Z} Long integer ⁵⁷	$\geq 4 + 2 + 4 = 10 \dots 422$
2	\mathbb{R} Real number ⁵⁸ (<i>real34</i>)	$4 + 16 = 20$
3	\mathbb{C} Complex number (<i>complex34</i>)	$4 + 2 \times 16 = 36$
4	Angle ⁵⁹ (<i>angle34</i>)	$4 + 16 = 20$
5	Time ⁶⁰	$4 + 16 = 20$
6	Date ⁶¹	$4 + 16 = 20$
7	Alphanumeric string (a.k.a. <i>text string</i> ; each character requires 16 bits) ⁶²	$4 + 2 + n \times 2$
8	Real matrix (featuring <i>n</i> rows and <i>m</i> columns)	$4 + n \times m \times 16$
9	Complex matrix (featuring <i>n</i> rows and <i>m</i> columns)	$4 + n \times m \times 32$
10	Short integer ⁶³	$\leq 4 + 8 = 8 \text{ or } 12$
11 ... 13	n/a	n/a
14	Constant ⁶⁴	$4 + 0 = 4$

⁵⁷ This DT 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.

⁵⁸ Deviating from the WP 34S, standard *reals* on your WP 43S feature 128 bits and 34-digits precision. See the chapter after next.

⁵⁹ A tagged *angle* is stored as a *real* number, just with a specific header.

⁶⁰ 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.

⁶¹ 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.

⁶² 2 bytes are for the size (in *bytes*) of the string following, including the trailing zero. The size must be even.

⁶³ This DT is for computer science problems. Most probably, such a storage space will be either 4 + 4 or 4 + 8 bytes long.

⁶⁴ A pointer is sufficient here regardless of the precision of the constant itself.

DT number and meaning	Size [bytes]	
15 Extended precision <i>real</i> (39 digits, exclusively for internal use, not exposed to the user) ⁶⁵	4 + 32 =	36
16 <i>Label</i> (each character requires 16 <i>bits</i>)	4 + 7 × 2 =	18
17 System integer (for internal use only)	4 + 4 =	8
18 System and user flags (128 <i>flags</i> each)	4 + 2 =	6
19 User-created <i>menu</i> (limited to 1 <i>view</i>)	4 + 18 × 7 × 2 =	256
20 Predefined <i>menu</i> (featuring <i>n</i> <i>views</i>)	4 + n × 18 × 14	
21 Calculator <i>configuration</i> (as stored by STOCFG)	4 + M⁶⁶	
22 Program step, may be stored as <i>text string</i> (7)	4 + M⁶⁷	
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>text string</i> (7)	4 + M	
25 <i>Directory</i> (proposed by P. 2012-12)	4 + M	

DTs 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 *DT* 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.

⁶⁵ There are also even longer (i.e. more precise) *reals* used in internal computations. See further below.

⁶⁶ The size will vary according to the number of user assignments being part of the *configuration* (see Section 6 of the OM).

⁶⁷ 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 *GP registers* unlike they did on HP's pocket calculators. Their contents can be recalled individually using their names (cf. pp. 57 and 89f).

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 $\text{CL}\Sigma$.

SAVE and LOAD...

SAVE stores the calculator *configuration*, all *flags* and *registers* (including the *summation registers*, if allocated), and all programs present in *RAM*. **It stores all these data in FM**⁶⁸ Thus, this storage will survive **RESETs**, even hard ones via the RESET button, and is battery fail safe. So you can use it as on-board backup of your programs and data, for instance.

The various flavours of LOAD allow resuming calculator operation after such events, according to your requests:

- LOADSS recovers the *configuration* only, similar to RCLCFG,⁶⁸
- LOAD Σ recalls just the *summation registers*,
- LOADR all the numbered *registers*,
- LOADP the programs, and
- LOAD recalls the entire data set stored by SAVE at once.

Turn to the *I/O* for more information about these individual commands.

⁶⁸ Note STO CFG stores the calculator *configuration* in a *register* or variable (i.e. in *RAM*) instead.

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. Its 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.⁶⁹ Results should be accurate within $\pm 1 \times 10^{-33}$ per operation (e.g. **n** **1/x** **2n** **x** returns a plain 2 for all primes < 500 – except **7** **1/x** **1** **4** **x** **2** **-** returning 1×10^{-33} , and for 67 and 83 the respective results are -1×10^{-33}). Note that rounding errors due to calculating with a finite number of digits will accumulate (as other errors do) – see two examples in the footnote here.

⁶⁹ Actually this is the minimum; some modulo calculations for the trigonometric functions are performed with more than a thousand digits to avoid cancellation.

Test any calculator you may use with this: $729^{33.5} /_{3^{201}} - 1$. Your WP 43S will return 0.

There is a quasi standard to find out about processors, firmware, and assess accuracy of calculators – compute $\arcsin\{\arccos[\arctan(\tan\{\cos[\sin(9^\circ)]\})]\}$ although this formula is mathematical nonsense. An ideal number cruncher featuring infinite internal precision would return exactly 9° or equivalent (see e.g. [https://www.wolframalpha.com/input/?i=arcsin\(arccos\(arctan\(tan\(cos\(sin\(pi/20\)\)\)\)\)\)](https://www.wolframalpha.com/input/?i=arcsin(arccos(arctan(tan(cos(sin(pi/20)))))))). Real calculators (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 – $6,2465 \cdot 10^{-29}$ for $\arcsin\{real(\arccos\{real[\arctan(\tan\{\cos[\sin(9^\circ)]\})]\})\}$.

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 **x²** 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 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 **x²** 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.

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,⁷⁰ 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 (128th) 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 twelve are reserved for the exponent (<4096). For the top five bits (below the sign bit) it becomes complicated – following the standard *IEEE 754*. If these five bits read...
 - 00ttt,
01ttt, or
10ttt then ttt takes the leftmost digit of the *significand* (0 – 7), and the top two bits will be the most significant bits of the exponent;
 - 11uut then t will be added to 1000₂ and the result (8₁₀ or 9₁₀) will represent the leftmost digit of the *significand*.

⁷⁰ 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.

If **uu** reads **00**, **01**, or **10₂** then these will represent the two most significant bits of the exponent. If it reads **11₂**, 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_{10} 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 $-6176 + 34 = -6142$, DT2 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		0010 0010 0000 1000 00	6176
111.111 $(111\ 111 \times 10^{-3})$	22 07 40 00 00 00 00 00 00 00 00 00 00 01 bc 6f	06f 06f	0010 0010 0000 0111 01	6173
-123.000 123 $(-123\ 000\ 123 \times 10^{-6})$	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
9.99×10⁹⁹ (999×10^{97})	22 20 40 00 00 00 00 00 00 00 00 00 00 00 03 e7		0010 0010 0010 0000 01	6273

Floating point number	Hexadecimal value stored	Bottom bits in groups of 10	Top 18 bits in binary notation	SE
1×10^{-99}	21 ef 40 00 00 00 00 00 00 00 00 00 00 00 00 01		0010 0001 1110 1111 01	6077
-1×10^{-6143}	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 1.999 999 999 999 999 999 999 999 999 999 999 999 999 999 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
9.999 999 999 999 999 999 999 999 999 999 999 $\times 10^{6144}$ ($= 9\ 999 \dots 999\ 999$ $\times 10^{6110})$	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

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

Floating point 'number'	Hexadecimal value stored	Top 8 bits in binary notation
∞	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 **∞ 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 69 on pp. 164f. And errors accumulate as explained in footnote 42 on p. 131.

As mentioned above, some internal calculations are executed in “*internal high precision*”, employing 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!

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 FF₁₆ in unsigned mode for WSIZE 64, equivalent to some 1.8×10^{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. 3.3×10^{24}), however, NEXTP cannot find primes anymore but ‘probable primes’ only (cf. p. 60). If you want to apply NEXTP or PRIME? above this limit, consult appropriate reference literature.⁷¹
- Trigonometric functions actually operate between $+\pi$ and $-\pi$ (or their equivalents) exclusively. The necessary modulo 2π reduction 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. 71).

⁷¹ See 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 9×10^{-16} . With some care and a lot of time, NEXTP can find ‘probable primes’ for long integers up to some 6.3×10^{1001} , and PRIME? will check them and their neighbouring numbers (USB-power recommended) but you cannot see most of their 1002 digits directly. You can, however, get a 296-digit prime within some 40 seconds – and read it entirely.

- **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 ∞ , 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 ∞ , 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 ∞ , 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 π (or their equivalents) only.
- **5:** **xxx**
- **6:** **xxx**
- **10:** The maxima are as specified for input above.

Further output limitations for *data type* ...

- **21:** *Configurations* cannot be displayed. Thus, if any register or variable containing such data is on screen, **Configuration data** is shown instead of its content.

What Happens when Changing Word Size?

WSIZE affects the allocated stack registers and **L** only: **xxx**

Special Results

Throughout this chapter, SPCRES is presumed to be set. Thus, infinities and non-numeric results are legal – no error messages will be thrown if such results happen to occur (cf. p. 168).⁷²

The following monadic functions will return either ∞ , $-\infty$, or NaN under the conditions stated below:

Input x	Operation(s)	Output for \mathbb{R} lit
-1.	artanh	
0 or 0.	In , Ig , $\text{lb } x$	$-\infty$
0.	$\frac{1}{x}$	
1.	artanh	∞
0 or 0.	$\Gamma(x)$	
$\text{Re}(x) < 1$	arcosh	
$ \text{Re}(x) > 1$	arccos , arcsin , artanh	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 , $\sqrt[3]{x}$	$-\infty$
	arctan	-90° or equivalents
	\tanh	-1.
	$\frac{1}{x}$, e^x , 10^x , 2^x , sinc	0.
	x^2 , arsinh	∞

⁷² 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.

Input x	Operation(s)	Output for ℝ lit
$-\infty$	arcosh	NaN
$-\infty \leq x < 0$	ln , lg , $\text{lb } x$	NaN
∞	$\frac{1}{x}$, sinc	0.
	\tanh	1.
	\arctan	90. [°] or equivalents
	ln , e^x , x^2 , \sqrt{x} , lg , 10^x , $\text{lb } x$, x^3 , $\sqrt[3]{x}$, \sinh , \cosh , arsinh , arcosh	∞
$-\infty$ or ∞	\cos , \sin , \tan , artanh	NaN

For dyadic functions, we combined the respective tables:

Input y	Input x	Op.(s)	Output for ℝ lit
∞	$x \neq -\infty$	+	∞ ⁷³
$-\infty$	$x \neq \infty$		$-\infty$ ⁷³
$-\infty$	∞	+	NaN
∞	$x \neq \infty$		∞ ⁷⁴
$-\infty$	$x \neq -\infty$	-	$-\infty$ ⁷⁴
$-\infty$	$-\infty$		NaN
∞	∞		NaN
∞	$x > 0$	x	∞ ⁷³
$-\infty$	$x < 0$		NaN
∞	$x < 0$	x	$-\infty$ ⁷³
$-\infty$	$x > 0$		NaN
0 or ∞	$-\infty$ or ∞	x	NaN

⁷³ Swapping x and y will return the same result here.

⁷⁴ Swapping x and y will return this result times -1.

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

The functions printed on light yellow background in the three tables above will also return NaN (or NaN+i×NaN) with complex results allowed (i.e. CPXRES set). Others will change their output when 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 ⁷⁵		r(x)	$\varphi(x)$	Op.	Output for C lit
Re(x)	Im(x)				
$-\infty$	—	—	—		
$-\infty$	0	∞	180°	\sqrt{x}	$\infty \notin 90^\circ = 0 + i \times \infty$
$-\infty$	0	∞	180°		$\infty \notin 360^\circ = \infty + i \times 0.$
0.	∞	∞	90°	x^2	$\infty \notin 180^\circ = -\infty + i \times 0.$
$-\infty$	—	—	—		$-\infty$
$-\infty$		∞			$-\infty + i \times 0.$ ⁷⁶
-10^{999}	0	10^{999}	180°	$\sqrt[3]{x}$	$1 \times 10^{333} \notin 60^\circ = 0.5 \times 10^{333} + i \times 0.866\ 025\ 404 \times 10^{333} = 5 \times 10^{332} (1 + i \times \sqrt{3})$
$-\infty$	—	—	—	x^3	$-\infty$
$-\infty$	0	∞	180°		$-\infty + i \times 0.$
$-\infty$	—	—	—		
$-\infty$	0	∞	180°	\ln	$\infty + i \times 3.141\ 592\ 65\dots = \infty + i\pi$
$-\infty$	0	∞	180°		$\infty + i \times 2.356\ 194\ 49\dots = \infty + i^{3\pi/4}$
0.	∞	∞	90°	\ln	$\infty + i \times 1.570\ 796\ 32\dots = \infty + i^{\pi/2}$
∞	∞	∞	45°		$\infty + i \times 0.785\ 398\ 16\dots = \infty + i^{\pi/4}$

⁷⁵ 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).

⁷⁶ 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 in your WP 43S.

Input ⁷⁵ Re(x)	Im(x)	r(x)	$\varphi(x)$	Op.	Output for \mathbb{C} lit
∞	—	—	—	In	∞
∞	0	∞	0°	In	$\infty + i \times 0.$
∞	$-\infty$	∞	-45°	In	$\infty - i \times 0.785\ 398\ 16\dots = \infty - i \pi/4$
0.	$-\infty$	∞	-90°	In	$\infty - i \times 1.570\ 796\ 32\dots = \infty - i \pi/2$
$-\infty$	$-\infty$	∞	-135°	In	$\infty - i \times 2.356\ 194\ 49\dots = \infty - i^{3\pi/4}$
0.	0.	0.	0.	In	$-\infty + i \times 0.$
0.	—	—	—	In	$-\infty$
$-\infty$	0	∞	180°	e^x	$0. + i \times 0.$
-10^{999}	10^{999}	10^{999}	135°	e^x	$0. + i \times 0.$
$-\infty$	∞	∞			$\text{NaN} + i \times \text{NaN}$
0.	∞	∞	90°	e^x	$\text{NaN} + i \times \text{NaN}$
∞	∞	∞	45°		$\infty + i \times 0.$
∞	0	∞	0°	e^x	$\text{NaN} + i \times \text{NaN}$
∞	$-\infty$	∞	-45°		$\text{NaN} + i \times \text{NaN}$
0.	$-\infty$	∞	-90°	e^x	$\text{NaN} + i \times \text{NaN}$
$-\infty$	$-\infty$	∞	$0. + i \times 0.$		
-10^{999}	-10^{999}	10^{999}	-135°	e^x	$0. + i \times 0.$

Computation of lg and $\text{lb } x$ is derived from 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 DT
22 on p. 162. xxx

DRAFT

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} , $\Sigma+$, $\Sigma-$, σ , \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, one of the following messages will be thrown (listed alphabetically – *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. ⁷⁷
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 I , J , or K is out of valid range for the distribution function called (e.g. LGNRM is called with $j < 0$).
Flash memory is full	23	Delete a program from <i>FM</i> to regain space.
Flash memory is write protected	19	There was an attempt to edit or delete program steps in <i>FM</i> . See PRCL and PSTO to circumvent.

⁷⁷ 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.

	EC	Explanations, countermeasures and examples
Function to be coded for that data type	30	Functions may not be coded yet during FW development.
Illegal digit in integer input for this base	9	{10} E.g. 2 in binary or 9 in octal input. Will be thrown as soon as input is closed.
Illegal input data type for this operation	24	Convert what is necessary, if possible (see also “operation is undefined in this mode”). But this error may also appear in attempts to calculate with a configuration.
Input data types do not match	31	Attempt to operate on different DTs (e.g. for a Boolean operation: real AND short integer).
Input is too long	10	{7} Keyboard input is too long for the buffer.
Invalid or corrupted data	18	Thrown when there is a checksum error either in FM or as part of a serial download. Also thrown if a FM segment is otherwise not usable.
Item to be coded	29	Functions may not be coded yet during FW development.
I/O error	17	See Section 3 of the OM.
Matrix mismatch	21	{8, 9} <ul style="list-style-type: none"> • A matrix isn't square although it should be. • Matrix sizes aren't miscible.
No backup data found	35	Futile attempt to LOAD something from FM.
No root found	20	{2} The Solver did not converge.

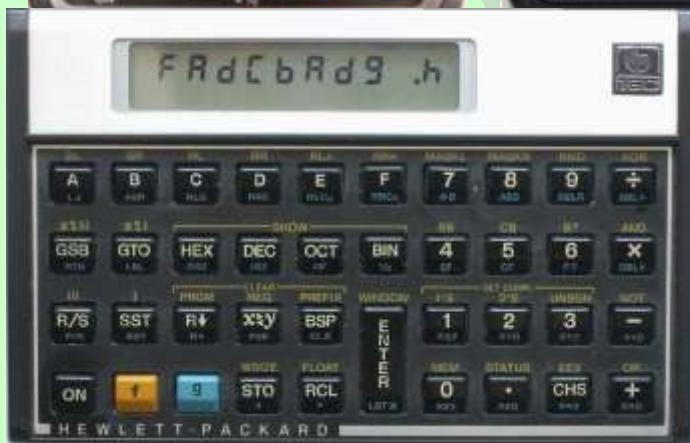
	EC	Explanations, countermeasures and examples
No such function	7	Thrown when calling a nonexistent function via XEQ α ... ENTER↑ (check for typos!) or running a routine containing a nonprogrammable command.
No such label found	6	Attempt to address an undefined label.
No summation data present	28	Attempt 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"> A number exceeds the valid range. This can be caused by specifying more decimals than 34, <i>word size</i> > 64, short integers $\geq 2^{64}$, <i>hours</i> or <i>degrees</i> > 9 000, invalid <i>dates</i> or <i>times</i>, denominators > 9 999, etc. 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. An R-operation (e.g. R-COPY) attempts accessing invalid <i>register</i> addresses.
Output would exceed 196 characters	33	{7} Maximum length of <i>alphanumeric strings</i> .
Overflow at $+\infty$	4	<p>{1, 2, 3, 8, 9} unless SPCRES is set</p> <ul style="list-style-type: none"> Division of a number > 0 by 0. Divergent sum or product or integral. Positive overflow (see p. 163).

	EC	Explanations, countermeasures and examples
Overflow at $-\infty$	5	{1, 2, 3, 8, 9} unless SPCRES is set <ul style="list-style-type: none"> • Division of a number < 0 by 0. • Divergent sum or product or integral. • Negative overflow (see p. 163).
Please enter a NEW name	26	Attempt to define a new variable or user <i>menu</i> with a <i>name</i> already in use.
RAM is full	11	May be caused by attempts to write too large routines, allocate too many variables, and the like (see pp. 160ff for the space required by different <i>DTs</i>). May happen also in program execution due to dynamic allocations (see Sect. 3 of the OM).
Singular matrix	22	{8, 9} <ul style="list-style-type: none"> • Attempt to use a LU decomposed matrix for solving a system of equations. • Attempt to invert a matrix which isn't of full rank.
Stack clash	12	STOS or RCLS attempts using <i>registers</i> that would overlap the <i>stack</i> (see Section 1 of the OM). Will happen with e.g. ssIZE8 set and STOS 93 .
This does not work with an empty string	34	{7} Self-explanatory.
This system flag is write protected	32	Self-explanatory.
Too few data points for this statistic	15	{2} A statistical calculation was attempted with too few data, e.g. <i>regression</i> or <i>standard deviation</i> for less than 2 points.
Undefined op-code	3	An instruction with an undefined operation code occurred. Should never happen – but who knows?

	<i>EC</i>	Explanations, countermeasures and examples
Word size is too small	14	{10} Input or <i>register</i> content is too great to be handled by the <i>word size</i> currently set.
	25	Left unused for <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. 130ff and the tables on pp. 168ff). 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 **C** 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.



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. 189, one for the *HP-21S* on p. 191, and another one for the *WP 34S* on p. 193. Functions newly introduced with *WP* calculators are compiled on pp. 197ff.

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*.

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

HP-42S	WP 43S	Remarks
AOFF	CF ALPHA	
AON	SF ALPHA	
ARCL	Disposable	Any register or variable can take a <i>text string</i> . Simply press RCL instead.
AROT	αRL or αRR	In <u>$\alpha.FN$</u>
ASHF	αSL	
ASINH	$arsinh$	In <u>EXP</u>
ASTO	Disposable	Any register or variable can take a <i>text string</i> . Simply press STO instead.
ATANH	$artanh$	In <u>EXP</u>
ATOX	$\alpha \rightarrow x$	In <u>$\alpha.FN$</u>
AVIEW	Disposable	Any register or variable can take a <i>text string</i> . Simply press VIEW instead.
BASE	INTS or BITS	
BASE+	Disposable	Your WP 43S executes these arithmetic commands automatically for <i>short integer</i> inputs.
BASE-		
BASE \times		
BASE \div		
BASE $+\!-\!$		
BINM	Disposable	Press # 2 for converting any closed integer number or integer part in x to binary.
BIT?	BS?	In <u>BITS</u>
BST	 ()	Shortcut works if no <i>multi-view menu</i> is open.
CLA	0 STO K	
CLD	Disposable	Any keystroke will clear <i>temporary information</i> .
CLEAR	CLR	
CLKEYS	n/a	See <i>Section 6</i> of the OM.
CLRG	CLREGS	In <u>CLR</u>
CLST	CLSTK	Press 0 FILL in run mode.

