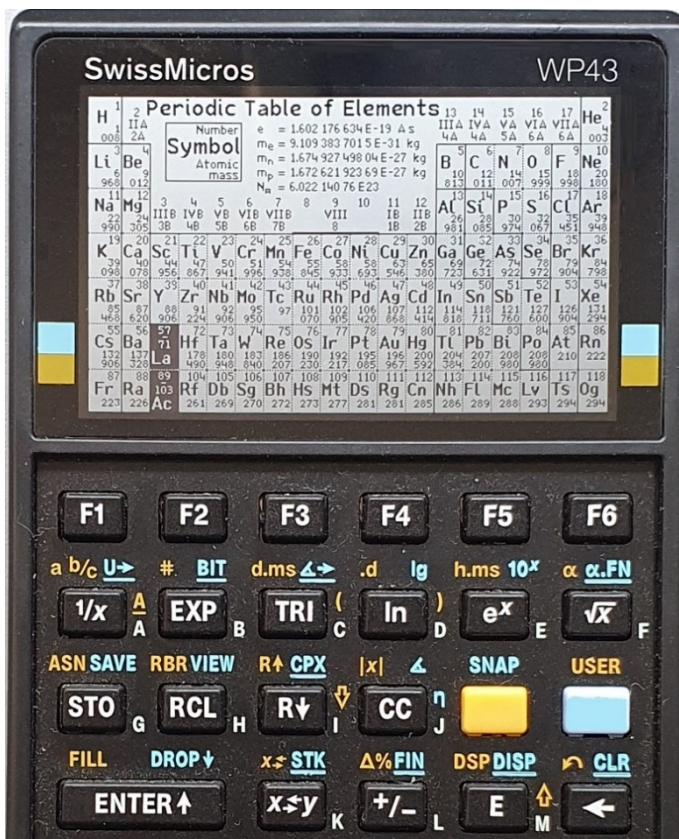




WP43 REFERENCE MANUAL

This manual supplements the *WP43 Owner's Manual and Problem Solving Guide*. (*WP43* is a free scientific software for *DM42* calculator hardware built by *SwissMicros*. You can redistribute that software 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. *WP43* 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 <http://www.gnu.org/licenses/> for more details.)

This manual is preliminary still; it will change while we develop *WP43* in course of this project. We reserve the right to do so at any time. The fundamental principles of *WP43* will remain constant, however. You can stay informed by watching <https://gitlab.com/rpncalculators/wp43>.

Copyright © 2015 - 2026 Walter Bonin, Radebeul, Germany

All rights reserved. The materials provided in this manual are freely accessible for personal self-study. For any commercial or corporate use, no part of this manual may be copied, displayed, extracted, reproduced, utilized, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical or otherwise including but not limited to photocopying, recording, or scanning without the prior written consent of the author. The locations highlighted cyan are open issues – information is missing there or needs further discussion or investigation to be determined. Any contributions in these matters are appreciated.

HP is a registered trade mark of *Hewlett-Packard*. *Excel* and *WordPad* are applications of *Microsoft*.

The drawing on p. 187 was kindly supplied by *SwissMicros*. The plots in *Appendix I* (unless specified otherwise) and the pictures on pp. 128, 180 (bottom left), and 333 are based on material found in *Wikipedia*. The photos processed for the pictures on p. 259 were found in the *MoHPC* (see p. 13). Other sources are stated locally. The pictures, diagrams, and graphics beyond were created by the author.

Over 100 internet addresses and links are specified as found and verified on 2025-05-11. 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 in old Europe) – using it we can reach the maximum number of readers without further translations. This is the only reason. I apologize to the people of other languages and inserted some 'translator's notes' where applicable.

And if you find some barbed remarks in the text, they just reflect the author's impressions watching this world for seven decades so far. No personal offense intended but over-confident people might get some.

Printed in the *USA* since I got the *ISBN* there most easily.

ISBN-13: 978-172950106-1

ISBN-10: 172950106-0

WP43 would neither have been created without our love for *Classics*, *Woodstocks*, *Stings*, *Spices*, *Nuts*, *Voyagers*, and *Pioneers* nor without our ability to imagine better products on a way towards a better world. Thus we want to quote what was printed in HP 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*

Just in Case You Do Not Have Your Own WP43 Calculator yet ...

A pilot batch of WP43 calculators was made on SwissMicros' production line in China in the summer of 2022. These were sold out in few weeks – thus, please turn to <https://www.swissmicros.com/> expressing your wishes.