HP-42S	WP 43S	Remarks
CLV	See remark	Variables are cleared as specified in <i>Section 6</i> of the OM.
COMPLEX	CC	You can also enter <i>complex</i> numbers directly using CC as explained in <i>Section 2</i> of the OM.
CONVERT	L→ & PARTS	
CUSTOM	n/a	You can create as many <i>menus</i> as memory will hold – not only one CUSTOM menu. See <i>Section 6</i> of the OM.
DECM	Disposable	Any input featuring a 0 or an E is interpreted as a <i>real</i> (decimal) number.
DEL	n/a	Not featured. Too dangerous, in our opinion.
DELAY	DLAY	In <u>PRINT</u>
DELR	M.DELR	In <u>MATX</u>
DET	 M 	
DIM	M.DIM	
DIM?	M.DIM?	
EDIT	M.EDI	
EDITN	M.EDIN	
FCSTX	x	In <u>STAT</u>
FCSTY	y	
FNRM	ENORM	In <u>MATX</u> . Euclid is older than Frobenius.
GAMMA	Γ(x)	In <u>PROB</u>
GETKEY	KEY?	In <u>P.FN</u>
GETM	M.GET	In <u>MATX</u>
GROW	M.GROW	
HEXM	Disposable	Press # H for converting any closed integer number or integer part in x to hexadecimal.
H.MS+	Disposable	Your WP 43S executes the respective command automatically for sexagesimal times in x and y when + or - is pressed.
H.MS-		

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

HP-42S	WP 43S	Remarks
PRA		In <u>PRINT</u>
[PRGM]	[P/R]	
PRLCD		In <u>PRINT</u>
PROFF		
PROMPT	Disposable	Use , instead.
PRON		
PRP		
PRSTK		
PRUSR		
PRV		
PRX		Press .
PRΣ		In <u>PRINT</u>
PUTM	M.PUT	In <u>MATX</u>
PWRF	PowerF	In <u>STAT</u>
RAN	RAN#	In <u>PROB</u>
RND	ROUND	In <u>PARTS</u>
RNRM	RNORM	In <u>MATX</u>
ROTXY	RL, RLC, RR, and RRC	In <u>BITS</u>
RTN		Press .
SDEV	s	In <u>STAT</u>
SIZE	Disposable	There are 100 global <i>GP registers</i> always.
SLOPE	L.R.	In <u>STAT</u>
SOLVE	SLV	In <u>ADV</u>
SQRT		
[SST]	()	Shortcut works if no <i>multi-view menu</i> is open.
STR?	STRI?	In <u>TEST</u>

HP-42S	WP 43S	Remarks
TOP.FCN	Disposable	Obsolete – no top functions are overwritten.
TRACE	SF	
TRANS	[M] ^T	In <u>MATX</u>
UVEC	UNITY	In <u>MATX</u> and <u>CPX</u>
VARMENU	VARMNU	Truncated to 6 characters to fit the <i>menu</i> space.
VIEW		Press VIEW .
WMEAN	\bar{x}_w	In <u>STAT</u>
WRAP	M.WRAP	In <u>MATX</u>
XTOA	$x \rightarrow \alpha$	The conversion is done in X .
$X < 0?, X < Y?$	$x < ?$	In <u>TEST</u>
$X \leq 0?, X \leq Y?$	$x \leq ?$	
$X = 0?, X = Y?$	$x = ?$	
$X \neq 0?, X \neq Y?$	$x \neq ?$	
$X \geq 0?, X \geq Y?$	$x \geq ?$	
$X > 0?, X > Y?$	$x > ?$	
YINT	L.R.	In <u>STAT</u>
y^x		Press y^x .
ΣREG	Disposable	There are 100 global <i>GP registers</i> always.
$\Sigma REG?$		Statistical registers are separate.
$\rightarrow DEC$	$\rightarrow INT$ 10	Press # D
$\rightarrow HR$		Press .d
$\rightarrow H.MS$		Press h.ms ... for closed input.
$\rightarrow OCT$	$\rightarrow INT$ 8	Press # 8
$\rightarrow POL$		Press $\rightarrow P$.
$\rightarrow REC$		Press R↔ .
%CH	$\Delta\%$	Press $\Delta\%$.
		Cf. ISO 80000-2: "The symbol \div should not be used."

Corresponding Operations on HP-16C

The table for the functions of the *HP-16C* is sorted following its keyboard layout, 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	In <u>BITS</u>
[RR] , [RRn]	RR	
[RLC] , [RLCn]	RLC	
[RRC] , [RRCn]	RRC	
[÷]	/	(see also ISO 80000-2: "The symbol \div should not be used.")
[DBL÷]	DBL/	
[x_i(i)]	Superfluous	Any register may be used for indirection.
[x_i]		
SHOW HEX	n/a	
SHOW DEC		
SHOW OCT		
SHOW BIN		
[B?]	BS?	In <u>BITS</u>
[GSB]	XEQ	
[HEX]	# H	
[DEC]	# D	
[OCT]	# 8	
[BIN]	# 2	
[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 <u>FLAGS</u>
[I]	Dispensable	Any register may be used for indirection.

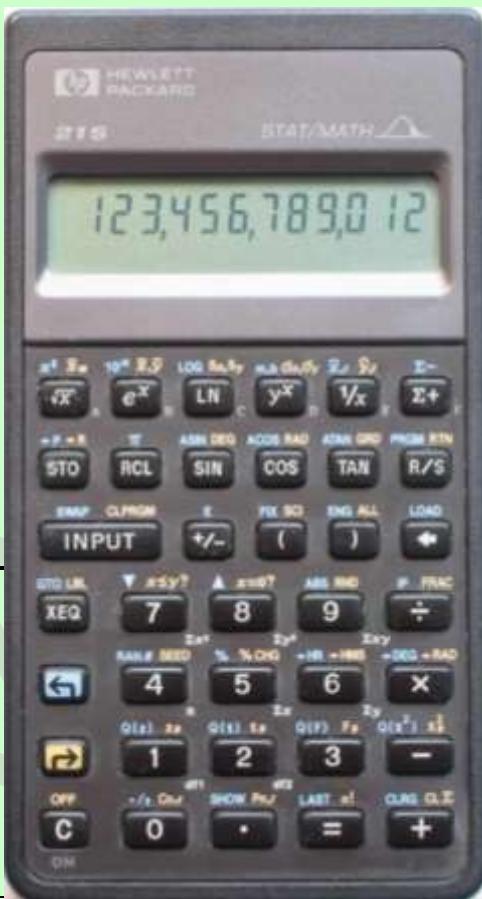
HP-16C	WP 43S	Remarks
CLEAR PRGM	CLP	In <u>CLR</u> . Note here is also CLPALL.
CLEAR REG	CLREGS	In <u>CLR</u>
CLEAR PREFIX	Dispensable	See Section 2 of the OM.
WINDOW	Dispensable	64 bits can be displayed in one row.
SET COMPL 1S	1COMPL	
SET COMPL 2S	2COMPL	In <u>MODE</u> and <u>BITS</u> . Note here is also SIGNMT.
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≤ ?	
x<0	x< ?	In <u>TEST</u> . Note far more tests are covered here.
x>y , x>0	x> ?	
FLOAT	FIX	In <u>DISP</u>
MEM	STATUS	In <u>FLAGS</u>
CHS		
<, >	Dispensable	64 bits can be displayed in one row.
LSTX	RCL L	
x≠y , x≠0	x≠ ?	In <u>TEST</u> . 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* follows the same rules as the one for *HP-16C*. It is, however, 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.

<i>HP-21S</i>	<i>WP 43S</i>	Remarks
[\bar{x}_w]	\bar{x}_w	
[\bar{x}, \bar{y}]	\bar{x}	
[s_x, s_y]	s	
[m.b]	L.R.	In STAT
[σ_x, σ_y]	σ	
[$\hat{x} \cdot r$]	r, \hat{x}	
[$\hat{y} \cdot r$]	r, \hat{y}	
[PRGM]	[P/R]	
[SWAP]	[x\leftrightarrowy]	
[CLPRGM]	CLP	In CLR
[INPUT]	Dispensable	This functionality is contained in ENTER. Also your <i>WP 43S</i> features a command called INPUT but this works in programs.
(,)	Dispensable	You can forget these keys in RPN.
[LOAD]	n/a	Loads predefined programs in the <i>HP-21S</i> . Also your <i>WP 43S</i> features a command called LOAD but this recalls data from backup.



HP-21S	WP 43S	Remarks
ABS	 x 	In <u>PARTS</u>
RND	ROUND	
FRAC	FP	
÷	/	Cf. ISO 80000-2: "The symbol \div should not be used."
SEED	SEED	In <u>PROB</u>
%CHG	$\Delta\%$	
Q(z)	Norm_e	In <u>submenus of PROB</u> . Note your WP 43S features the normal distribution for <u>arbitrary</u> μ and σ instead of the standardized one ($\mu=0$, $\sigma=1$). And the implementation of inverse distribution functions deviates on both calculators: your WP 43S calculates with the probability P while the HP-21S calculates with the error probability $Q = 1 - P$ as input (cf. the OM, Section 2). Labeling these functions on the HP-21S using the letter p may add some confusion.
z_p	Norm_e⁻¹	
Q(t)	t_e(x)	
t_p	t⁻¹(p)	
Q(F)	F_e(x)	
F_p	F⁻¹(p)	
Q(χ^2)	$\chi^2_e(x)$	
X²p	(χ^2)⁻¹	
C_{n,r}	COMB	In <u>PROB</u>
P_{n,r}	PERM	
LAST	RCL L	
=	Disposable	You can forget this key in <u>RPN</u> .
n!	x!	
CLRG	CLREGS	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 Δ	
Binom _u	Binom Δ	
Binom	Binom Δ	
Cauch _u	Cauch Δ	
CL _a	0 [STO] [K]	Check the OM for the conditions when this register is used.
CONST	CNST	For keyboard space reasons.
CONV	U \leftrightarrow	
DBLOFF		Your WP 43S features 34-digit DTs per default – it does neither need nor feature a <i>double precision mode</i> .
DBLON	Dispensable	
Expon	Expon Δ	
Expon _u	Expon Δ	
F(x)	F Δ (x)	
F _u (x)	F Δ (x)	
dRCL	Dispensable	Your WP 43S features various <i>data types</i> .

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

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

WP 34S	WP 43S	Remarks
$t(x)$	$t_{\Delta}(x)$	
$t_u(x)$	$t_{\Delta}(x)$	
VIEWa, VWa+	Disposable	Use VIEW instead; <i>alphanumeric strings</i> are just another <i>DT</i> . Combine text and numeric data easily using + as shown in the OM, Sect. 2.
Weibl	Weibl _Δ	
Weiblu	Weibl _Δ	
XEQa	Disposable	Use XEQ with an appropriate parameter instead.
XTAL?	Disposable	A quartz crystal is installed by default.
YDOFF, YDON	Disposable	Your <i>WP 43S</i> displays <i>y</i> whenever possible and wanted. See DSTACK.
αDATE, αDAY	Disposable	You can combine text and numeric data easily using + as shown in <i>Section 2</i> of the OM.
αGTO	Disposable	Use GTO w/ an appropriate parameter instead.
αIP, αMONTH	Disposable	You can combine text and numeric data easily using + as shown in <i>Section 2</i> of the OM.
αRCL, αRC#	Disposable	Your <i>WP 43S</i> features various <i>data types</i> and 'knows' which type is in the <i>register</i> specified. Appending texts is done by + .
αSTO	Disposable	Simply press STO instead (any <i>register</i> can take a <i>text string</i>).
αTIME	Disposable	See αDATE.
αXEQ	Disposable	Use XEQ with an appropriate parameter instead.
β	β(x,y)	
Γ	Γ(x)	
ΔDAYS	Disposable	Simply subtract two <i>dates</i> .
ζ	ζ(x)	
Φ(x) ...	Disposable	Use NORML... with $\mu=0$ and $\sigma=1$ instead.
$\chi^2(x)$	χ²Δ(x)	

WP 34S	WP 43S	Remarks
$\chi^2_u(x)$	$\chi^2_{\Delta}(x)$	
$\rightarrow H$	$\rightarrow HR$	
$\blacksquare PLOT$	n/a	See gCLR.
$\blacksquare C_{XY}$	Disposable	Use $\blacksquare r$ instead.
$\blacksquare \alpha,$ $\blacksquare \alpha +,$ $\blacksquare + \alpha$	Disposable	Combine text and numeric data easily using $\blacksquare +$ as shown in <i>Section 2</i> 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 350 operations, not counting the unit conversions and constants provided; 80 of them are even new or extended compared to earlier *WP* calculators. The commands are printed below as spelled on your *WP 43S*.

Command	WP 43S	WP 31S	WP 34S
2^x AGM	●	—	new
ALL	●	●	extended
AND ASR NOT OR XOR	●	—	extended
BACK CASE SKIP	●	—	new
BATT?	●	●	new
BC? FB	●	—	new
BestF	extended	●	●
BestF?	new	—	—

Command	WP 43S	WP 31S	WP 34S
Binom_p Binom_{Δ} (of Binomial distribution)	●	●	new
B_n B_n^* CEIL FLOOR	●	—	new
Cauch_p Cauch_{Δ} Cauch_{Δ} Cauch^{-1}	●	●	new
CauchF GaussF HypF ParabF RootF	new	—	—
CLCVAR	new	—	—
CLFall CLPall CONJ CONVG? COV	●	—	new
CX \rightarrow RE RE \rightarrow CX	new	—	—
DATE TIME	●	—	(●)
DATE \rightarrow DAY MONTH YEAR \rightarrow DATE	●	—	new
DEC DSL INC ISE	●	—	new
DECOMP	●	●	new
DEG \rightarrow D.MS \rightarrow GRAD \rightarrow RAD \rightarrow	●	—	new
DROP	●	—	new
DROPy DSTACK	new	—	—
D \rightarrow J J \rightarrow D	●	—	new
EIGVAL EIGVEC	new	—	—
ENTRY?	●	—	new
EQ.DEL EQ.EDI EQ.NEW	new	—	—
erf erfc ERR MSG	●	—	new
EVEN? ODD?	●	—	new
Expon_p Expon_{Δ} Expon_{Δ} Expon^{-1}	●	●	new
EXPT MANT	●	—	new
FBR	new	—	—
FC?F FC?S FF FS?F FS?S	●	—	new
FIB	●	—	new
FILL	●	●	new
FLASH? FP?	●	—	new
$F_p(x)$ $F_{\Delta}(x)$ (of F distribution)	●	●	new

Command	WP 43S	WP 31S	WP 34S
f' f''	new	—	—
f'(x) f''(x)	extended	—	new
GAP	extended	●	new
GCD LCM	●	●	new
g_d g_d ⁻¹	●	—	new
Geom _p Geom _Δ Geom _△ Geom ⁻¹	●	—	new
H _n H _{nP} L _m L _{ma} P _n T _n U _n	●	—	new
Hyper _p Hyper _Δ Hyper _△ Hyper ⁻¹	new	—	—
IDIV	●	—	new
IDIVR IM RE	new	—	—
INT? ISM? I _{xyz} IΓ _p IΓ _q	●	—	new
J _y (x)	new	—	—
J/G	extended	●	new
KEY? KTyp? LBL? LEAP?	●	—	new
LgNrm _p LgNrm _Δ LgNrm _△ LgNrm ⁻¹	●	—	new
LNβ LNΓ LOADP LOADR LOADSS LOADΣ LocR LocR? LOG ₂ LOG _{xy}	●	—	new
LOAD SAVE	●	●	new
Logis _p Logis _Δ Logis _△ Logis ⁻¹	●	●	new
max min MIRROR	●	—	new
MOD	●	●	new
MULπ MULπ→	new	—	—
M.LU M.SQR? NAND NaN? NEIGHB NOR	●	—	new
NBin _p NBin _Δ NBin _△ NBin ⁻¹	new	—	—
NEXTP PRIME?	extended	●	new
Norml _p Norml _Δ Norml _△ Norml ⁻¹	●	●	new
nΣ (callable by name)	●	●	new
OrthoF PLOT POINT	new	—	—

Command	WP 43S	WP 31S	WP 34S
PAUSE	●	—	extended
Poiss _p Poiss _Δ Poiss _Δ Poiss ⁻¹	●	●	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 \Rightarrow 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
sinc π	new	—	—
SL SR	●	—	extended
SLVQ SPEC?	●	—	new
S _m S _{mw} S _w	●	●	new
SNAP	new	—	—
SSIZE?	●	●	new
STATUS	extended	—	extended
S _{xy}	●	—	new
s(a) TDISP	new	—	—
TICKS	●	—	new

Command	WP 43S	WP 31S	WP 34S
TIMER	●	—	(●)
TOP? ULP?	●	—	new
$t_p(x)$ $t_{\Delta}(x)$ (of t distribution)	●	●	new
$t_x^y z_x^y$	●	—	new
undo (UNDO)	●	new	—
V ₄	new	—	—
VERS? WDAY WHO?	●	●	new
Weibl _p Weibl _Δ Weibl _Δ Weibl ⁻¹	●	●	new
W _m W _p W ⁻¹ WSIZE? \bar{x}_G XNOR	●	—	new
\bar{x}_H \bar{x}_{RMS} x→DATE	new	—	—
x<? x≤? x=? x≠? x≥? x>?	extended	—	extended
x=+0? x=-0? x≈?	●	—	new
y ^x	extended	●	●
Y.MD	●	●	new
αLENG?	extended	—	●
αPOS?	extended	—	—
αRL αRR αSL αSR	●	—	extended
$\beta(x,y)$ Γ_{xy} γ_{xy} ϵ ϵ_m ϵ_p $\zeta(x)$ Π_n Σ_n σ_w	●	—	new
Σ^1/x Σ^1/x^2 Σ^1/y Σ^1/y^2 $\Sigma \ln y/x$ $\Sigma x^2/y$ Σx^3 Σx^4 $\Sigma x/y$	new	—	—
$\Sigma \ln^2 x$ $\Sigma \ln^2 y$ $\Sigma \ln x$ $\Sigma \ln xy$ $\Sigma \ln y$ Σx Σx^2 $\Sigma x^2 y$ $\Sigma x \ln y$ Σxy Σy $\Sigma y \ln x$ Σy^2 (callable by names)	●	●	new
$\chi^2_p(x)$ $\chi^2_{\Delta}(x)$ (of chi-square distribution)	●	●	new
(-1) ^x xMOD ^MOD	●	—	new
±∞?	new	—	—
→DEG →RAD	●	●	new
→D.MS →MULπ	new	—	—
→GRAD	●	—	new

Command	WP 43S	WP 31S	WP 34S
→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 & applications	All vintage HP calculator manuals can be recommended
Programming	<i>HP-42S Owner's Manual</i> ⁷⁸ <i>HP-42S Programming Examples & Techniques</i> ⁸²

⁷⁸ Download from <http://www.hp41.net/forum/files/hp41net/manuel-hp42s-us.pdf>

Topic	Recommended literature
Root finding and numeric integration	<p><i>HP-34C Owner's Handbook and Programming Guide</i>⁷⁹</p> <p><i>HP-15C Owner's Handbook</i>⁸⁰</p> <p><i>HP-15C Advanced Functions Handbook</i>⁸¹</p> <p><i>HP-42S Programming Examples & Techniques</i>⁸²</p>
Accuracy of numerical calculations	<i>HP-15C Advanced Functions Handbook</i> , pp. 172 – 211. ⁸¹
Manipulating bits and <i>short integers</i>	<i>HP-16C Owner's Handbook</i> ⁸³
Statistical distributions and their application	<i>HP-21S Owner's Manual</i> ⁸⁴
Financial calculations	<i>HP-17BII+ User's Guide</i> ⁸⁵

Depending on your educational background and professional qualification, textbooks about various mathematical, scientific, or engineering topics may be helpful in addition. Ensure you know enough about what you compute (and check footnote 93 on p. 229 below as well as the last paragraph on p. 16 of the OM).

 The floating point standard *IEEE 754* was developed in 1985, after most of the calculators mentioned above were launched already (see https://en.wikipedia.org/wiki/Floating-point_arithmetic as a starter, also about floating point numbers in general).

⁷⁹ Read here: <https://www.yumpu.com/en/document/read/19323790/hp34c-slide-rule-museum>

⁸⁰ Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c03030589.pdf>

⁸¹ Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c03308725.pdf>

⁸² Download from <http://www.hp41.net/forum/fileshp41net/hp42s-programming-examples.pdf>

⁸³ Download from <http://www.hp41.net/forum/fileshp41net/hp16c.pdf>

⁸⁴ Download from <https://www.manualslib.com/manual/940232/Hp-Hp-21s.html##manual>

⁸⁵ Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c00363348.pdf>

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 at <https://www.msys2.org>. Install it following the on-site instructions (but in **c:/msys64**) until step 7. Then enter in the black *MinGW* window:

```
pacman -S mingw-w64-x86_64-gcc git base-devel mingw-w64-x86_64-gtk3  
This step may take many minutes.
```

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

```
git config pull.rebase false
```

```
PATH=/mingw64/bin:$PATH
```

```
PKG_CONFIG_PATH=/mingw64/lib/pkgconfig:$PKG_CONFIG_PATH
```

```
cd wp43s          for changing to the proper directory.  
Continues at 5).
```

1.b) alternatively do the following:

Open the folder

https://gitlab.com/Over_score/wp43s/tree/master/windows%20binaries

Open README.md and proceed as described therein.

Eventually run wp43s.exe. Continue at 6).

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.  
Continues at 5).
```

3) Under OSX (Mac):

Here is *Harald Overbeek's* step by step solution if you want the WP 43S simulator 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. 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** Continue with 6).

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.

4) Updating the simulator:

4.a) The following is for *Linux*, only if necessary:

cd wp43s Continue at 4b).

4.b) The following is for *Windows* (within the *MinGW* window) & *Linux*:

git pull for pulling all the changed files from *gitlab* repository.⁸⁶ Continue at 4c)

4.c) Optionally enter

rm backup.bin if you want to start the simulator reset to default.
Continue at 5).

5) Enter

make for compiling and building a new executable.⁸⁷
./wp43s.exe for starting the simulator.

6) The **simulator window will open** (looking like one of the pictures overleaf though larger).

⁸⁶ 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.

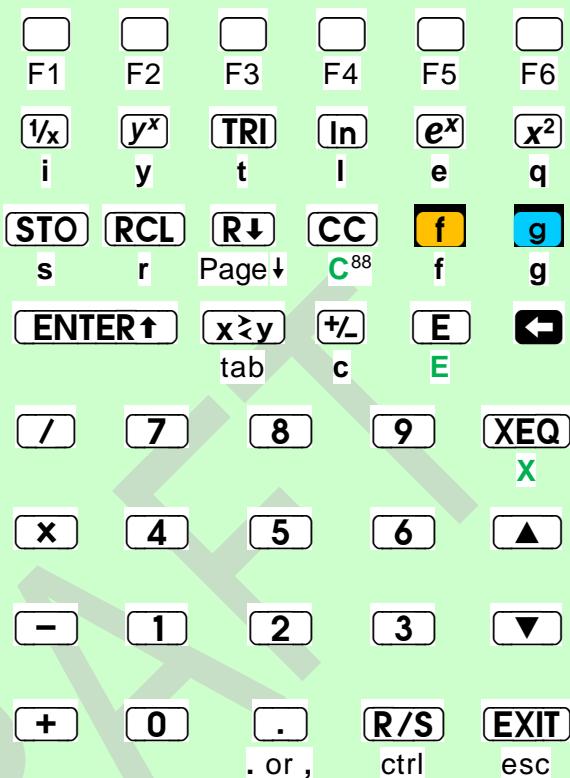
⁸⁷ There may be files updated by **git pull** but no new build possible sometimes. Then **make** will throw a corresponding message.

There may be also other obstacles or error messages in compilation (do not care for warnings or notes!); then **make rebuild** will clean the field before compiling (this cannot, however, overcome real errors in the software).

Operate the simulator with the mouse. The ten digits as well as . , $\text{ENTER}\uparrow$, $+$, $-$, \times , and $/$ may also be entered via the numeric keypad of your computer directly, \blacktriangleup and \blacktriangledown via the cursor keys, and \blackleftarrow via [\leftarrow Backspace]. Further computer keyboard shortcuts to simulator keys are presented overleaf.

(A vertical screen size of ≥ 980 px is required for the portrait window; else the landscape window will open needing 1000×568 px. The lower picture shows an old keyboard layout here.)





Right clicking will call **g**-shifted labels directly in any mode.

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 **B** making no sense on a computer application, ‘SL’ is displayed far right in the *status bar* whenever *automatic stack lift* is enabled (cf. *Section 1* of the OM).

⁸⁸ Capitals are printed green here for better differentiation.

APPENDIX F: FLASHING AND UPDATING YOUR WP 43S

There are two ways to get your hands on a *WP 43S*:

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

Way 2 (explained in next chapter) allows you to repurpose a *DM42* you own already, so you may save costs – but you will have to live with stickers on the keys then.

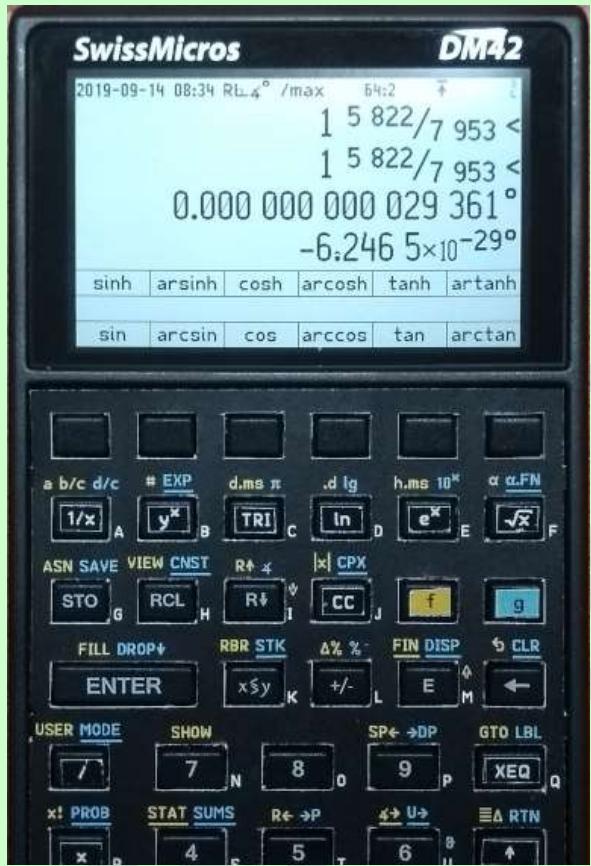
The chapter thereafter (on p. 212) shows how to update your existing *WP 43S*, be it bought or flashed, when a new firmware becomes available.

How to Flash Your *WP 43S*

1. Start your computer. Take your *DM42* and turn it on; then press **SETUP**
 - 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.

 - 6** Activate *USB* disk. The flash disk of your *DM42* should show up as an external mass storage volume on your computer now.
2. 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.



in https://gitlab.com/Over_score/wp43s/-/tree/master/artwork). Print them, cut, and apply (see above picture of an earlier WP 43S layout).

(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.)

3. Press **SETUP** **5** **2** **4** **3** WP43S.pgm **ENTER** **ENTER**. Wait some 15 s for flashing completed. Then press **EXIT** **EXIT** **1** **EXIT**. Now, your WP 43S is up and waiting for your commands.

The graphic files supplied may ease your life as long as you rely on a converted DM42 (find them

To leave the WP 43S program, enter **MODE** **9** **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); else EXIT/ON returns to the bottom left key now.

To retrieve the original DM42 keyboard layout (cf. p. 156), 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: https://technical.swissmicros.com/dm42-devel/dmcp-devel_manual/

How to Update Your WP 43S

If you have *Free42* still running on your *DM42* 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 **SAVE** to save your programs and data. Then leave *WP 43S* pressing **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. 156).

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.

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 the keys to match the *WP 43S* layout also when leaving *WP 43S*. Unless you have done this, **ON** remains bottom left.

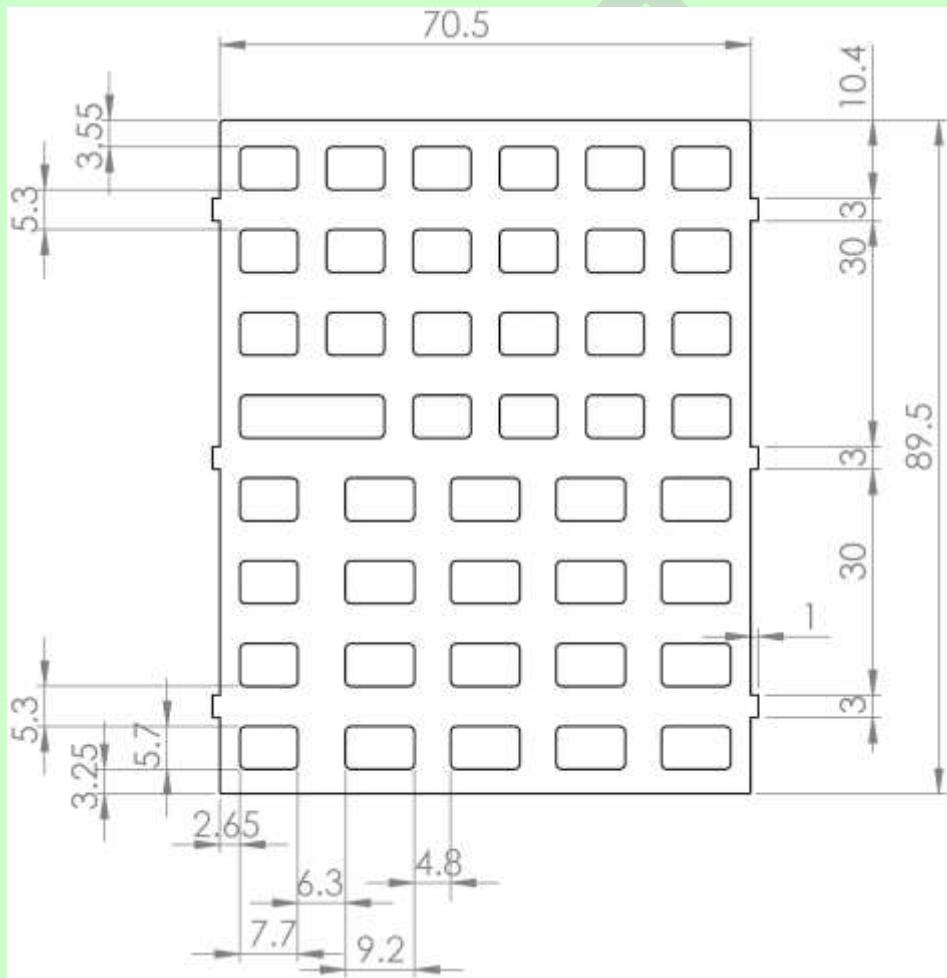
5. Press **EXIT** **ENTER** **3** (Load Program), use the cursor keys to select *WP43S.pgm*, and press **ENTER** **ENTER**.
6. Wait for flashing completed (~15 s). Then press **EXIT** **EXIT** **1** **+**. Recall your saved programs and data via **I/O LOAD**. You can resume your work with your updated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

Sometimes, you may need an update of the *DMCP* on your *WP 43S*; look

at https://www.swissmicros.com/dm42/doc/dm42_user_manual/#quick_update_guide
for how to do this.

Overlays

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



Find the original print (to scale) of keys and keyplate for the current version of WP 43S on the last page of this manual. Furthermore, there is a scaled table which may be used for creating your own overlay.

DRAFT

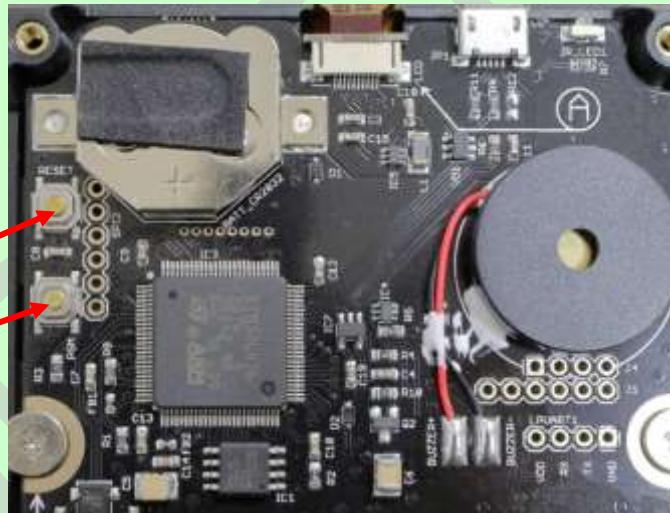
APPENDIX G: TROUBLESHOOTING GUIDE

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 find a printed circuit board (*PCB*) with its top probably looking like this → (cf. p. 159 for an earlier *PCB*).

In any case, you will see two small buttons, one labeled RESET; the other one is called PGM or BOOT0.



2. Now:

- a. Press and hold the PGM (or BOOT0) button.
- b. Press and release the RESET button.
- c. Release the PGM (or BOOT0) button.

This shall reset your *DM42* and put it in bootloader mode.⁸⁹

3. Then you can reflash your calculator using *dm_tool.exe* as written in https://www.swissmicros.com/dm42/doc/dm42_user_manual/.

⁸⁹ If this method should not work, however, this may point to a real hardware problem. We recommend contacting SwissMicros then.

Fresh Battery Constantly Low

This was observed with factory-fresh calculators in 2020: The low-battery indicator  is lit and BATT stubbornly returns 1.21 V, although the battery is actually good and measures over 3 V. If resetting the calculator, removing the battery, updating the firmware, etc. does not change the output of BATT, open the calculator and inspect the *PCB*. There may be an improper soldering at FB1 (compare previous picture).

If you have the right tools and are experienced, feel free to fix this yourself – else contact SwissMicros.



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  (Activate *USB Disk*).
4. Start the internet browser on your computer and go to 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 trouble again.

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

DRAFT

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 X : $B_n = (-1)^{n+1} \cdot n \cdot \zeta(1 - n)$ B_n^* works with the old definition instead:
B_n^*	$B_n^* = 2 \cdot \frac{(2n)!}{(2\pi)^{2n}} \cdot \zeta(2n)$ See p. 255 for $\zeta(x)$.

Name	Remarks (see pp. 13ff for general information)
COMB, PERM	<p>For $y \geq x \geq 0$ and $x, y \in \mathbb{N}$, $C_{y,x} = \binom{y}{x} = \frac{y!}{x!(y-x)!}$ is the number of <i>combinations</i> and $P_{y,x} = \frac{y!}{(y-x)!} = x!C_{y,x}$ the number of <i>permutations</i> of x and y as explained in the IOP (see pp. 27 and 58, respectively).</p> <p>Note $C_{y,0} = 1$, $C_{y,1} = y$, and $C_{y,2} = \frac{1}{2}y(y-1)$.</p> <p>$C_{y,x}$ applies to the <i>binomial distribution</i> (see p. 220): In a <i>Galton box</i>⁹⁰ (a.k.a. <i>bean machine</i>) featuring y rows of pins and fed with 2^y balls, $C_{y,x}$ is the number of balls expected in column x of that box (start column counting with zero).</p> <p>Generally, $P_{y,x} = \frac{\Gamma(y+1)}{\Gamma(y-x+1)}$ and $C_{y,x} = \frac{\Gamma(y+1)}{\Gamma(x+1) \cdot \Gamma(y-x+1)}$ work also for non-integer numbers and in <i>complex</i> domain.</p>
FIB	<p>For integers, FIB returns the <i>Fibonacci</i> number f_n with $n = x$. The <i>Fibonacci</i> numbers are defined as $f_0 = 0$, $f_1 = 1$, and $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$. With UNSIGN, f_{93} is the maximum before an overflow occurs. For <i>long integers</i>, f_{4791} is the maximum on your WP 43S.</p> <p>For non-integers, 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 x, with $\Phi = \frac{1+\sqrt{5}}{2}$ denoting the <i>golden ratio</i>.</p>

⁹⁰ 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 p₀* in **I** and the *sample size n* in **J**.

BINOM_P returns

$$p_B(g; n; p_0) = \binom{n}{g} p_0^g (1 - p_0)^{n-g} = C_{n,g} p_0^g (1 - p_0)^{n-g} \quad (\text{see COMB on p. 219 for the explanation of the notation}).$$

BINOM_M 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_M**

returning 0.633 – so the odds are almost two out of three that you

will not detect any defect in your sample! ⁹¹

Read here for more information:

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

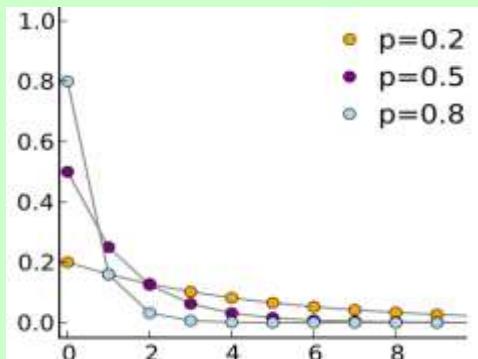
Geom: Geometric distribution:

GEOM_P returns

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

GEOM_A returns

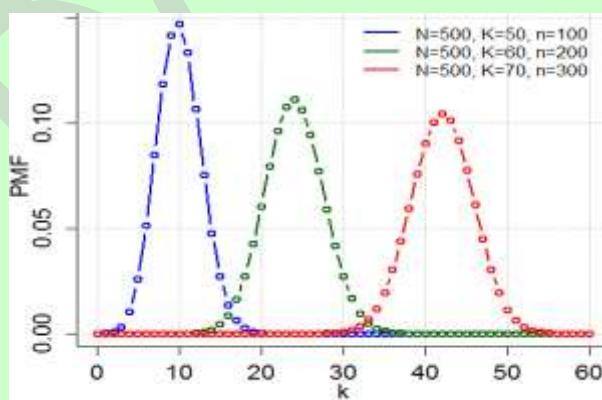
$F_{Ge}(m) = 1 - (1 - p_0)^{m+1}$, being the probability for a first success after $m = x$ Bernoulli experiments. The probability p_0 for a success in each such experiment must be specified in I.



Start reading here for more:

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

Hyper: Hypergeometric distribution with the number of successes g in X, gross probability of a success p_0 in I, sample size n in J, and batch size n_0 in K (in the diagram, $g=k$, $p_0=K/N$, and $n_0=N$).



⁹¹ 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.

HYPERP returns $p_H(g; n; p_0; n_0) = \frac{\binom{n_0 p_0}{g} \cdot \binom{n_0 (1-p_0)}{n-g}}{\binom{n_0}{n}}$ (see COMB on p. 219 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 so-called 'large' batches ($n_0 > 10$) and for small sample sizes (<10% of n_0). Start reading here for more: http://en.wikipedia.org/wiki/Hypergeometric_distribution.

NBin: Negative binomial distribution with the total number of failures f (in n draws) in \mathbf{X} , the gross probability of a success in a single draw p_0 in \mathbf{I} , and n in \mathbf{J} .

NBINP 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. 219 and cf. BINOM).

Start reading here for more:

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

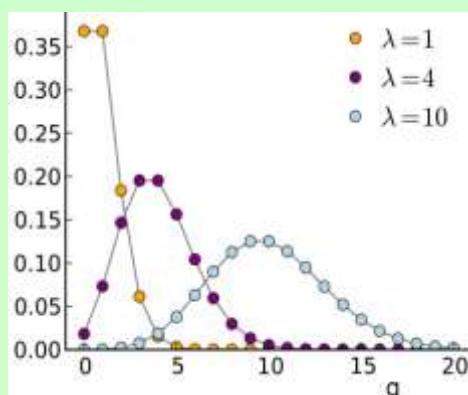
Poiss: Poisson distribution with the number of successes \mathbf{g} in \mathbf{X} and the Poisson parameter λ in \mathbf{J} .

POISSP 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 \mathbf{X} .

The Poisson distribution provides the mathematically simplest model for industrial



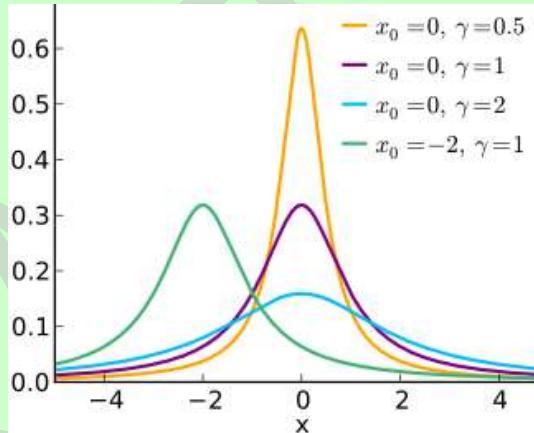
sampling tests – use $\lambda = np_0$ with the gross error probability p_0 and the sample size n (cf. BINOM). For the example introduced with BINOM above, POISS returns 0.638.

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 γ in J.



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

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

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

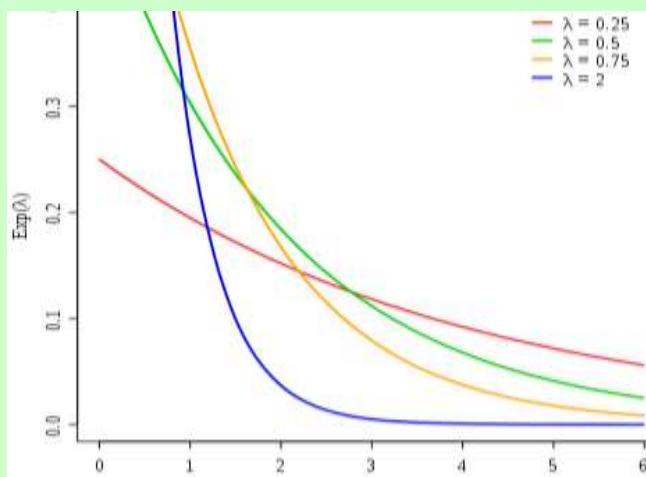
This distribution is quite popular in physics. It is a special case of Student's t distribution. Start reading here for more:

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

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

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

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



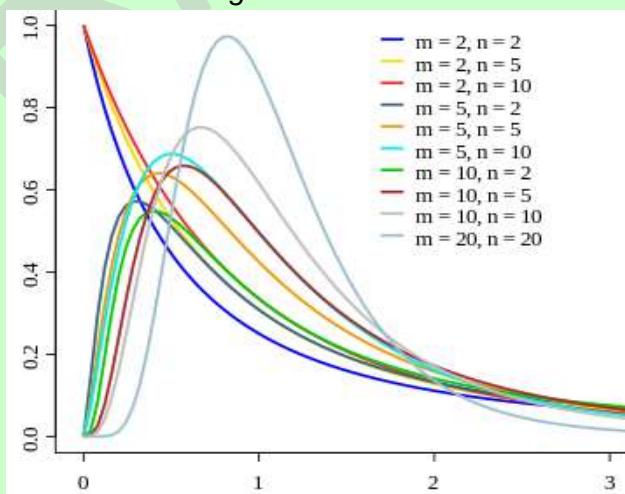
Read here for more information:

<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 .



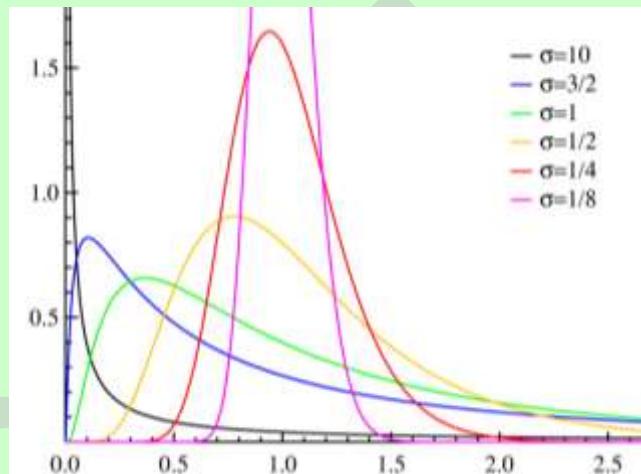
Read here for more information:

<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).

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

LGNRM_A: returns $F_{Ln}(x) = \Phi\left(\frac{\ln(x)-\mu}{\sigma}\right)$ with $\Phi(z)$ denoting the *standardized normal CDF* as presented on p. 226.



Read here for more information:

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

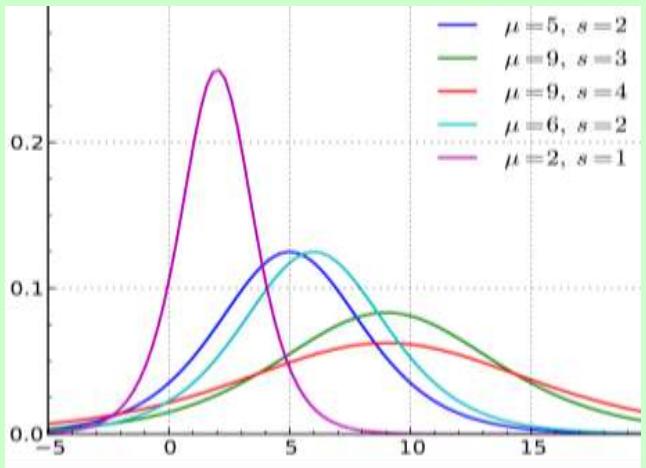
Logis: *Logistic distribution* with an arbitrary *mean* μ given in **I** and a *scale parameter* s in **J**.

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

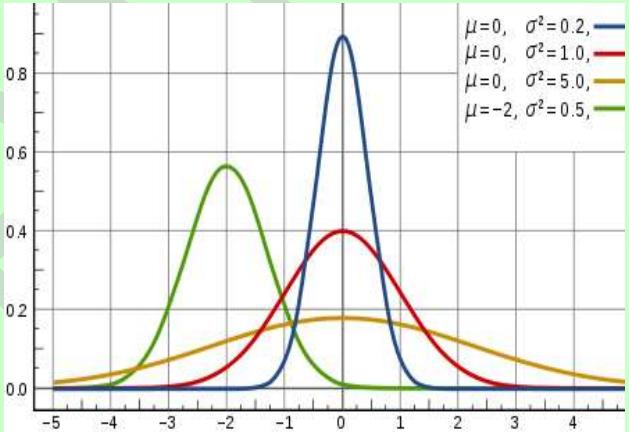
$$\text{LOGIS}_{\text{P}} \text{ returns } f_{Lg}(x) = \frac{e^{-\xi}}{(1+e^{-\xi})^2 s} \text{ (plotted overleaf) and}$$

$$\text{LOGIS}_{\text{A}} \text{ returns } F_{Lg}(x) = \frac{1}{1+e^{-\xi}}.$$

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



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



NORML_P 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

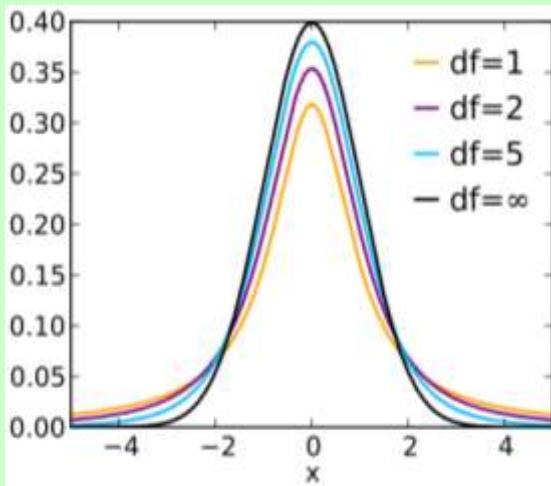
NORML_A returns $F_N(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$ with $\Phi(z)$ denoting the *standardized normal CDF* (cf. the *error function* on p. 251).

Read here for more information:

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

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

It is used for hypothesis testing and calculating confidence intervals e.g. for means. The picture shows its *PDF* plotted for different *degrees of freedom*. For $df \rightarrow \infty$, the shoulders of $t(x)$ shrink and it approaches the *PDF* of the *standard normal distribution* (compare the red *Gaussian* curve at NORML on p. 226).



Read here for more information:

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

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

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

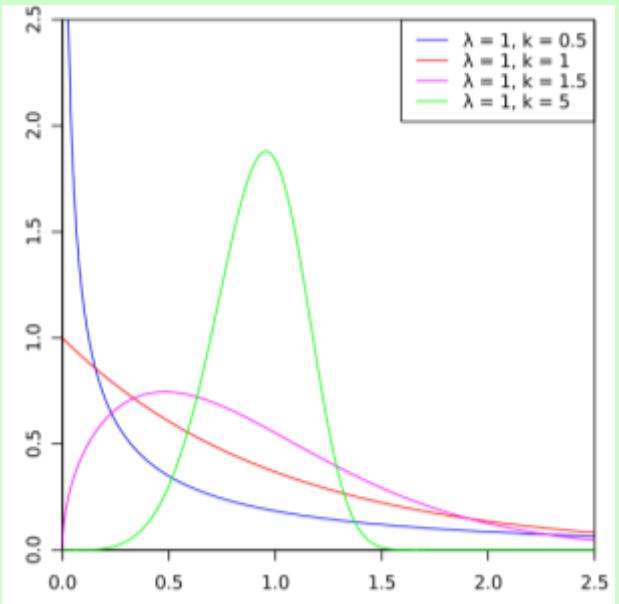
WEIBL_A returns $F_W(t) = 1 - e^{-(t/T)^b}$