Alternatively, you can create a WP43 most easily by flashing a DM42 you own – this way will cost you next to nothing but an overlay will become inevitable. Please consult *Appendix F of this manual* for what and how to do.

The WP43 software package includes a free full size *Simulator* (→ *Appendix E*), so you can check it out on your computer before buying any hardware. And later on, you may test and edit your programs there as well before sending them to your WP43 calculator. Function sets are 99% identical on both platforms.

For the following, we assume you hold your own WP43 in your hands – or have access to its *Simulator* at least.

TABLE OF CONTENTS

Welcome!	11
Print Conventions and Common Abbreviations	12
Section 1: The Index of Items (IOI)	14
Commands, Menus, and Further Item Names Reserved	15
	0 – 9 20
	A 20
	B 24
	C 26
	D 33
	E 37
	F 40
	G 43
	H 46
	I 47
	J 50
	K 51
	L 53
	M 57
	N 66
	O 68
	P 68
	Q 72
	R 72
	S 81
	T 89
	U 92
	V 93
	W 94
	X 96

Y	99
Z	99
A, α	100
β	101
Γ , γ	101
Δ , δ	102
ε	102
ζ	102
λ	103
μ	103
Π , π	103
Σ , σ	103
Φ	105
χ	105
ψ	106
ω	106
(, [, +, -, $\times$, /, ^	106
?	107
$\rightarrow$,	109
%	112
$\sqrt{}$, $\int$, $\approx$, $\equiv$, #	113
Constants	116
System Flags	129
Tests	133
Nonprogrammable Commands and Keys	134

Section 2: Menus and Catalogs 135

One to Find and Rule Them All – the CATALOG	135
Rapid Access to Cataloged Items	138
Further Menus and Their Contents	140
Unit Conversions	152

Section 3: Calling and Executing Commands	165
Using XEQ for Executing Operations	165
Command Parameter Input and Closing It	166
Alphanumeric Input in X and Closing It	168
Operations Requiring Trailing Parameters	172
Operations Allowing For Indirect Addressing	174
Operations Changing Data Types	174
Operations in Equations	177
 Appendix A: Hardware	 179
Off-Images	186
Overlays	186
 Appendix B: Memory Management	 189
Data Types	189
What Will Happen when Changing Word Size?	192
SAVE and LOAD...	192
Range and Representation of Real Numbers	193
Limitations	197
Special Results	199
Program Step Sizes	204
Program Files	206
 Appendix C: Messages and Error Codes	 208
 Appendix D: Comparison to the Function Sets of <i>HP-42S</i>, <i>HP-16C</i>, <i>HP-21S</i>, and <i>WP 34S</i>	 215
Corresponding Operations on <i>HP-42S</i>	215
Corresponding Operations on <i>HP-16C</i>	221
Corresponding Operations on <i>HP-21S</i>	223

Corresponding Operations on <i>WP 34S</i>	225
New Commands on your <i>WP43</i>	230
Reference Literature	238

Appendix E: Emulating a *WP43* on Your *PC* **240**

How to Exchange Programs Between Your <i>WP43</i> and Your <i>PC</i>	242
--	-----

Appendix F: Flashing & Updating Your *WP43* **245**

How to Create Your <i>WP43</i> by Flashing a <i>DM42</i>	245
How to Update Your <i>WP43</i>	247

Appendix G: Troubleshooting Guide **249**

First a Few Words About Coin Cells	249
Calculator Freeze	251
Exception	252
Fresh Quality Battery Constantly Low	252
Individual Keys not Reacting Properly	253
Keymap Trouble	254
Other Trouble	254

Appendix H: Historic Remarks and Caveats for Younger Users (What Grandpa Still Knew) **255**

Keys and Keyboards	256
Reverse Polish Notation	257
Why Does ENTER↑ Work the Way It Does? And Why Are there Automatic Stack Lifts and Drops?	259
How Many Stack Levels Are Necessary?	261
How Many Stack Levels Shall Be Displayed?	264
Precision and Accuracy of Calculations Using Finite Real Numbers	265
About the Classical Mach Number Formula	268

Appendix I: Advanced Mathematical Functions and Tasks 271

Number Generating Functions	271
Statistical Distribution Functions (PMF, PDF, CDF, etc.)	273
More Statistical Formulas, also for Curve Fitting	281
Curve Fitting Models Provided	290
The Pedestrian's Measurement System Analysis (MSA)	299
Error Propagation in Calculations	302
Solving Differential Equations (1)	303
Solving Differential Equations (2)	308
Areas, Curve Lengths, and Volumes	316
Orthogonal Polynomials	321
Even More Mathematical Functions	326