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>

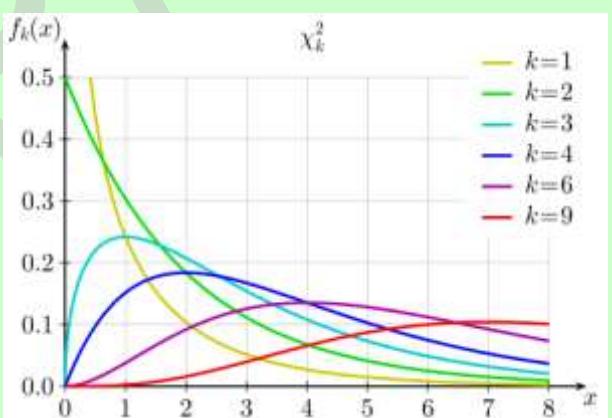
You may even find some more application fields mentioned in https://en.wikipedia.org/wiki/Weibull_distribution#Applications.



$\chi^2(x)$: Chi-square distribution with its degrees of freedom given in \mathbf{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 ε_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.⁹² Be assured not everything is *Gaussian* in real world.⁹³ 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.

⁹² 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.

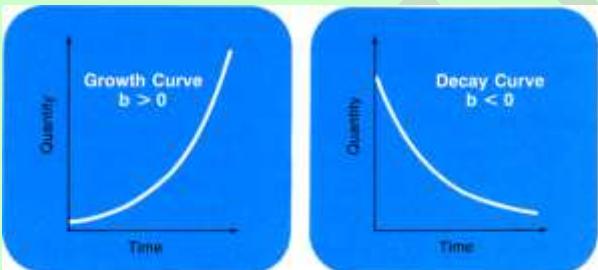
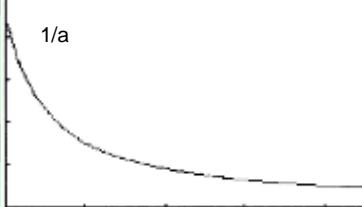
As mentioned in the main text, we recommend you look deeply into statistics textbooks to ensure you fully understand what you do with the functions provided in your WP 43S. The real world shows lots of sad examples where people full of good will caused large damages by applying tools they did not know 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).

⁹³ 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.

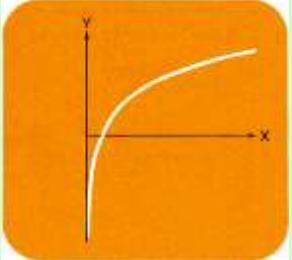
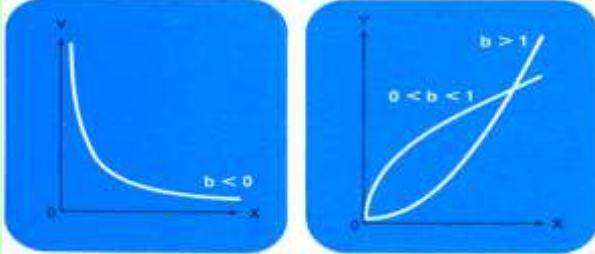
The following functions as named in the left column (sorted alphabetically) are all found in STAT:

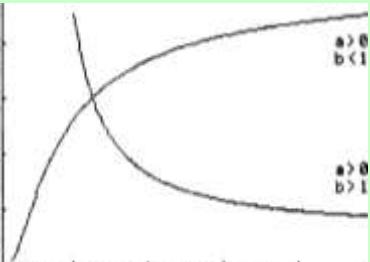
Name	Remarks (see pp. 13ff for general information)
CauchF	Selects a <i>Cauchy</i> (a.k.a. <i>Lorentz</i> , <i>Breit-Wigner</i>) peak fit model $R(x) = \frac{1}{[a_0(x + a_1)^2 + a_2]}$ for least squares regression. ⁹⁴ See p. 223 for shapes of such peaks.
CORR	For any set of data points (x_i, y_i) , 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 independent data x . For $r^2 = 1$, y is fully determined by x ; for $r^2 = 0$, y is completely independent of x ; and e.g. $r^2 = 0.85$ means 85% of the variation of y is due to x . 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. 227).

⁹⁴ 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	<p>For any set of data points (x_i, y_i), the <i>population covariance</i> is</p> $COV_{xy} = \frac{1}{n^2} \left(n \sum x_i y_i - \sum x_i \sum y_i \right)$ <p style="text-align: right;">Compare s_{XY} below.</p>
ExpF	<p>Selects the exponential curve fit model $R(x) = a_0 e^{a_1 x}$ for least squares regression.⁹⁴ 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).⁹⁵</p> 
GaussF	<p>Selects a <i>Gauss peak</i> fit model $R(x) = a_0 e^{\frac{(x-a_1)^2}{a_2}}$ for least squares regression.⁹⁴ See p. 226 for the shapes of such peaks.</p>
HypF	 <p>Selects the hyperbolic fit model $R(x) = \frac{1}{(a_0 + a_1 x)}$ for least squares regression.⁹⁴</p>
LinF	<p>Selects the linear fit model $R(x) = a_0 + a_1 x$ for least squares regression.⁹⁴ Generally, this will be a good choice if the measured data follow a straight line, raising or falling (but compare ORTHOF).</p>

⁹⁵ 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 $R(x) = a_0 + a_1 \ln(x)$ for least squares regression.⁹⁴ 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> $n - 2$ (cf. p. 227).</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. 235ff 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. ⁹⁴
PowerF	<p>Selects the power curve fit model $R(x) = a_0 x^{a_1}$ for least squares regression.⁹⁴ 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).⁹⁵</p> 

Name	Remarks (see pp. 13ff for general information)
RootF	 <p>Selects the root curve fit model $R(x) = a b^{1/x} = a_0 a_1^{1/x}$ for a least squares regression.⁹⁴</p>
s_{xy}	<p>For any set of data points (x_i, y_i), 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 \bar{x}) is $s_m = s / \sqrt{n}$</p>
s_w, s_{mw}	<p>The <i>sample SD</i> for <u>weighted</u> data (where the weight y_i of each data point x_i was entered via $\Sigma+$) is</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 \bar{x}_w) 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 $\bar{x}_w = \frac{\sum x_i y_i}{\sum y_i}$
ε	The scattering factor ε_x for a sample of <i>log-normally</i> distributed data is calculated via: $\ln(\varepsilon_x) = \sqrt{\frac{1}{n-1} \left[\sum \ln^2(x_i) - 2n \ln(\bar{x}_G) \right]}$ Compare s.
ε_m	The scattering factor of the geometric mean is $\varepsilon_m = \varepsilon^{\frac{1}{\sqrt{n}}}$. Compare s.m.
ε_p	The scattering factor ε_p for a population of <i>log-normally</i> distributed data is calculated via: $\ln(\varepsilon_p) = \sqrt{\frac{n-1}{n}} \ln(\varepsilon)$ Compare σ .

Name	Remarks (see pp. 13ff for general information)
σ	The <i>SD</i> of a population of <i>normally</i> distributed data is calculated via $\sigma = \sqrt{\frac{n-1}{n}} s$
σ_w	The <i>SD</i> of the population for <u>weighted</u> data (where the weight y_i of each data point x_i was entered via $\Sigma+$) is $\sigma_w = \sqrt{\frac{\sum y_i (x_i - \bar{x}_w)^2}{\sum y_i}}$

About 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. 89f and increments **n**.
2. The evaluation depends on the fit model you select (cf. pp. 231ff):

- 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. 236 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. 236 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. 236 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

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

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

- 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 $R(x) = a_0 + a_1 x + a_2 x^2$ 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 42), 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 Δx_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 ΔR , even if f is ‘strongly curved’:

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

Being through with all n parameters, you will end with

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

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

For ‘small’ errors or less curvature 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

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

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 δ taking care of the viscosity of the medium as well as size and shape of the falling body as a whole.

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

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

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

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

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

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

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

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

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

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

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

$$f_1 = f_0 + \left(\frac{df}{dt}\right)_{1/2} \text{ and } \left(\frac{df}{dt}\right)_{3/2} = \left(\frac{df}{dt}\right)_{1/2} + a\Delta t$$

$$f_2 = f_1 + \left(\frac{df}{dt}\right)_{3/2} \Delta t, \text{ etc.}$$

Let us drop the restriction for b now. Replacing a in the previous set of equations by the right side of the differential equation on p. 243, we will get the following new set:

$$\frac{df}{dt_{1/2}} \approx \frac{df}{dt_0} + \left[a - b \left(\frac{df}{dt} \right)_0^2 \right] \frac{\Delta t}{2}$$

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

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

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

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

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

$$f_2 = f_1 + \left(\frac{df}{dt} \right)_{3/2} \Delta t, \text{ etc.}$$

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

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

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

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

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

$$f_{i+1} = f_i + \left[\frac{df}{dt} \right]_{i-1/2} \Delta t$$

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:

$$\begin{aligned} \left(\frac{dx}{dt}\right)_{1/2} &\approx \left(\frac{dx}{dt}\right)_0 + K_x \frac{\Delta t}{2} & \left(\frac{dy}{dt}\right)_{1/2} &\approx \left(\frac{dy}{dt}\right)_0 + K_y \frac{\Delta t}{2} \\ \left(\frac{dx}{dt}\right)_{i+1/2} &\approx \left(\frac{dx}{dt}\right)_{i-1/2} + K_x \Delta t & \left(\frac{dy}{dt}\right)_{i+1/2} &\approx \left(\frac{dy}{dt}\right)_{i-1/2} + K_y \Delta t \\ x_{i+1} &\approx x_i + \left(\frac{dx}{dt}\right)_{i+1/2} \Delta t & y_{i+1} &\approx y_i + \left(\frac{dy}{dt}\right)_{i+1/2} \Delta t \end{aligned}$$

Orthogonal Polynomials

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

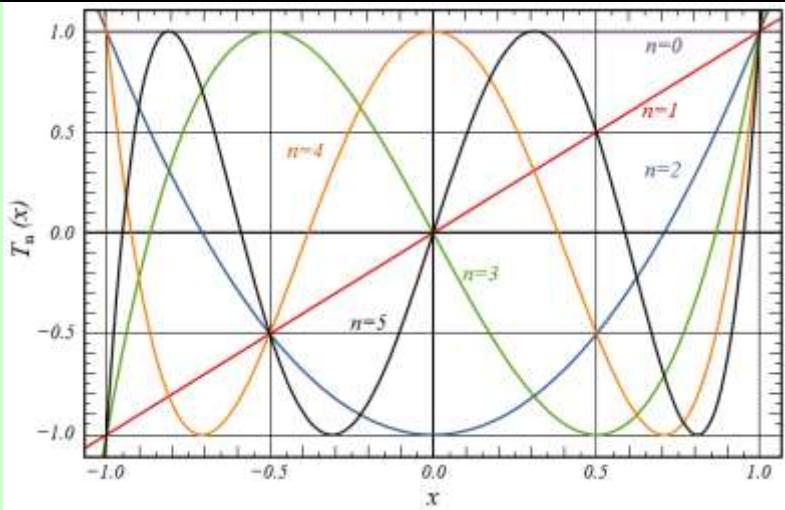
Name	Remarks (see pp. 13ff for general information)
H_n	<p>Hermite polynomials for <u>probability</u>: $H_n(x) = (-1)^n \cdot e^{x^2/2} \cdot \frac{d^n}{dx^n} \left(e^{-x^2/2}\right)$</p> <p>with n in \mathbb{Y}, solving the differential equation</p> $f''(x) - 2x \cdot f'(x) + 2n \cdot f(x) = 0 .$ <p>See the first five polynomials plotted overleaf.</p>

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

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

Name	Remarks (see pp. 13ff for general information)
P_n	<p>Legendre polynomials: $P_n(x) = \frac{1}{2^n n!} \cdot \frac{d^n}{dx^n} [(x^2 - 1)^n]$ with n in \mathbb{Y}, solving the differential equation</p> $\frac{d}{dx} \left[(1 - x^2) \cdot \frac{d}{dx} f(x) \right] + n(n+1)f(x) = 0.$ <p>See the first six polynomials plotted here:</p>
T_n	<p>Chebyshev (a.k.a. Čebyšev, Tschebyschow, Tschebyscheff) polynomials of first kind</p> <p>$T_n(x) = \begin{cases} \cos(n \arccos(x)) & \text{for } -1 \leq x \leq 1 \\ \cosh(n \operatorname{arcosh}(x)) & \text{for } x > 1 \\ (-1)^n \cosh(n \operatorname{arcosh}(-x)) & \text{for } x < -1 \end{cases}$ with n in \mathbb{Y}, solving the differential equation</p> $f''(x) - \frac{x}{1-x^2} f'(x) + \frac{n^2}{1-x^2} f(x) = 0$ <p>The plot overleaf shows $T_0(x) \dots T_5(x)$.</p>

Name	Remarks (see pp. 13ff for general information)
------	--

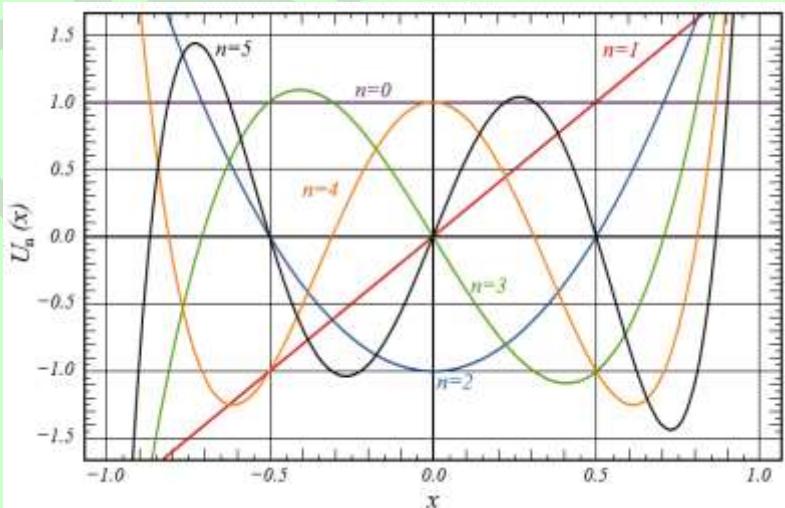


U_n

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

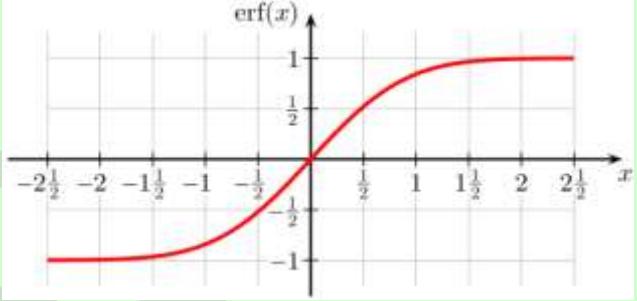
$$f''(x) - \frac{3x}{1-x^2}f'(x) + \frac{n(n+2)}{1-x^2}f(x) = 0$$

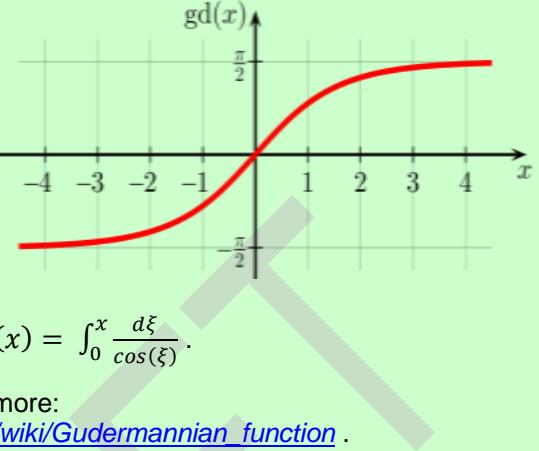
The plot below shows $U_0(x) \dots U_5(x)$:



Even More Mathematical Functions

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

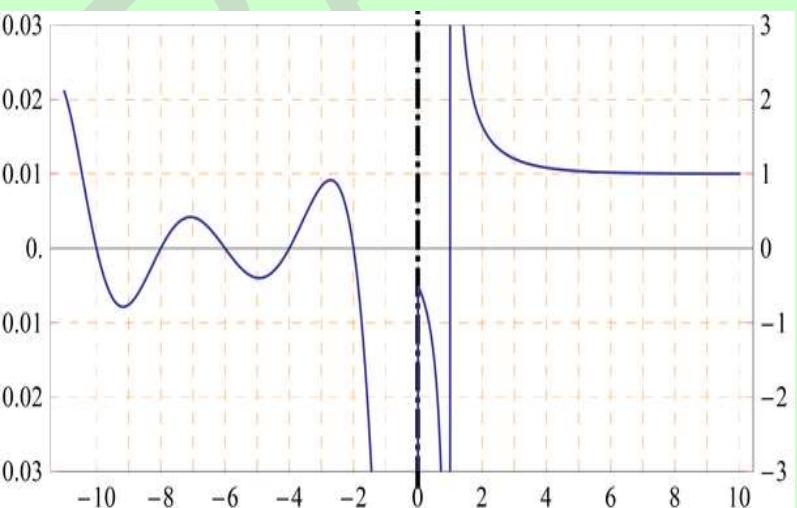
Name	Remarks (see pp. 13ff for general information)
AGM	Returns the <i>arithmetic-geometric mean</i> . Find more about it here: http://mathworld.wolfram.com/Arithmetic-GeometricMean.html .
erf	<p>Returns the <i>error function</i> $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\tau^2} d\tau$.</p> <p>Note that</p>  <p>$\text{erf}\left(\frac{x}{\sqrt{2}}\right) = 2 \Phi(x) - 1$ with $\Phi(x)$ representing the <i>standardized normal CDF</i> as described on p. 226.</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>standardized 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 real values. The inverse of this function is $g_d^{-1}(x) = \int_0^x \frac{d\xi}{\cos(\xi)}$.</p> <p>Start reading here for more: http://en.wikipedia.org/wiki/Gudermannian_function .</p> 
I_{xyz}	<p>Returns the <i>regularized (incomplete) Beta function</i> $\beta_x(x, y, z) / B(y, z)$</p> <p>with $\beta_x(x, y, z) = \int_0^x \tau^{y-1} (1-\tau)^{z-1} d\tau$ being the <i>incomplete Beta function</i> and $B(y, z)$ being <i>Euler's Beta function</i> (see p. 87 and https://en.wikipedia.org/wiki/Beta_function).</p>
$I\Gamma_p$	<p>Returns the <i>regularized Gamma function</i> $P(x, y) = \frac{\gamma(x, y)}{\Gamma(x)}$</p> <p>See γ_{XY} below for $\gamma(x, y)$ and p. 87 for $\Gamma(x)$.</p>
$I\Gamma_q$	<p>Returns the <i>regularized Gamma function</i> $Q(x, y) = \frac{\Gamma_u(x, y)}{\Gamma(x)}$</p> <p>See Γ_{XY} below for $\Gamma_u(x, y)$ and p. 87 for $\Gamma(x)$.</p>

See here for more:
<https://en.wikipedia.org/wikilink>

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>$J_y(x)$ returns the <i>Bessel function of first kind</i> and order $y=\nu$. For arbitrary ν, this is</p> $J_\nu(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m! \Gamma(m+\nu+1)} \left(\frac{x}{2}\right)^{2m+\nu}$ <p>For integer ν, this is also</p> $J_\nu(x) = \frac{1}{\pi} \int_0^\pi \cos[\nu t - x \sin(t)] dt$ <p>Start reading here for more information: http://en.wikipedia.org/wiki/Bessel_function.</p>

Name	Remarks (see pp. 13ff for general information)
sinc, sincπ	<p>The graph shows two functions plotted against x from -20 to 20. The x-axis has major ticks at $-6\pi, -4\pi, -2\pi, 0, 2\pi, 4\pi, 6\pi$. The y-axis ranges from -0.2 to 1.0. A red curve represents $\frac{\sin(x)}{x}$, which has a sharp peak at $(0, 1)$ and oscillates with decreasing amplitude as x increases. A blue curve represents $\frac{\sin(\pi x)}{\pi x}$, which also has a peak at $(0, 1)$ but exhibits much smaller, higher-frequency oscillations around the x-axis.</p>
W_p , W_m	<p>Return Lambert's W with its principal branch (called W_p here) and its negative branch (called W_m for minus). The connecting point is $(x, y) = (-1/e, -1)$. The diagram shows the <i>real</i> values of both branches.</p> <p>Start reading here for more information: http://en.wikipedia.org/wiki/Lambert_W_function. Learn more here: http://mathworld.wolfram.com/LambertW-Function.html.</p> <p>The graph plots the real values of the Lambert W function against x from -1 to 4. The y-axis ranges from -3 to 1. Two curves are shown: the principal branch $W_p(x)$, which starts at $(-1/e, -1)$ and increases monotonically, and the negative branch $W_m(x)$, which starts at $(-1/e, -1)$ and decreases towards negative infinity as x approaches 0 from the left.</p>

Name	Remarks (see pp. 13ff for general information)
γ_{xy}	Returns the <i>lower incomplete Gamma function</i> $\gamma(x, y) = \int_0^y t^{x-1} e^{-t} dt$. Required for $I\Gamma_p$ above.
Γ_{xy}	Returns the <i>upper incomplete Gamma function</i> $\Gamma_u(x, y) = \int_y^\infty t^{x-1} e^{-t} dt$. Required for $I\Gamma_q$ above.
$\zeta(x)$	Returns <i>Riemann's Zeta</i> for <i>real</i> arguments, with $\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x}$ for $x > 1$, and its analytical continuation for $x < 1$: $\zeta(x) = 2^x \pi^{x-1} \sin\left(\frac{\pi}{2}x\right) \cdot \Gamma(1-x) \cdot \zeta(1-x).$ Note the different vertical scales for negative and positive x in the plot: 

See here for more:
https://en.wikipedia.org/wiki/Incomplete_gamma_function

Look here for more:
<http://mathworld.wolfram.com/RiemannZetaFunction.html>.

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

Beyond what is printed in this appendix, you will also find lots of information about the special functions implemented in your *WP 43S* in the internet. Generally, *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. 220ff) is a competent source. And *Mathworld* (quoted on pp. 251ff) or *WolframAlpha* may contain more details than you ever want to know. Further references are found at these sites.

DRAFT

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 FACT₂ will allocate a local register (**R.00₂**) and store **3** therein. After decrementing x , FACT will call itself again.

Then FACT₃ will allocate a local register (**R.00₃**) and store **2** therein. After decrementing x , FACT will call itself once more.

Then FACT₄ will return to FACT₃ with $x = 1$. This x will be multiplied by **r.00₃** there, returning to FACT₂ with $x = 2$. This x will be multiplied by **r.00₂** 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:\msys64** 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:\msys64\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 the *name* of each and every *item* provided, be it a command, constant (# ...), 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:

\hat{x}	81	# K_J	133	# Δv_{Cs}	136	\uparrow Lim (V)	104
-	91	# l_{PL}	133	# ε_0	136	\rightarrow (c)	92
\bar{x}	81	# m_p	134	# λ_C	136	\rightarrow D.MS	92
\bar{x}_w	83	# M_{\oplus}	134	# λ_{Cn}	136	\rightarrow DATE	92
$\cancel{4} \rightarrow$ (M)	97	# M_{\odot}	134	# λ_{CP}	136	\rightarrow DEG	92
\bar{x}_G	82	# m_e	133	# G_B	137	\rightarrow GRAD	92
# 99		# M_{Moon}	133	# Φ	137	\rightarrow H.MS	92
# μ_p	137	# m_n	133	# Φ_0	137	\rightarrow HR	92
# μ_0	137	# m_n/m_p	133	# ω	138	\rightarrow INT	92
# μ_B	137	# m_p	134	#B	99	\rightarrow MUL π	93
# μ_e	137	# m_p/m_e	134	#DEC (V)		\rightarrow POL	93
# μ_e/μ_B	137	# m_{PL}	134	104		\rightarrow RAD	93
# μ_n	137	# m_u	134	%	95	\rightarrow REAL	93
# μ_p	137	# m_vc^2	134	%+MG	95	\rightarrow REC	93
# μ_u	137	# N_A	134	%MRR	95	\downarrow Lim (V)	104
# ∞	138	# NaN	134	%T	95	\geq	94
# $-\infty$	138	# p_0	134	% Σ	95	$^{\circ}$ C \rightarrow F	17
# a	131	# R_{∞}	135	(-1) x	91	$^{\circ}$ F \rightarrow C	17
# a_{\oplus}	132	# R	134	(χ^2) $^{-1}$	90	\blacksquare #	99
# a_0	131	# R_{\odot}	135	/	92	\blacksquare ADV	97
# a_{Moon}	131	# R_{\oplus}	135	[M] $^{-1}$	91	\blacksquare CHAR	97
# c	132	# r_e	134	[M] T	91	\blacksquare DLAY	97
# c_1	132	# R_K	135	\wedge MOD	92	\blacksquare LCD	97
# c_2	132	# R_{Moon}	135		95	\blacksquare MODE	97
# e	132	# Sa	135	M	94	\blacksquare PROG	97
# e_E	132	# Sb	135	x	94	\blacksquare r	98
# F_{α}	132	# Se^2	135	+	91	\blacksquare REGS	98
# F	132	# Se'^2	135	+/-	91	\blacksquare STK	98
# F_{δ}	132	# Sf^{-1}	135	$\pm\infty?$	91	\blacksquare TAB	98
# g_{\oplus}	133	# T_0	135	x	91	\blacksquare USER	98
# G	132	# T_p	135	xMOD	92	\blacksquare WIDTH	98
# G_0	132	# t_{PL}	135	\sqrt{x}	96	\blacksquare Σ	99
# G_C	132	# V_m	136	$\sqrt[3]{x}$	18	1/x	17
# g_e	133	# Z_0	136	$\sqrt[4]{y}$	83	10^x	17
# GM_{\oplus}	133	# α	136	$\not x$	96	$1COMPL$	17
# h	133	# γ	136	\int	96	$2COMPL$	18
# \hbar	133	# γ_{EM}	136	$\int f$ (M)	96	2^x	18
# k	133	# γ_p	136	$\int f dx$ (M)	96	A (V)	100

A...Ω (M)	86	BeginP	22	CONST (M)		DENMAX (V)	
A:	21	BestF	22		27		100
ABS	18	BestF?	23	CONVG	27	DET	31
ac→m ²	18	Binom _Δ	23	CORR	28	DIGITS (M)	
ac _{us} →m ²	18	Binom ₋₁	23	cos	28		31
ACC (V)	100	Binom _▲	23	cosh	28	DISP (M)	31
ACOS	18	Binom:	23	COV	28	DMY (S)	105
ADM (V)	100,	Binom _p	23	CPX (M)	28	DOT	31
101		BITS	23	CPX?	28	DROP	31
ADV	18	B _n	23	CPXj (S)	105	DROPy	31
AGM	18	B _n *	23	CPXRES (S)		DSE	31
AGRAPH	18	BS?	23		105	DSL	32
ALL	19	Btu→J	23	CPXS (M)	28	DSTACK	32
ALLENG (S)		C (V)	100	CROSS	29	DSZ	32
107		cal→J	24	ct→kg	29	E:	35
ALP.IN (S)		CARRY (S)		cwt→kg	29	EIGVAL	33
108			106	CX→RE	29	EIGVEC	33
ALPHA (S)		CASE	24	D (V)	100	END	33
106		CATALOG	24	D.MS	32	ENDP	33
AND	19	Cauch ⁻¹	25	D.MS→	32	ENG	33
ANGLES (M)		Cauch _Δ	25	D.MS→D	32	ENORM	33
19		Cauch _▲	25	D.MY	32	ENTER↑	34
arccos	19	Cauch:	25	D→D.MS	32	ENTRY?	34
arcosh	19	CauchF	25	D→J	33	EQ.DEL	34
arcsin	20	Cauch _p	25	D→R	33	EQ.EDI	34
arctan	20	CB	25	DATE	29	EQ.NEW	34
artanh	20	CEIL	25	DATE→	29	EQN (M)	34
ASIN	20	CF	25	DATES (M)		erf	34
ASL.BLK (S)		CHARS (M)			29	erfc	34
108			25	DAY	30	ERR	34
ASR	20	CLALL	25	dB→fr	30	EVEN?	34
ASSIGN	20	CLCVAR	26	dB→pr	30	EXITALL	34
ATAN	20	CLFALL	26	DBL/	30	EXP (M)	35
atm→Pa	21	CLK (M)	26	DBL×	30	ExpF	35
au→m	21	CLLCD	26	DBLR	30	Expon _Δ	35
AUTOFF (S)		CLMENU	26	DEC	30	Expon _▲	35
107		CLP	26	DECIM. (S)		Expon ⁻¹	35
AUTXEQ (S)		CLPALL	26		107	Expon:	35
107		CLR (M)	26	DECOMP	30	Expon _p	35
B (V)	100	CLREGS	26	DEG	30	EXPT	35
BACK	21	CLSTK	26	DEG→	30	e ^x	34
bar→Pa	21	CLX	27	DENANY (S)		e ^{x-1}	35
BATT?	21	CLΣ	27		105	f' (M)	37
bbl→m ³	21	CNST	27	DENFIX (S)		f'' (M)	37
BC?	21	COMB	27		106	f''(x)	38
BEEP	21	CONJ	27	DENMAX	31	F&p:	38

F ⁻¹ (p)	37	Geom ⁻¹	39	IP	42	LgNrm ₄	47
f'(x)	38	Geom:	39	ISE	42	LgNrm ⁻¹	47
F _Δ (x)	37	Geom _p	39	ISG	43	LgNrm _Δ	47
F _Δ (x)	37	gl _{UK} →m ³	39	ISM?	43	LgNrm:	47
F:	37	gl _{us} →m ³	39	ISZ	43	LgNrm _p	47
FB	35	GRAD	39	I _{xyz}	43	LinF	47
FBR	35	GRAD→	39	IΓ _p	43	LJ	47
FC?	35	GRAMOD (V)		IΓ _q	43	L _{mα}	47
FC?C	36	101		J-	44	L _m	47
FC?F	36	GROW (S)	107	J (V)	101	LN	47
FC?S	36	GTO	39	J/G	44	LN(1+x)	48
FCNS (M)	35	GTO.	40	J+	44	LNβ	48
FF	36	̄x _H	82	J→Btu	44	LNΓ	48
FIB	36	ha→m ²	40	J→cal	44	LOAD	48
FILL	36	H _n	40	J→D	44	LOADP	48
FIN (M)	36	H _{nP}	40	J→Wh	44	LOADR	48
FIX	36	hp _{UK} →W	40	J _y (x)	44	LOADSS	48
FLAGS (M)		hp _E →W	40	K (V)	101	LOADV	49
36		hp _M →W	40	KEY	45	LOADΣ	49
FLASH (M)		Hyper _Δ	40	KEY?	45	LocR	49
36		Hyper _Δ	40	KEYG	45	LocR?	49
FLASH?	36	Hyper ⁻¹	40	KEYX	45	LOG ₁₀	49
FLOOR	36	Hyper:	41	kg→ct	45	LOG ₂	49
fm.→m	36	Hyper _p	40	kg→cwt	45	LogF	49
FP	36	HypF	41	kg→lb.	45	Logis _Δ	49
F _p (x)	37	I-	43	kg→oz	45	Logis _Δ	49
FP?	37	I (V)	101	kg→s.t	45	Logis ⁻¹	49
fr→dB	37	i%/a (V)	101	kg→scw	45	Logis:	49
FRACT (S)		I/O (M)	43	kg→sto	45	Logis _p	49
105		I+	43	kg→ton	45	LOG _{xy}	49
FS?	37	IDIV	41	kg→trz	45	LOOP (M)	49
FS?C	37	IDIVR	41	KTYP?	46	LOWBAT (S)	
FS?F	37	IGN1ER (S)		L (V)	101	106	
FS?S	37	108		L.INTS (M)		ly→m	50
ft.→m	37	iHg→Pa	41	50		M.DELR	53
ft _{us} →m	37	Im	41	L.R.	50	M.DIM	53
FV (V)	101	in.→m	42	LASTx	46	M.DIM?	53
fz _{UK} →m ³	37	INC	41	lb.→kg	46	M.DY	53
fz _{us} →m ³	37	INDEX	41	lbft→Nm	46	M.EDI	53
GAP	38	INFO (M)	41	LBL	46	M.EDIN	53
GaussF	38	INPUT	42	LBL?	46	M.EDIT (M)	
GCD	38	INT?	42	LCM	46	53	
g _d	38	INTING (S)		LEAD.0 (S)		M.GET	53
g _d ⁻¹	38	108		106		M.GOTO	54
Geom _Δ	39	INTS (M)	42	LEAP?	47	M.GROW	54
Geom _Δ	39	INVRT	42			M.INSR	54

M.LU	54	MEM?	51	OrthoF	57	PROG (M)	60
M.NEW	54	MENU	51	ORTHOG (M)		PROGS (M)	
M.OLD	54	MENUS (M)		57		60	
M.PUT	54	51		OVERFL (S)		PROPFR (S)	
M.RZR	54	mi. \rightarrow m	52	106		105	
M.SIMQ (M)	55	min	51	oz \rightarrow kg	57	PRTACT (S)	
M.SQR?	55	MIRROR	52	P.FN (M)	61	108	
M.WRAP	55	mmH \rightarrow Pa	52	P.FN2 (M)	61	psi \rightarrow Pa	60
m:	55	MOD	52	P:	61	PSTO	60
m \rightarrow au	55	MODE (M)	52	Pa \rightarrow atm	58	pt. \rightarrow m	61
m \rightarrow fm.	55	MONTH	52	Pa \rightarrow bar	58	PUTK	61
m \rightarrow ft _{us}	55	MSG	52	Pa \rightarrow iHg	58	PV (V)	102
m \rightarrow ft.	55	MULTx (S)		Pa \rightarrow mmH	58	qt. \rightarrow m ³	61
m \rightarrow in.	55	107		Pa \rightarrow psi	58	QUIET (S)	
m \rightarrow ly	55	MUL π	52	Pa \rightarrow tor	58	107	
m \rightarrow mi.	55	MUL π \rightarrow	52	ParabF	58	R \uparrow	69
m \rightarrow nmi.	55	MVAR	52	PARTS (M)		R \rightarrow D	69
m \rightarrow pc	55	MyMenu	52	58		R \downarrow	69
m \rightarrow pt.	55	My α (M)	52	PAUSE	58	RAD	61
m \rightarrow yd.	55	N \rightarrow lbf	57	pc \rightarrow m	58	RAD \rightarrow	61
m 2 \rightarrow ac _{us}	50	NaN?	55	PER/a (V)		RAM (M)	61
m 2 \rightarrow ac	50	NAND	55	102		RAN#	62
m 2 \rightarrow ha	50	NBin _△	56	PERM	58	RANGE	61
m 3 \rightarrow bbl	50	NBin ₋₁	56	PGMINT	58	RANGE?	62
m 3 \rightarrow fz _{us}	50	NBin _△	56	PGMSLV	58	RANI#	62
m 3 \rightarrow fz _{uk}	50	NBIN:	56	PIXEL	59	RBR	62
m 3 \rightarrow gl _{uk}	50	NBin _p	56	PLOT	59	RCL	62
m 3 \rightarrow gl _{us}	50	NEIGHB	56	PMT (V)	102	RCL-	63
m 3 \rightarrow sqt.	50	NEXTP	56	P _n	59	RCL/	63
MANT	50	Nm \rightarrow lbf	56	POINT	59	RCL+	63
MASKL	51	nmi. \rightarrow m	56	Poiss ⁻¹	59	RCLx	63
MASKR	51	NOP	56	Poiss _△	59	RCL \uparrow	63
Mat_A (V)	102	NOR	56	Poiss _△	59	RCL \downarrow	63
Mat_B (V)	102	Norml _△	57	Poiss:	59	RCLCFG	62
Mat_X	51	Norml ₋₁	57	Poiss _p	59	RCLEL	62
Mat_X (V)	102	Norml _△	57	POLAR (S)		RCLIJ	63
Mat_X	102	Norml:	57	105		R-CLR	67
MATR?	51	Norml _p	57	PopLR	59	RCLS	63
MATRS (M)	51	NOT	57	PowerF	59	R-COPY	68
MDY (S)	105	NPER (V)	102	pr \rightarrow dB	60	RDP	63
MAX	51	NUM.IN (S)		PRCL	60	Re	63
MAXT (M)	51	108		PRIME?	60	RE \rightarrow CX	64
max	51	n Σ	57	PRINT (M)	60	Re \leftrightarrow Im	64
MDY (S)	105	ODD?	57	PRINT (S)		REAL?	64
		OFF	57	106		REALDF (V)	
		OR	57	PROB (M)	60	103	

REALS (M)	SETIND	70	STO+	74	TVM (M)	78
63, 64	SETJPN	70	STOx	74	U→ (M)	79
RECV	SETSIG	70	STO↑	75	ULP?	78
REGS (V)	SETTIM	70	sto→kg	75	Un	78
RESET	SETUK	70	STO↓	75	UNDO	78
RJ	SETUSA	70	STOCFG	74	UNITV	78
RL	SF	70	STOEL	74	UNSIGN	79
RLC	SHOW	71	STOIJ	74	USER (S)	106
RM	SIGN	71	STOP	74	V:	79
RM?	SIGNMT	71	STOS	74	V4	79
RMD	SIM_EQ	71	STRI?	75	VAR (M)	79
X_RMS	sin	71	STRING (M)		VARMNU	79
RNORM	sinc	71		75	VARS (M)	79
RootF	sinh	71	SUM	75	VERS?	79
ROUND	SKIP	72	s _w	75	VIEW	79
ROUNDI	SL	72	s _{xy}	75	VMDISP (S)	
RR	SLOW (S)	107	SYS.FL (M)		108	
RRC	s _m	72		75	W ⁻¹	80
RSD	s _{mw}	73	SYSTEM	75	W→hp _{UK}	81
R-SORT	SNAP	73	t ⁻¹ (p)	77	W→hp _E	81
RSUM	SOLVE	73	t _Δ (x)	77	W→hp _M	81
R-SWAP	Solver (M)		t _Δ (x)	77	WDAY	80
RTN	73		t:	78	Weibl _Δ	80
RTN+1	SOLVING (S)		t _Δ :	78	Weibl _Δ ⁻¹	80
RUNIO (S)	108		tan	76	Weibl _Δ	80
106	SPCRES(S)		tanh	76	Weibl:	80
RUNTIM (S)	107		TDISP	76	Weibl _p	80
106	SPEC?	73	TDM24 (S)		Wh→J	80
s	SR	73		104	WHO?	80
s(a)	SSIZE?	73	TEST (M)	76	W _m	80
S.INTS (M)	SSIZE8 (S)		TICKS	76	W _p	80
76	107		TIME	77	WSIZE	81
s.t→kg	ST.A (V)	103	TIMER	77	WSIZE?	81
s→year	ST.B (V)	103	TIMES (M)		x < ?	84
SAVE	ST.C (V)	103		77	x = ?	84
SB	ST.D (V)	103	T _n	77	x ≠ ?	84
SCI	ST.X (V)	103	ton→kg	77	x = -0?	84
scw→kg	ST.Z (V)	103	TONE	77	x = +0?	84
SDIGS?	ST.T (V)	103	TOP?	77	x > ?	84
SDL	ST.Y (V)	103	tor→Pa	77	x ≈ ?	84
SDR	STAT (M)	74	t _p (x)	77	x ≤ ?	84
SEED	STATUS	74	TRACE (S)		x ≥ ?	84
SEND	STK (M)	74		106	x!	83
SETCHN	STO	74	TRANS	77	X.FN (M)	83
SETDAT	STO-	74	TRI (M)	78	x:	83
SETEUR	STO/	74	trz→kg	78	x→DATE	83

$x \rightarrow \alpha$	84	$\alpha \rightarrow z$	85	ε_m	88	σ_w	89
$x \rightarrow x$	84	$\alpha \cdot (\text{M})$	86	ε_p	88	Σx	89
$x \rightarrow y$	84	$\alpha.\text{FN} (\text{M})$	86	$\zeta(x)$	88	$\Sigma x/y$	89
x^2	81	$\alpha \rightarrow x$	87	$\pi (\text{C})$	88	Σx^2	89
x^3	81	$\alpha\text{CAP} (\text{S})$	106	Π_n	88	$\Sigma x^2/y$	89
XEQ	81	$\alpha\text{INTL} (\text{M})$	85	σ	88	$\Sigma x^3/y$	89
xIN	82	$\alpha\text{LENG?}$	85	$\Sigma-$	90	$\Sigma x^4/y$	89
x_{\max}	82	$\alpha\text{MATH} (\text{M})$		$\Sigma (\text{M})$	88	$\Sigma x\ln y$	89
x_{\min}	82		86	$\Sigma 1/x$	88	Σxy	89
$XNOR$	82	$\alpha\text{POS?}$	86	$\Sigma 1/x^2$	88	Σy	90
XOR	82	αRL	86	$\Sigma 1/y$	88	Σy^2	90
$xOUT$	82	αRR	86	$\Sigma 1/y^2$	88	$\Sigma y\ln x$	90
\hat{y}	85	αSL	86	$\Sigma +$	90	$\chi^2_{\Delta}(x)$	90
$Y.MD$	85	αSR	86	$\Sigma \ln^2 x$	89	$\chi^2_{\Delta}(x)$	90
$y \hat{x}$	85	$\Gamma(x)$	87	$\Sigma \ln^2 y$	89	$X^2:$	90
$yd \rightarrow m$	85	γ_{xy}	87	$\Sigma \ln x$	89	$\chi^2_p(x)$	90
$YEAR$	85	Γ_{xy}	87	$\Sigma \ln xy$	89		
$year \rightarrow s$	85	$\Delta\%$	87	$\Sigma \ln y$	89		
$YMD (\text{S})$	105	$\delta x (\text{L})$	87	$\Sigma \ln y/x$	89		
y^x	85	ε	87	Σ_n	89		

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

${}^\circ C \rightarrow {}^\circ F$	16	ALL	19	AUTXEQ (S)	107	Binom _p	23
${}^\circ F \rightarrow {}^\circ C$	16	ALLENG (S)	107	au \rightarrow m	21	BITS	23
10^x	16	ALPHA (S)	106	A... Ω (M)	86	B_n	23
1COMPL	16	ALP.IN (S)	108	A:	21	B_n^*	23
$1/x$	16	AND	19	B (V)	100	BS?	23
2COMPL	16	ANGLES (M)	19	BACK	21	Btu \rightarrow J	23
2^x	16	arccos	19	bar \rightarrow Pa	21	C (V)	100
$3\sqrt{x}$	16	arcosh	19	BATT?	21	cal \rightarrow J	24
A (V)	100	arcsin	20	bbl \rightarrow m ³	21	CARRY (S)	106
ABS	18	arctan	20	BC?	21	CASE	24
ac _{us} \rightarrow m ²	18	artanh	20	BEEP	21	CATALOG	24
ac \rightarrow m ²	18	ASIN	20	BeginP	22	Cauch ⁻¹	25
ACC (V)	100	ASL.BLK (S)	108	BestF	22	Cauch _{Δ}	25
ACOS	18	ASR	20	BestF?	23	Cauch _{Δ}	25
ADM (V)	100	ASSIGN	20	Binom _{Δ}	23	Cauch:	25
ADV	18	ATAN	20	Binom ⁻¹	23	CauchF	25
AGM	18	atm \rightarrow Pa	21	Binom _{Δ}	23	Cauch _p	25
AGRAPH	18	AUTOFF (S)	107	Binom:	23	CB	25

CEIL	25	DAY	29	ENORM	33	FLOOR	36
CF	25	DBLR	29	ENTER†	33	fm. \rightarrow m	36
CHARS (M)	25	DBLx	29	ENTRY?	33	FP	36
CLALL	25	DBL/	29	EQN (M)	33	F _p (x)	36
CLCVAR	26	dB \rightarrow fr	29	EQ.DEL	33	FP?	36
CLFALL	26	dB \rightarrow pr	29	EQ.EDI	33	fr \rightarrow dB	36
CLK (M)	26	DEC	29	EQ.NEW	33	FRACT (S)	103
CLLCD	26	DECIM. (S)	105	erf	33	FS?	36
CLMENU	26	DECOMP	29	erfc	33	FS?C	36
CLP	26	DEG	30	ERR	34	FS?F	36
CLPALL	26	DEG \rightarrow	30	EVEN?	34	FS?S	36
CLR (M)	26	DENANY (S)	103	EXITALL	34	ft. \rightarrow m	36
CLREGS	26	DENFIX (S)	103	EXP (M)	34	ft _{us} \rightarrow m	36
CLSTK	26	DENMAX	30	ExpF	34	FV (V)	99
CLX	27	DENMAX (V)	98	Expon	34	fz _{uk} \rightarrow m ³	36
CLΣ	27	DET	30	Expone	34	fz _{us} \rightarrow m ³	36
CNST	27	DIGITS (M)	30	Expon _p	34	F Δ (x)	36
COMB	27	DISP (M)	30	Expon $^{-1}$	34	F Δ (x)	36
CONJ	27	DMY (S)	102	Expon:	34	F $^{-1}$ (p)	36
CONST (M)	27	DOT	30	EXPT	34	F:	36
CONVG	27	DROP	30	e x	34	f' (M)	37
CORR	28	DROPy	30	e $^{x-1}$	34	f' (x)	37
cos	28	DSE	31	E:	34	f" (M)	37
cosh	28	DSL	31	FB	35	f"(x)	37
COV	28	DSTACK	31	FBR	35	F&p:	37
CPX (M)	28	DSZ	31	FC?	35	GAP	37
CPXj (S)	105	D.MS	31	FC?C	35	GaussF	37
CPXRES (S)	105	D.MS \rightarrow	32	FC?F	35	GCD	37
CPXS (M)	28	D.MS \rightarrow D	32	FC?S	35	g _d	38
CPX?	28	D.MY	32	FCNS (M)	35	g _d $^{-1}$	38
CROSS	29	D \rightarrow D.MS	32	FF	35	Geom _p	38
ct \rightarrow kg	29	D \rightarrow J	32	FIB	35	Geom Δ	38
cwt \rightarrow kg	29	D \rightarrow R	32	FILL	35	Geom Δ	38
CX \rightarrow RE	29	EIGVAL	32	FIN (M)	35	Geom $^{-1}$	38
D (V)	98	EIGVEC	32	FIX	35	Geom:	38
DATE	28	END	32	FLAGS (M)	35	gl _{us} \rightarrow m ³	38
DATES (M)	28	ENDP	32	FLASH (M)	35	gl _{uk} \rightarrow m ³	38
DATE \rightarrow	28	ENG	33	FLASH?	35	GRAD	38

GRAD→	38	Γ_p	42	LEAD.0 (S)	104	$m^2 \rightarrow ac_{us}$	48
GRAMOD (V)	99	Γ_q	42	LEAP?	45	$m^2 \rightarrow ha$	48
GROW (S)	105	I+	42	$LgNrm_p$	45	$m^3 \rightarrow bbl$	48
GTO	38	I-	42	$LgNrm_{\Delta}$	45	$m^3 \rightarrow fz_{uk}$	48
GTO.	39	I/O (M)	42	$LgNrm_{\Delta^{-1}}$	45	$m^3 \rightarrow fz_{us}$	48
ha→ m^2	39	i%/a (V)	99	$LgNrm:$	45	$m^3 \rightarrow gl_{uk}$	48
H_n	39	J (V)	99	LinF	45	$m^3 \rightarrow gl_{us}$	48
H_{np}	39	$J_y(x)$	42	LJ	46	MANT	49
$hp_E \rightarrow W$	39	J+	42	L _m	46	MASKL	49
$hp_H \rightarrow W$	39	J-	42	L _{ma}	46	MASKR	49
$hp_{uk} \rightarrow W$	40	J/G	42	LN	46	MATRS (M)	49
Hyper _p	39	J→Btu	43	LN β	46	MATR?	49
Hyper _Δ	39	J→cal	43	LNΓ	46	MATX (M)	49
Hyper _{Δ^{-1}}	39	J→D	43	LN(1+x)	46	Mat_A (V)	100
Hyper ₋₁	39	J→Wh	43	LOAD	46	Mat_B (V)	100
Hyper:	40	K (V)	99	LOADP	47	Mat_X	49
HypF	40	KEY	43	LOADR	47	Mat_X (V)	100
I (V)	99	KEYG	43	LOADSS	47	max	49
IDIV	40	KEYX	43	LOADV	47	MDY (S)	102
IDIVR	40	KEY?	43	LOADΣ	47	MEM?	49
IGN1ER (S)	106	kg→ct	44	LocR	47	MENU	49
iHg→Pa	40	kg→cwt	44	LocR?	47	MENUS (M)	50
Im	40	kg→lb.	44	LOG ₁₀	47	min	50
INC	40	kg→oz	44	LOG ₂	47	MIRROR	50
INDEX	40	kg→scw	44	LogF	48	mi.→m	50
INFO (M)	40	kg→sto	44	Logis _p	48	mmH→Pa	50
INPUT	41	kg→s.t	44	Logis _Δ	48	MOD	50
INT?	41	kg→ton	44	Logis _{Δ^{-1}}	48	MODE (M)	50
INTING (S)	106	kg→trz	44	Logis:	48	MONTH	50
INTS (M)	41	KTYP?	44	LOG _{xy}	48	MSG	50
INVRT	41	L (V)	99	LOOP (M)	48	MULTx (S)	105
in.→m	41	LASTx	44	LOWBAT (S)	104	MULπ	50
IP	41	lb.→kg	44	ly→m	48	MULπ→	50
ISE	41	lbf→N	44	L.INTS (M)	48	MVAR	50
ISG	42	lbft→Nm	44	L.R.	48	MyMenu	51
ISM (V)	101	LBL	45	$m^2 \rightarrow ac$	48	Myα (M)	51
ISZ	42	LBL?	45			M.DELR	51
I _{xyz}	42	LCM	45			M.DIM	51

M.DIM?	51	nmi. \rightarrow m	54	P _n	57	RCL	60
M.DY	51	Nm \rightarrow lbft	55	POINT	57	RCLCFG	60
M.EDI	51	NOP	55	Poiss _p	57	RCLEI	60
M.EDIN	51	NOR	55	Poiss _A	57	RCLIJ	61
M.EDIT (M)	51	Norml _p	55	Poiss _A ⁻¹	57	RCLS	61
M.GET	52	Norml _A	55	Poiss _A ⁻¹	57	RCL+	61
M.GOTO	52	Norml _A ⁻¹	55	Poiss:	57	RCL-	61
M.GROW	52	Norml _A ⁻¹	55	POLAR (S)	103	RCLx	61
M.INSR	52	Norml _A :	55	PopLR	57	RCL /	61
M.LU	52	NOT	55	PowerF	57	RCL \uparrow	61
M.NEW	52	NPER (V)	100	PRCL	58	RCL \downarrow	61
M.OLD	52	NUM.IN (S)	105	PRIME?	58	RDP	61
M.PUT	52	n Σ	55	PRINT (M)	58	Re	61
M.R \Rightarrow R	53	N \rightarrow lbf	55	PRINT (S)	104	REAL?	62
M.SIMQ (M)	53	ODD?	55	PROB (M)	58	REALDF (V)	101
M.SQR?	53	OFF	55	PROG (M)	58	REALS (M)	62
M.WRAP	53	OR	55	PROGS (M)	58	RECV	62
m:	53	OrthoF	55	PROPFR (S)	103	REGS (V)	101
m \rightarrow au	53	ORTHOG (M)	55	PRTACT (S)	105	RESET	62
m \rightarrow fm.	53	OVERFL (S)	104	pr \rightarrow dB	58	RE \rightarrow CX	62
m \rightarrow ft _{us}	53	oz \rightarrow kg	56	psi \rightarrow Pa	58	Re \rightarrow Im	62
m \rightarrow ft.	53	ParabF	56	PSTO	58	RJ	63
m \rightarrow in.	53	PARTS (M)	56	pt. \rightarrow m	59	RL	63
m \rightarrow ly	53	PAUSE	56	PUTK	59	RLC	63
m \rightarrow mi.	53	Pa \rightarrow atm	56	PV (V)	100	RM	63
m \rightarrow nmi.	53	Pa \rightarrow bar	56	P.FN (M)	59	RMD	64
m \rightarrow pc	53	Pa \rightarrow iHg	56	P.FN2 (M)	59	RM?	64
m \rightarrow pt.	53	Pa \rightarrow mmH	56	P:	59	RNORM	64
m \rightarrow yd.	53	Pa \rightarrow psi	56	qt. \rightarrow m ³	59	RootF	64
NAND	54	Pa \rightarrow tor	56	QUIET (S)	105	ROUND	64
NaN?	54	pc \rightarrow m	56	RAD	59	ROUNDI	64
NBin _p	54	PERM	56	RAD \rightarrow	59	RR	64
NBin _A	54	PER/a (V)	100	RAM (M)	59	RRC	65
NBin _A ⁻¹	54	PGMINT	57	RANGE	59	RSD	65
NBin _A ⁻¹	54	PGMSLV	57	RANGE?	60	RSUM	65
NBin:	54	PIXEL	57	RANI#	60	RTN	65
NEIGHB	54	PLOT	57	RAN#	60	RTN+1	65
NEXTP	54	PMT (V)	100	RBR	60	RUNIO (S)	104

RUNTIM (S)	104	s_m	70	SUM	73	UNDO	77
R-CLR	65	SMODE?	71	s_w	73	UNSIGN	77
R-COPY	66	s_{mw}	71	s_{xy}	74	USER (S)	104
R-SORT	66	SNAP	71	SYSTEM	74	U \rightarrow (M)	77
R-SWAP	66	SOLVE	71	SYS.FL (M)	74	VAR (M)	77
R \rightarrow D	67	Solver (M)	71	s(a)	74	VARMNU	77
R \uparrow	67	SOLVING (S)	106	S.INTS (M)	74	VARS (M)	77
R \downarrow	67	SPCRES (S)	104	s.t \rightarrow kg	74	VERS?	77
s	67	SPEC?	71	s \rightarrow year	74	VIEW	77
SAVE	67	SR	72	T_0	74	VMDISP (S)	106
SB	67	SSIZE8 (S)	105	tan	74	V:	77
SCI	67	SSIZE?	72	tanh	74	V \downarrow	78
scw \rightarrow kg	67	STAT (M)	72	TDISP	75	WDAY	78
SDIGS?	67	STATUS	72	TDM24 (S)	102	Weibl $_p$	78
SDL	67	STK (M)	72	TEST (M)	75	Weibl $_{\Delta}$	78
SDR	67	STO	72	TICKS	75	Weibl $_{\Delta}$	78
SEED	68	STOCFG	72	TIME	75	Weibl $^{-1}$	78
SEND	68	STOEL	72	TIMER	75	Weibl:	78
SETCHN	68	STOIJ	72	TIMES (M)	75	WHO?	78
SETDAT	68	STOP	72	T_n	75	Wh \rightarrow J	78
SETEUR	68	STOS	72	TONE	75	W $_m$	78
SETIND	68	STO+	73	ton \rightarrow kg	75	W $_p$	78
SETJPN	68	STO-	73	TOP?	75	WSIZE	79
SETSIG	68	STO \times	73	tor \rightarrow Pa	76	WSIZE?	79
SETTIM	68	STO/	73	t $_p$ (x)	76	W $^{-1}$	79
SETUK	68	STO \uparrow	73	TRACE (S)	104	W \rightarrow hp $_{UK}$	79
SETUSA	68	sto \rightarrow kg	73	TRANS	76	W \rightarrow hp $_{E}$	79
SF	68	STO \downarrow	73	TRI (M)	76	W \rightarrow hp $_{M}$	79
SHOW	69	STRING (M)	73	trz \rightarrow kg	76	\hat{x}	79
SIGN	69	STRI?	73	TVM (M)	76	\bar{x}	79
SIGNMT	69	ST.A (V)	101	t $_{\Delta}$ (x)	76	x 2	79
SIM_EQ	69	ST.B (V)	101	t $_{\Delta}$ (x)	76	x 3	79
sin	69	ST.C (V)	101	t $^{-1}$ (p)	76	XEQ	79
sinc	69	ST.D (V)	101	t:	76	\bar{x}_G	80
sinh	69	ST.T (V)	101	t \gtrless	76	\bar{x}_H	80
SKIP	70	ST.X (V)	101	ULP?	76	xIN	80
SL	70	ST.Y (V)	101	U $_n$	76	x $_{max}$	80
SLOW (S)	104	ST.Z (V)	101	UNITY	77	x $_{min}$	80

XNOR	80	αSL	84	Σlny	87	Σ	92
XOR	80	αSR	84	Σxy	87	M	92
xOUT	80	α· (M)	84	Σy	87	x	92
✗RMS	80	α.FN (M)	84	Σy ²	87		93
✗w	81	A...Ω (M)	84	Σlnx	87	%	93
✗y	81	α→x	84	Σ+	88	%MRR	93
x!	81	β(x,y)	84	Σ-	88	%T	93
X.FN (M)	81	Γ _{xy}	84	χ ² _p (x)	88	%Σ	93
x:	81	γ _{xy}	84	χ ² _△ (x)	88	%+MG	93
x→DATE	81	Γ(x)	85	χ ² _△ (x)	88	✓x	94
x→α	81	δx (L)	85	χ ² :	88	∫	94
x✗	81	Δ%	85	(χ ²) ⁻¹	88	∫f (M)	94
x✗y	82	ε	85	(-1) ^x	88	∫fdx (M)	94
x < ?	82	ε _m	85	[M] ^T	88	↳	94
x ≤ ?	82	ε _p	85	[M] ⁻¹	89	↳ (M)	95
x = ?	82	ζ(x)	85	+	89	ADV	95
x ≠ ?	82	π	86	+/-	89	CHAR	95
x ≈ ?	82	Π _n	86	±∞?	89	DLAY	95
x ≥ ?	82	Σ (M)	86	-	89	LCD	95
x > ?	82	σ	86	×	89	MODE	95
x = +0?	82	Σ 1/x	86	×MOD	89	PROG	95
x = -0?	82	Σ 1/x ²	86	/	89	r	96
ŷ	82	Σ 1/y	86	^MOD	89	REGS	96
yd.→m	82	Σ 1/y ²	86	→ (c)	90	STK	96
YEAR	82	Σln ² x	86	→DATE	90	TAB	96
year→s	83	Σln ² y	86	→DEG	90	USER	96
YMD (S)	102	Σlnx	86	→D.MS	90	WIDTH	96
y ^x	83	Σlnxy	86	→GRAD	90	Σ	97
Y.MD	83	Σlny	86	→HR	90	#	97
y✗	83	Σlny/x	86	→H.MS	90	# a	131
z✗	83	Σ _n	87	→INT	90	# a ₀	131
αCAP (S)	104	σ _w	87	→MULπ	90	# a _{Moon}	131
αINTL (M)	83	Σx	87	→POL	90	# a _⊕	132
αLENG?	83	Σx ²	87	→RAD	91	# c	132
αMATH (M)	83	Σx ² y	87	→REAL	91	# c ₁	132
αPOS?	83	Σx ² /y	87	→REC	91	# c ₂	132
αRL	84	Σx ³	87	↑Lim (V)	102	# e	132
αRR	84	Σx ⁴	87	↓Lim (V)	102		

# e_E	132	# m_n/m_p	133	# R_\oplus	135	# λ_{Cn}	136
# F	132	# m_p	134	# Sa	135	# λ_{Cp}	136
# F_α	132	# m_p/m_e	134	# Sb	135	# μ_0	137
# F_δ	132	# m_{PL}	134	# Se^2	135	# μ_B	137
# G	132	# m_u	134	# Se'^2	135	# μ_e	137
# G_0	132	# $m_u c^2$	134	# Sf^{-1}	135	# μ_e/μ_B	137
# G_C	132	# m_μ	134	# T_0	135	# μ_n	137
# g_e	133	# M_\oplus	134	# T_p	135	# μ_p	137
# GM_\oplus	133	# M_\odot	134	# t_{PL}	135	# μ_u	137
# g_\oplus	133	# N_A	134	# V_m	136	# μ_ν	137
# h	133	# NaN	134	# Z_0	136	# σ_B	137
# \hbar	133	# p_0	134	# α	136	# Φ	137
# k	133	# R	134	# γ	136	# Φ_0	137
# K_J	133	# r_e	134	# γ_{EM}	136	# ω	138
# I_{PL}	133	# R_K	135	# γ_p	136	# $-\infty$	138
# m_e	133	# R_{Moon}	135	# Δv_{Cs}	136	# ∞	138
# M_{Moon}	133	# R_∞	135	# ε_0	136	# B	97
# m_n	133	# R_\odot	135	# λ_c	136	# DEC (V)	102

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 Museum of HP Calculators (https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234685#234685). 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 Jake's feedback and further thoughts, distributed to <i>Eric</i> , <i>Jake</i> , <i>Marcus</i> , and <i>Pauli</i> .
0.3	21.3.16	Split the manual in three; moved LBL onto the keyboard, renamed STOM to STOCFG, RCLM to RCLCFG, SERR to s_m , and SERR _w to s_{mw} ; refined the <i>Key Response Table</i> . Passed to <i>Michael</i> for information.
0.4	28.3.16	Renamed LOGS to EXP and EEX to E . Added hardware information from 2 nd manufacturer.
0.5	29.10.16	Returned EEX . Changed keyboard layout .
0.6	22.8.17	Merged the Applications and Owner's Manual. Changed the input order of complex number parts on Pauli's request. Changed keyboard layout introducing D.MS, SST, BST, and % while removing ÿ, RAN#, 'FRC, and 'CFIT. Put 'CFIT into 'STAT and 'FRC into 'MODE. Placed OFF below EXIT for easier customizing. Renamed cc to C5, EEX to E , STOPW to TIMER, SHOW to REGS, 'SOLVE to 'ADV, DLINES to DSTACK, 12h to CLK12, and 24h to CLK24. Replaced IND by →. Deleted %MG since covered by Δ%, added EIGVAL and EIGVEC. Swapped CNST and CONST. Defined the echo rows for alphanumeric and command input. Expanded and modified the character sets for better use of display space. Added the QRG.
0.7		Changed keyboard layout. Replaced the labels BST by ≡Δ , SST by ≡▼ , and UNDO by ↶ ; added some <i>alpha input mode</i> reminders on the keyboard. Added AGRAPH, CLLCD, EQ.xxx, HYP , J/G, M.GOTO, ORTHOF, PIXEL, POINT, TDISP, and USER . Moved the background considerations out of <i>ReM App. D</i> . Introduced K as <i>alpha register</i> for alphanumeric constants in programs. Removed <i>fraction data type</i> . Extended <i>items</i> from 6 to 7 characters to match 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 <i>HP-42S</i> .
0.8	Changed keyboard layout: introduced <u>TRG</u> containing trigonometric functions, removed <u>HYP</u> into <u>EXP</u> and $\blacksquare\pi$ to g-shifted $\blacksquare\theta$, 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.
7.5.18	
20.9.18	Corrected errors and inconsistencies. Added one more example. Moved the key response table into an appendix.
0.9	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 <i>DP</i> 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>$\blacksquare\chi$</u> to match <i>HP-42S</i> . Added a chapter about curve fitting. Modified functionalities of <u>ENTER</u> ↑ and $\blacksquare\leftarrow$. Expanded <i>App. K</i> . Renamed DOUBLE to →DP. Added →SP and conversions of <i>quarts</i> . Rearranged <u>X.FN</u> . Replaced <u>USR</u> by <u>UM</u> . Changed keyboard moving <u>UM</u> , <u>$\blacksquare X$</u> , and <u>TRI</u> . Moved \blacksquare to $\blacksquare f$ <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.
3.1.19	
0.10	Returned <i>angle data type</i> and α SR. Added IDIVR and VANGLE. Refined FP, IP, IMPFRC, PROFRC, SDIGS?, →DP, →HR, →INT, →REAL, →SP, explanation of ALL, the summary of integer functions, and handling of long alpha strings. Modified contents of <u>CPX</u> , <u>MATX</u> , and <u>α</u> . Added a summary of matrix functions. Removed the <u>ON</u> -key combinations. Modified MEM?. Rewrote the angular conversions. Renamed infinite and finite integers to <i>long</i> and <i>short integers</i> . Added a chapter about $\pm\infty$ and NaN. Modified RBR and the menu for STO and RCL. Removed \blacksquare from the keyboard. Renamed X_u to X_e for the distributions.
3.3.19	

	Date	Release notes
0.11	8.5.19	Changed keyboard making CC primary and user mode shifted, removing x^2 , $x\bar{x}$, 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> , <u>SHOW</u> , <u>STAT</u> , <u>U\rightarrow</u> , <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_i} , \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} , Π to Π_n , Σ to Σ_n , and some constants to avoid search ambiguities. Refined App. J, Sect. 3 and 4, \rightarrow INT, <u>CLR</u> , and the functions of Δ and ∇ . 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> , <u>RTN</u> , <u>SHOW</u> , <u>STAT</u> , and <u>a.FN</u> . Rearranged A...Q and Sect. 2 of the OM.
0.12	16.10.19	Rearranged the appendices of the ReM from App. D on. Expanded App. A of the OM and App. K. Deleted the standardized normal distribution Φ and rearranged <u>PROB</u> . Updated <u>CNST</u> following CODATA 2018. Renamed the angular conversions. Changed the composing and cutting functionality of CC . Refined exiting <i>short integer</i> input. Expanded App. D. Specified maximum size of <i>long integers</i> . Changed keyboard adding Δ , moving <u>CPX</u> , <u>FIN</u> , <u>RBR</u> , <u>R\uparrow</u> , and <u>SHOW</u> , removing $\%$. Renamed VANGLE to V Δ . 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 $10^{\pm 999}$. 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\bullet</u> , <u>SHOW</u> , and <u>STATUS</u> . Refined the sorting order of <i>items</i> , ALL, CX \rightarrow RE, MEM?, RE \rightarrow CX, RBR, RM, SLVQ, and <u>U\rightarrow</u> . Started filling App. F and G. Refined App. 2. Added a <i>long integer</i> example, CPXR?, LZ?, Δv_{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'SYS.FL</u> , <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>U\rightarrow</u> and <u>$\Delta$$\rightarrow$</u> , moving <u>CPX</u> , <u>FILL</u> , <u>RBR</u> , <u>R\uparrow</u> , <u>USER</u> , <u>a.FN</u> , <u>aINTL</u> , <u>\sqrt{x}</u> , and <u>\neg</u> , displaying <u>PRINT</u> , <u>RMD</u> , <u>STATUS</u> , x^2 , and $'$, and removing <u>c/d</u> , <u>$\neg x$</u> , \rightarrow SP, and \rightarrow DP. Renamed <u>DISP</u> to <u>DSP</u> and <u>SUMS</u> to <u>Σ</u> , changed <u>\leftrightarrow</u> to <u>\leftrightarrow</u> . Refined the addressing tables and catalog access, <u>a b/c</u> , <u>ADV</u> , <u>BATT?</u> ,

Date	Release notes
	<p><u>BITS</u>, <u>CATALOG'CHARS</u> and '<u>MENUS</u>, CLALL, CLFALL, <u>CPX</u>, <u>EXP</u>, GAP, <u>INTS</u>, <u>I/O</u>, <u>MODE</u>, NEIGHB, <u>PARTS</u>, PRIME?, <u>P.FN</u>, SHOW, <u>STAT</u>, <u>STK</u>, <u>X.FN</u>, <u>gINTL</u>, and <u>g•</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 14.6.20	<p>Added BESTF?, RANGE, RANGE?, <u>REGIST</u>, SNAP, and s(a), as well as errors 28 and 31 – 35. Changed DSZ and ISZ to comply with HP-16C. Changed keyboard shifting N, O, P, and Q, swapping ? and Z, moving <u>CNST</u>, <u>CPX</u>, <u>FLAGS</u>, <u>RBR</u>, <u>RTN</u>, <u>R↑</u>, <u>VIEW</u>, and <u>█</u>, removing :, and adding MOD, ✓, and SNAP. Renamed <u>DSP</u> to <u>DISP</u>, <u>CNST</u> to <u>CONST</u>, CONST to CNST, ASL.BLK to ASLIFT, SSIZE to SSIZE8, TDM to TDM24, and the left and right sided probabilities. Refined ASSIGN, <u>CATALOG</u>, <u>CNST</u>, <u>DISP</u>, <u>INFO</u>, NEXTP, PRIME?, <u>PROB</u>, RBR, RESET, SHOW, SINC, <u>STAT</u>, <u>U→</u>, VIEW, $x=+0?$, $x=-0?$, y^x, $\alpha \rightarrow x$, 4, pp. 54 – 57 and 205 – 207 (and consequences) as well as <i>Section 6</i> of the OM, pp. 108 – 117, App. B, C, and E of the ReM, and some looping and statistical explanations. Reduced the maximum number of <i>local registers</i> from 100 to 99. Changed ALLSCI to ALLENG and RECTN to POLAR. Added <i>data type</i> matrices for powers. Corrected.</p>
0.16 24.10.20	<p>Added torque and mmHg conversions, x_{\max} and x_{\min}, ISM, and LOADV. Added UNDO to the <u>I/O</u>. Refined <u>I/O</u> and the descriptions of LOAD, LOADSS, RESET, and UNDO. Marked the not-undoable <i>items</i> in the <u>I/O</u>. Renamed the constants according to the OM and kicked them out of the <u>I/O</u>. Added USB. Refined <i>Basic Kinds of Program Steps</i>, App. E and F. Changed bit numbering from 1 ... 64 to 0 ... 63. Renamed SMODE? to ISM?. Refined IM, RE, SINC, <u>TRI</u>, WSIZE, X.FN and the labels of torque conversions. Added SINCπ. Expanded App. G. 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 *GP 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 **x↔ nn** swaps x and the contents of **Rnn**.

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

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

Programs consist of ≥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 (CF POLAR and press **1.23 [CC] 4.56 [+/-] [ENTER]**) or its magnitude and phase like $-7.89 \angle 120^\circ$ in polar mode (SF POLAR 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 π* , or *grades*. 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 number 2 ... 16 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 (a.k.a. text) strings: Enter *alpha input mode* (AIM) by pressing **[@]**. Data entered in AIM become an *alphanumeric string* when closed (unless they are function parameters). All available Latin letters (incl. 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 *grades*, 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	SCI	ENG	ALL	ROUNDI	ROUND	

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

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

ALL *n* Displays all digits present 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 number of stack registers to be displayed (1 ... 4).

EXECUTING FUNCTIONS AND PROGRAMS

Any function or program can be executed via **[XEQ] α name [ENTER↑]** where **name** is the function *name* or the program label. If **name** is not unique, the global label closest to the permanent end (.END.) has precedence. If **name** is a local label, WP 43S searches in 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 **$\equiv \triangleright$** (or **\triangleright** if no multi-view menu is displayed). Press **$\equiv \Delta$** (or **Δ**) 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] α ABC [ENTER↑]**.

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

Indirect addressing: Rather than 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

CLR	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 (≤ 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 _{yx}	P _{yx}	F:	χ^2 :	
Binom:	Binom _p		Binom _▲	Binom _▲		Binom ⁻¹	

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

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

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

Following naming convention holds for most distributions, e.g. for the *normal d.*: **Norml_p** denotes the *probability density function*, **Norml_▲** the *cumulated d. function*, **Norml_▲** the *error probability*, and **Norml⁻¹** 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}	x_{max}	x_{min}		OrthoF	
		\bar{x}_H					
	L.R.	r	s_{xy}	cov	\hat{x}	\hat{y}	
STAT	CLΣ	\bar{x}_G	ε	ε_p	ε_m	PLOT	
	Σ^-	\bar{x}_w	s_w	σ_w	s_{mw}		
	Σ^+	\bar{x}	s	σ	s_m	SUM	

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

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

\bar{x}_G , ε , ε_p , ε_m Geometric mean value, sample scattering factor (*SF*), population *SF*, *SF* of the mean.

\bar{x}_H , \bar{x}_{RMS} , x_{max} , x_{min} Harmonic and quadratic means, max. and min. values.

SUM Recalls Σy and Σx .

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).

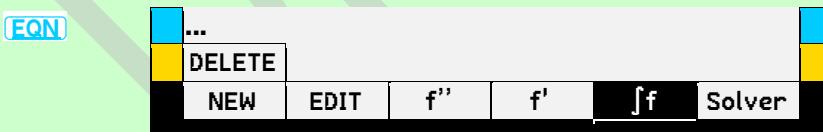
r	Returns the correlation coefficient.
s_{xy} , cov	Return the sample or population covariance.
\hat{x}, \hat{y}	Return the forecast for x or y according to the fit model selected.
LinF	Linear fit model: $y = a_0 + a_1x$.
ExpF	Exponential fit model: $\ln(y) = \ln(a_0) + a_1x$ or $y = a_0 e^{a_1 x}$.
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)				
SOLVE	SLVQ	f'(x)	Π_n	Σ_n	$\int f dx$	

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

Π_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).

Σ_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_{dx} 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_{dx}**. 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	
BITS	SB	BS?	#B	FB	BC?	CB	
	NAND	NOR	XNOR		MIRROR	ASR	
	AND	OR	XOR	NOT	MASKL	MASKR	

AND, OR, XOR, NAND, NOR, XNOR	Boole's binary operators.
NOT	Inverts every bit in X .
MASKL, MASKR	Create masks of x bits on the left (or right) side.
MIRROR	Reflects all bits.
ASR n	Arithmetic shift x right by n places.
SB, FB, or CB n	Sets, flips, or clears bit # n in x (counting starts with #0).
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 to 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; and 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↑**, **x²y**, **+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^x**, **x²**, **fx**, **e^x**, **ln**, **10^x**, **lg**, **sin**, **cos**, and **tan** are the most popular – twelve candidates for seven key tops left. Either **x²** or **fx** shall be primary (we chose **x²** since a shifted **x²** is of little use); and we can ditch **10^x** and **lg** when we have **e^x** and **ln** primary. So **R↓**, **1/x**, **y^x**, **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^x**, 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 is 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. 122ff):

<u>ADV</u>	SOLVE	SLVQ	$f'(x)$	Π_n	Σ_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?	ISM?	WSIZE?	KTYP?
<u>INTS</u>	A	B	C	D	E	F
<u>I/O</u>	LOAD	LOADP	LOADR	LOADSS	LOADV	LOADS
<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:	χ^2 :
<u>P.FN</u>	INPUT	END	ERR	TICKS	PAUSE	P.FN2
<u>STAT</u>	Σ+	̄x	s	g	s _m	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 \rightarrow s	F&p:	m:	x:
<u>X.FN</u>	AGM	B _n	B _n *	erf	erfc	Orthog
<u>α.FN</u>	x \rightarrow α	αRL	αRR	αSL	αSR	α \rightarrow x
<u>Σ</u>	n	Σx	Σx ²	Σxy	Σy ²	Σy
<u>→</u>	→DEG	→RAD	→GRAD		→D.MS	→MULπ

All these functions can be accessed via a single keystroke if and when their *menu* is open, else via three keystrokes. Thus, repeating any unshifted *softkey* as a shifted label on the keyboard is of limited value; and there is just one example where this is done (\sqrt{x} and INT_x).

$|x|$ and INT_x are also featured as shifted *softkeys*: accessing $|x|$ or INT_x needs two keystrokes while $|x|$ and INT_x require four maximum (and two if the *menu* is open). In consequence, $|x|$ and INT_x are actually not needed in any *menu* but are left in CPX and PARTS since space is available there.

See also *Layouting* on pp. B-18f.

Alpha Register

For long I thought we could do without a dedicated *alpha register* since each and every *register* is capable holding an *alphanumeric 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 *DT* 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 *DT* 4 returned with v0.10 for sake of keeping algebraic operations simple and avoiding special purpose commands like D.MS+, D.MS-, etc.

Actually, *angles* are displayed in five ‘modes’ (*decimal* and *sexagesimal degrees*, *radians*, *multiples of π* , and *gon* or *grades*). 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 need for high precision storage of π for modulo calculations and reduction of rounding errors. And, alas, *radians* are inevitable since Taylor series for trigonometric functions are written for angular input in *radians*.

Generally, trigonometric functions shall actually operate on *angles* within $\pm 180^\circ$ only; thus, angular input beyond this range shall be reduced modulo 360° , then minus 180° (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 2π suffice to warrant 16 digits accuracy of respective function results for the number range of *single precision reals*.⁹⁶ *WP 43S* uses 1065 digits for 2π 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 *DT* concept and had to be eventually abandoned officially when introducing named *system flags* with v0.14.

⁹⁶ See <https://forum.swissmicrom.com/viewtopic.php?f=2&t=350#p4349>.

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

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. Note Free42 may calculate even powers differently (as a series of multiplications using repeated squaring), never employing more than 34 digits (see also p. B-22).

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 16×32 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 14×20 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 $< 0020_{16}$ are for control purposes; some of them ($4, 10_{10}, 27_{10}$) 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 3.5×10^9 people using either Greek or Latin alphabets:

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

Afrikaans, aymara, Bahasa Indonesia, Bahasa Melayu, Basa Jawa, Basa Sunda, bosanski, català, Cebuano, čeština, Cymraeg, dansk, Deutsch, eesti, Ελληνικά, English, español, euskara, Filipino, français, Gaeilge, galego, hrvatski, italiano, Kiswahili, kreyòl (ayisien), kurdî, lietuvių, magyar, Malagasy, Nāhuatl (Mexicatlatolli), nederlands, nihongo (rōmaji), norsk, O'zbek tili, polski, português, quechua (runasimi), română, shqip, slovenščina, slovensky, srpski, suomi, svenska,

Tagalog, tatarça, Türkçe, Türkmen dili, Vlaams, walon, and zhōngwén (hànyǔ pīnyīn).

This makes the *WP 43S* the most versatile multilingual calculator available worldwide.⁹⁷ If you know of further living languages covered (with ≥ 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 common realistic scale:

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

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. 168), 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:

⁹⁷ 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.

$\text{Re}(z)$	$\text{Im}(z)$	$r(z)$	$\varphi(z)$	
$-\infty$	$-\infty$	∞	-135°	$-\frac{3\pi}{4}$
0	$-\infty$	∞	-90°	$-\frac{\pi}{2}$
∞	$-\infty$	∞	-45°	$-\frac{\pi}{4}$
∞	0	∞	0°	0
∞	∞	∞	45°	$\frac{\pi}{4}$
0	∞	∞	90°	$\frac{\pi}{2}$
$-\infty$	∞	∞	135°	$\frac{3\pi}{4}$
$-\infty$	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 75 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^{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 ∞ 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.⁹⁸

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

2020-01-06 17:28 CL4R /max 64:2 A S L
-1.428 571 428 571 428 571 428 571 428 571 429 $\times 10^{-235}$
-1.428 571 428 571 429 $\times 10^{-235}$

⁹⁸ 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.

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

Complex numbers in Cartesian notation require $1 + 15 + 12 + 15 + 1 = 44$ px for $+jx$ 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 2×16 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:

$$\begin{aligned} & -6.234\ 5 \times 10^{-925} + i \times 6.234\ 5 \times 10^{-925}, \\ & -6.234\ 56 \cdot 10^{-925} + i \cdot 6.234\ 56 \cdot 10^{-925} \end{aligned}$$

With SHOW, any complex number can be displayed with 34 digits precision in two rows:

2020-01-06 17:15 CL₄R /max 64:2 A \bar{x} S
-8.403 361 344 537 815 126 050 420 168 067 229×10^{-126}
 $-i \times 5.882 352 941 176 470 588 235 294 117 647 059 \times 10^{-158}$

$$-8.403 \times 10^{-126} - i \times 5.882 \times 10^{-158} \quad -1. \times 10^{-33}$$

Complex numbers in **polar notation** need $4 + 16 + 4 = 24$ px for $\frac{4}{4}$ 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:

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

With SCI or ENG, the minimum number of decimals depends on the angular display mode since output is confined to the interval -180° to $+180^\circ$ or its equivalents, e.g. $-\pi$ to $+\pi$ in *radians* or -200° 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*:

$$-4.234\ 5 \times 10^{-925} \angle -120.234\ 5^\circ$$

For *radians* or *multiples of π* , however, 5 decimals are displayable always at least:

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

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 $(1 + 15 + 21 \times 16 + 6 \times 8 = 400 \text{ px})$. 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 $\times 10^{21}$

With SHOW, one *long integer* may take up to 7 rows meaning up to 296 digits (see e.g. the prime number $2.8\dots\times 10^{295}$ shown at right displaying all its digits). Only the most significant 294 digits will be displayed of an integer $\geq 10^{297}$, with ellipses added at its end.

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 212₃

In base 4 (with narrow blanks every two digits), 19 digits representing 38 bits are displayable:

3 21 23 30 22 11 21 20 32 12₄

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 0000₂ .

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 0000₂.

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

For showing the maximum of 64 bits in base 2, two special 5 px wide characters were created:

111. 1100 .01. 11.1 .11. 111. 1100 .01. 11.1 .11. 111. 1100 .01. 11.1 .11. 11.₂ .

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

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

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

$$\begin{bmatrix} -6,609 \cdot 10^{-19} & -6,609 \cdot 10^{-19} & -1,609 \cdot 10^{-19} & -1,609 \cdot 10^{-19} \end{bmatrix}$$

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

$$\begin{bmatrix} -6.609 \cdot 2 \cdot 10^{-19} & -6.609 \cdot 2 \cdot 10^{-19} & -1.609 \cdot 2 \cdot 10^{-19} & -1.609 \cdot 2 \cdot 10^{-19} \end{bmatrix}$$

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

$$\begin{bmatrix} \dots & -6,609 \cdot 2 \cdot 10^{-19} & -6,609 \cdot 2 \cdot 10^{-19} & -1,609 \cdot 2 \cdot 10^{-19} & \dots \end{bmatrix}.$$

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 5×4 real matrix can be displayed entirely always, using SCI 3 in worst case.

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

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

$$\begin{bmatrix} \dots & -6.609 \cdot 226+i \cdot 6.609 \cdot 226 & -1.609 \cdot 226+i \cdot 1.609 \cdot 226 & \dots \end{bmatrix}$$

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

$$\begin{bmatrix} \dots & -6.60E^{-199}+i \cdot 6.60E^{-199} & -1.60E^{-199}+i \cdot 1.60E^{-199} & \dots \end{bmatrix}.$$

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

$$\begin{bmatrix} \dots & -6,609+i \cdot 6,609 & -6,609+i \cdot 1,609 & -1,609+i \cdot 1,609 & \dots \end{bmatrix}.$$

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

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

2017-05-08 23:49 RLE π /3 546f 64:u^o A $\sqrt{3}$ 0.4500

containing 45 characters.

Putting the alphabet in a row allows for

abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOP

i.e. 41 characters.

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

Command and variable *names* in menus 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 U₂, 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 *text string* in any *stack register*, its base line should be positioned where the base line of the respective numeric output would have been.
- With a *text 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 *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. 71 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. Furthermore, 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-30), 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 behind each operator like +, -, *, ×, /, and ! (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 (), then stepping back into the parentheses for specifying the argument. The same applies to dyadic (like C_{xy}) 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^{23} to xy^{23} . 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 this and seeing a forthcoming better display (than the very limited one of the *HP-20b/30b*) at the horizon allowing for softkeys, 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 hypothetical portrait *RPN*

pocket calculator in November 2012 (see <https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234505>). The concept and first layout for the *WP 43S* on the same site in return (<https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234685>). The picture here 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, development of the *DM42* 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 for our user interface is a *menu* encompassing three *views* containing up to 54 functions in total: the top *view*, one *view* going up via and one going down via . Larger *menus* lack efficiency, smaller *menus* lack functionality. Besides its visibility, a function 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. pp. B-1f).

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. 139f). So we can conclude that the real number range supported (cf. p. 163) 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 *DTs* 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 p. B-22).

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 experimental 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 (240×240 px for data, 242×240 px incl. the vertical axis). Hence softkeys can be positioned on one side of the diagram (max. 3×2 labels) 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. Although axis and diagonal can be created using AGRAPH and PIXEL, four characters are provided in the small font for 'drawing' the vertical axis and the 45° line:



Data points can then be plotted using POINT (containing 3×3 px); positioning them properly in the diagram, however, will require some background calculations best performed by a program. For POINT, some of the graphic control modes of HP-42S shall be implemented in analogy (the table below is copied from the HP-42S OM, p. 137):

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

* Default setting.

Settings 1 (0, 0) and 3 (1, 0) herein should suffice for our plotting (cf. GRAMOD on p. 101).

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 pixel 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' left 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 3×10^{-35} . During calculations, such errors accumulate (cf. footnote 69).

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° , you can nevertheless get 28-digits precision in the result since the input of 9° 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 (cf. footnote 69). Consequently, *WP 43S* also works with 39 digits internally most times and rounds results to 34 digits. *SLVQ* calculates using 72 digits. The statistical summation registers are 75 digits wide (used also for the initial steps of variance calculation). Range reductions for trigonometric functions use many more digits (cf. B-4).

Luckily, real-world problems are usually far less precisely defined than the internal precision of *WP 43S*. Compare also the set of physical and astronomical constants provided (cf. pp. 130ff).

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 placing the corresponding golden and blue labels.

Sorting in Detail

There is no international standard for sorting characters; we had to invent our own order. Sorting of *items*, variable *names*, *text strings*, *system flags*, etc. on *WP 43S* works as listed below, top down and left to right.

Note that sorting is a two-step procedure: step 1 sorts the *text 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.⁹⁹ The 4-digit number trailing each character in the table is its hexadecimal *Unicode*.¹⁰⁰

↳ 0020	2003	2004	...	2008	200a	↳ 2423
❶ 0030	❷ 220e	❸ 00b0	❹ 2070	❺ 2080		
❻ 0031	❼ 2027	❽ 00bc	❾ 00bd	❻ 2071	❼ 2081	❼ 2460
➋ 0032	➋ 00b2	➋ 2082	➋ 2461			
⌃ 0033	⌃ 00b3	⌃ 2083	⌃ 2462	⌃ 221b		
⌄ 0034	⌄ 2074	⌄ 2084	⌄ 2463			
⌅ 0035	⌅ 2075	⌅ 2085	⌅ 2464			
⌆ 0036	⌆ 2076	⌆ 2086	⌆ 2465			
⌇ 0037	⌇ 2077	⌇ 2087	⌇ 2466			
⌈ 0038	⌈ 2078	⌈ 2088	⌈ 2467			
⌉ 0039	⌉ 2079	⌉ 2089	⌉ 2468			
⌊ 2491	⌊ 2469					
⌋ 246a						
⌌ 246b						
⌍ 246c						
⌎ 246d						
⌏ 246e						
⌐ 246f						

⁹⁹ Applying this algorithm, a section of CATALOG'FCNS looks like e.g. **s**, **SAVE**, **SB**, **SCI**, **SCI0VR**, **scw→kg**, ..., **SLVQ**, **s_m**, **s_{mw}**, **SOLVE**, ...

Sorting is illustrated for the small font here. It holds also for the large font as far as characters are applicable.

¹⁰⁰ Characters printed on grey background are inaccessible for users; those printed on darker grey are not used at all so far.

A 0041	a 0061	ä 00aa	À 24b6	á 2090	à 249c	
	À 00c0	à 00e0	Á 00c1	á 00e1	À 00c2	â 00e2
	Ã 00c3	ã 00e3	Ä 00c4	ä 00e4	Ä 00c5	ã 00e5
	Æ 00c6	æ 00e6	Ā 0100	ā 0101	Ā 0102	ă 0103
					À 0104	ą 0105
B 0042	b 0062	ß 24b7	þ 249d			
C 0043	c 0063	ç 24b8	ç 249e	ç 00c7	ç 00e7	
	Ć 0106	ć 0107	Ć 010c	ć 010d	Ć 2102	ć 2201
D 0044	d 0064	đ 24b9	đ 249f	đ 00d0	đ 00f0	
			Đ 010e	đ 010f	Đ 0110	đ 0111
E 0045	e 0065	€ 24ba	€ 2091	€ 24a0	È 00c8	è 00e8
	É 00c9	é 00e9	É 00ca	é 00ea	É 00cb	ë 00eb
	Ē 0112	ē 0113	Ē 0114	ē 0115	Ē 0116	è 0117
	Ę 0118	ę 0119	Ę 011a	ę 011b	Ę 2073	
F 0046	f 0066	ƒ 24a1	ƒ 24bb			
G 0047	g 0067	¤ 24a2	¤ 24bc	¤ 011e	¤ 011f	
H 0048	h 0068	ḧ 210e	ḧ 24a3	ḧ 24bd	ḧ 2095	
					ḧ 0127	ḧ 210f
I 0049	i 0069	í 24be	í 24a4	í 00cc	í 00ec	
	Í 00cd	í 00ed	Í 00ce	í 00ee	í 00cf	í 00ef
	Ĭ 012a	í 012b	Ĭ 012c	í 012d	Ĭ 012e	í 012f
					í 0130	í 0131
J 004a	j 006a	ј 24bf	ј 24a5			
K 004b	k 006b	ќ 24c0	ќ 24a6	ќ 2096		
L 004c	l 006c	љ 24c1	љ 24a7	љ 2097	љ 0139	љ 013a
			љ 013d	љ 013e	љ 0141	љ 0142
M 004d	m 006d	ӎ 24c2	ӎ 24a8	ӎ 2098		
N 004e	n 006e	ڽ 24c3	ڽ 24a9	ڽ 2099	ڽ 00d1	ڽ 00f1
	ň 0143	ń 0144	ň 0147	ń 0148	ň 2115	

O 004f	o 006f	ø 00ba	ø c 00a9	ø 24c4	ø 24aa	ø 2092
	ò 00d2	ò 00f2	ó 00d3	ó 00f3	ò 00d4	ó 00f4
	ö 00d5	ö 00f5	ö 00d6	ö 00f6	ö 00d8	ö 00f8
	ö 014c	ö 014d	ö 014e	ö 014f	œ 0152	œ 0153
	p 0070	p 24c5	p 24ab	p 209a		
Q 0051	q 0071	q 24c6	q 24ac	Q 211a		
R 0052	r 0072	r 24ad	r 24c7	Ŕ 0154	ŕ 0155	
				Ŕ 0158	ŕ 0159	Ŕ 211d
S 0053	s 0073	s 24c8	s 24ae	s 209b	Ś 015a	ś 015b
	ſ 015e	ſ 015f	ſ 0160	ſ 0161	þ 00df	
T 0054	t 0074	t 24af	t 22a4	t 24c9	t 209c	
			ṭ 0162	ṭ 0163	ṭ 0164	ṭ 0165
U 0055	u 0075	u 24ca	u 24b0	u 1d64	Ù 00d9	ù 00f9
	ú 00da	ú 00fa	ú 00db	ú 00fb	Ü 00dc	ü 00fc
	ő 0168	ő 0169	ő 016a	ő 016b	ő 016c	ő 016d
			ő 016e	ő 016f	ő 0172	ő 0173
v 0056	v 0076	v 24cb	v 24b1			
w 0057	w 0077	w 24cc	w 24b2	ŵ 0174	ŵ 0175	
x 0058	x 0078	x 1d61	x 24cd	x 24b3	x 2093	
			᷇ 0379	᷈ 0378	᷉ 037f	᷊ 221c
y 0059	y 0079	y 24ce	y 24b4	ÿ 00dd	ÿ 00fd	
	ÿ 0176	ÿ 0177	ÿ 0178	ÿ 00ff	ÿ 0233	ÿ 0232
z 005a	z 007a	z 24cf	z 24b5	ž 0179	ž 017a	ž 017b
			ž 017c	ž 017d	ž 017e	ž 2124
A 0391	α 03b1	α 2065	á 03ac			
B 0392	β 03b2					
Γ 0393	γ 03b3					
Δ 0394	δ 03b4	δ 2066				

E 0395	ε 03b5	έ 03ad		
Z 0396	ζ 03b6			
H 0397	η 03b7	ή 03ae		
Θ 0398	θ 03b8			
I 0399	ι 03b9	ί 03af	ϊ 03aa	
		Ϊ 03ca	Ϊ 0390	
K 039a	κ 03ba			
Λ 039b	λ 03bb			
M 039c	μ 03bc	μ 00b5	μ 2067	
N 039d	ν 03bd			
Ξ 039e	ξ 03be			
Ο 039f	ο 03bf	ό 03cc		
Π 03a0	π 220f	π 03c0		
Ρ 03a1	ρ 03c1			
Σ 03a3	σ 03c3	ς 03c2		
Τ 03a4	τ 03c4			
Υ 03a5	υ 03c5	ύ 03cd	Ύ 03ab	
		ϋ 03cb	Ϋ 03b0	
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	23a4	23a5] 23a6
{ 007b	} 007d			
↳ 2430	⤵ 2431	⤶ 2432	⤷ 2433	
⊕ 002b	⊕ 207a	⊕ 208a	⊕ 00b1	
- 002d	- 207b	-1 2072	- 208b	⊖ 2213
× 00d7	· 00b7	• 2219	◦ 2218	* 002a
			*	208f

/ 002f	\ 005c	
^ 005e		
, 002c	، 2429	
. 002e	. 2428	... 2026
! 0021	¡ 00a1	
? 003f	¿ 00bf	
: 003a	: 2236	÷ 00f7
؛ 003b		
' 0027	' 2018	' 2019
" 0022	" 201c	" 201d
	,	" 201e
		" 201f
		« 00ab
		» 00bb
@ 0040		
- 005f	▬ 2427	
~ 007e		
→ 2192	↗ 21c0	
← 2190	↖ 21cd	
↑ 2191	↗ 21e7	▲ 21c9
↓ 2193	↘ 21e9	▼ 21cb
➤ 21c4		
↕ 2195		
☰ 21cc		
¬ 00ac		
׀ 2227	׀ 2228	׀ 22bb
& 0026		׀ 22bc
007c	2223	2224
		2225
		2226

¹⁰¹ Look up https://de.wikipedia.org/wiki/Anführungszeichen#Andere_Sprachen or https://en.wikipedia.org/wiki/Quotation_mark#Summary_table for properly using these characters in different languages.

« 226a	< 003c	≤ 2264	≡ 2261	:= 2254	= 003d	≈ 2243
	≈ 2248	≥ 2258	≠ 2259	≠ 2260	≥ 2265	> 003e
						» 226b
% 0025	\$ 0024	€ 20ac	¢ 00a2	£ 00a3	¥ 00a5	₪ 00a7
⌚ 221a	⌚ 221d					
♾ 221e	♾ 209e	♾ 209f				
⅀ 222b	⅀ 222c	⅀ 222d	₱ 222e	₱ 222f	₱ 2230	
₵ 2299	₵ 229a	₵ 2068				
⊕ 2295	⊕ 2069					
ܲ 221f	ܲ 22a5					
ܳ 2220	ܳ 2221	ܳ 2222				
ܴ 2308	ܴ 2309					
ܵ 230a	ܵ 230b					
ܶ 2399	ܶ 231b	ܶ 231a	ܶ 242a	ܶ 242c	ܶ 242f	ܶ 2434
# 0023						
ܷ 242d	ܷ 242e					
ܸ 2200	ܸ 2202	ܸ 2203	ܸ 2204	ܸ 2205	ܸ 2206	ܸ 2207
	ܸ 2208	ܸ 2209	ܸ 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. 42.

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 state as it was before executing last command. Until 2020, we assumed that such a complete UNDO was viable in *WP 43S* as well, but the overhead turned out forbidding.

Since

- any user *flags* altered inadvertently can be reverted easily and
- any wide-reaching clear commands (CLREGS, CLFALL, CLPALL, etc.) ask for confirmation before executing,

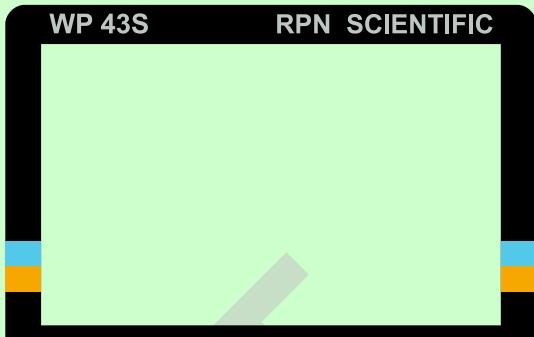
we implemented UNDO recalling not only the *stack* but also the summation *registers* and the *system flags* as they were before executing last command, for better user experience (this also allows for undoing $\Sigma-$).

For Your Convenience

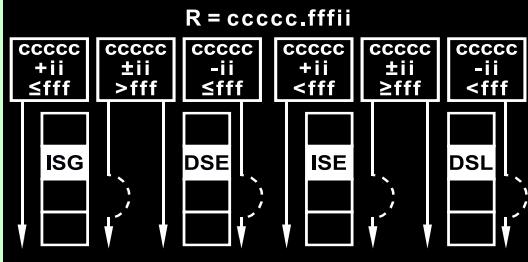
Please find here the display frame of your *WP 43S* as we designed it (printed to scale), and below its virtual keyboard in alpha input mode (*A/M*) plus a branching helper.

Overleaf, its original keys and keyplate are printed to scale. You will also find there an explanatory picture taken from the back of an *HP-16C* (with C denoting CARRY and G OVERFL).

Choose your favorites, cut them out, and use them with your *WP 43S*. If you bought it complete and flashed, keys and keyplate shall be printed properly on its top face and the picture shown here at right is on its rear for your reference.



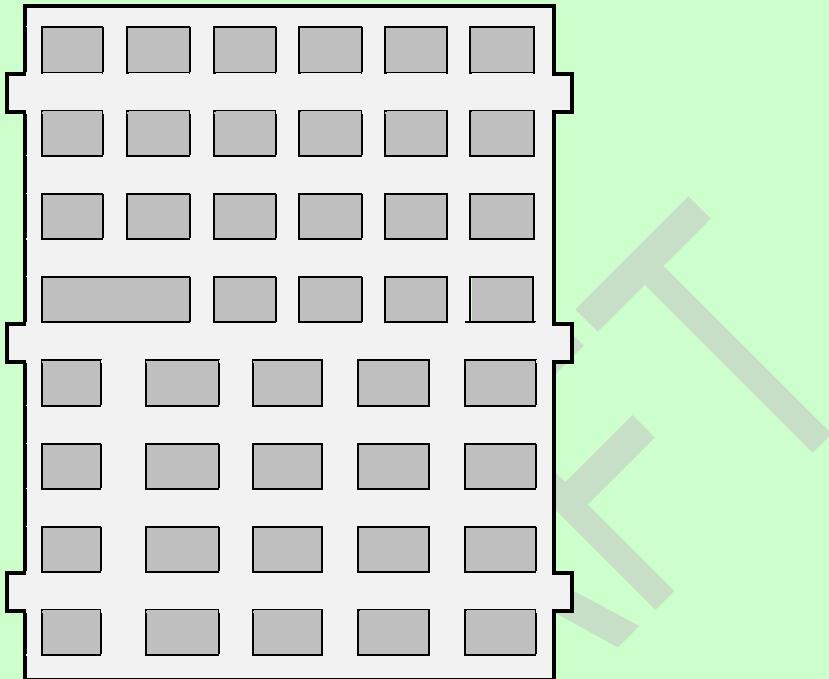
www.wp43s.com





	C	G	
[+/-]	x	x	
[X]	--	x	
[÷]	x	x	RMD≠0+C
[√x]	x	--	RMD≠0+C
CHS ABS	--	x	
DBL X	--	o	y-x+(X&Y)
DBL ÷	x	o	(Y&Z)÷x+x:X; RMD≠0+C
SL	x	--	o → [] → o
SR	x	--	o → [] → o
ASR	x	--	[] → o
RL	x	--	o → [] → o
RR	x	--	[] → o
RLC	x	--	[] → o
RRC	x	--	[] → o

Sample for a quick overlay, almost to scale:



1. Print out on standard copy paper (80 g/m^2). Fill in labels, if and as required.
2. Laminate with 80 mics stock (result is a thickness of about 0.25 mm).
3. Cut out voids for keys and perimeter. Attention: Tabs as shown are too wide and too long, cut smaller.