Appendix J: Information for Advanced Users 335

Recursive Programming	335
A Few Words About Factorizing	335
A Few Words About the Solver	336
More Than a Few Words About the Integrator	337
Differentiating Real Functions Using Complex Numbers	342
A Few Words About Random Numbers	343
A Few Words About Uncertainty, Significance, and Confidence	344
A Few Words About Curve Fitting	344
A Few Words About TVM	345
Translation of WP43 Characters Deviating from UTF-8	346
Where to Find Programs Readily Available?	348
Where to Find More about WP43, its Ancestors and Offsprings?	348

Appendix K: The Most Honorable Members of the Pilot Run 350

Appendix L: Making a WP43 Stand for Less Than \$2 Cost 352

Appendix M: Release Notes	353
Appendix N: Index of Names, Calculators, Companies, etc.	362

WP43 Quick Reference Guide	Q-1
ADDRESS SPACE	1
DATA TYPES	2
USING MENUS	3
MODES	4
DISPLAY FORMATS	4
PROGRAMMING	5
EXECUTING FUNCTIONS AND PROGRAMS	6
CLEARING AND DELETING	7
MATRIX OPERATIONS	8
PROBABILITY	10
STATISTICS	11
ADVANCED OPERATIONS	12
OPERATIONS ON SHORT INTEGERS	14
OPERATIONS ON TEXT STRINGS	15
USER WRITEABLE SYSTEM FLAGS	16

Background Facts and Considerations	B-1
Accessing Items	1
Angles	7
Calculation Internals	10
Character Sets	11
Complex Notation, Storage, and Calculations Close to Infinity	17
Display Segmentation	18

Echo and Fallback	19
Equations	20
Execution Times	21
Functionality of ENTER↑	22
Keystrokes	22
MSA	22
Names of User Defined Items	22
Number Range	23
Precision and Accuracy	23
Prefixes	24
Random Numbers	25
RPN vs. RPL or Vice Versa	25
Sorting in Detail	25
Stack Lift Disabling Functions	31
Stack Size	31
Structured Programming	32
UNDO	32
What Is Open Still	33

For Your Convenience

427

WELCOME!

This reference volume of the *WP43* documentation supplements your *WP43 Owner's Manual* with comprehensive information about each and every *item* (i.e. command, *menu*, *catalog*, browser, application, conversion, *variable*, constant, *system flag*, *global label*, and *character*) provided in your *WP43*. With these two manuals, you have got all the user information about *WP43* in your hands in merely two files, both conveniently readable on your smartphone.¹

Section 1 contains mainly the *Index of all Items (IOI)*, taking more than a quarter of this manual.

Section 2 presents the structures and contents of all the *menus* and *catalogs* provided.

Section 3 shows further access methods to operations and lists all functions requiring parameters, allowing for indirect addressing, changing *data types*, and being accessible in equations.

Appendices A – N cover special topics like hardware, error messages, the *WP43 Simulator*, and *WP43* firmware updates. And there are lots of information more – please check the [Table of Contents](#) for details.

A **Quick Reference Guide** (16 pages) is provided close to the end of this manual, followed by background information about this project, and some helpful prints for your convenience at the very end of this volume.

And here is the inevitable

WARNING: Read the **WARNINGS** in your *WP43 Owner's Manual*, pp. 16f – else you are only allowed to look at your *WP43*. And there is one important **WARNING** in *Appendix H*, concerning vintage calculators.

Enjoy!

Walter

¹ This manual might apply to *WP43* offsprings like *C43*, *C47*, *R47*, etc. to some extent. It can't replace dedicated manuals for those, however, and was never intended to do so.

Print Conventions and Common Abbreviations

- Standard text font in our manuals is *Arial*. Emphasis is added by underlining or **bold** printing. COMMANDS, MENUS, predefined VARIABLES and SYSTEM FLAGS (SET or CLEAR) are generally called by their *names*, printed capitalized in running text.

Examples are printed in green, quoted text in blue (as well as translator's notes). Bold italic letters (like ***n***) denote *variables*, bold regular *characters* (like **b**) constant sample values (e.g., specific labels, numbers, or symbols).

Specific *terms*, *titles*, *trademarks*, *names*, or *abbreviations* are printed in italics, hyperlinks in blue underlined italics; such links will point to page numbers (linked per default), the Table of Contents, or an external address fully specified.

- *Times New Roman* regular letters are used for unit symbols and mathematical functions, italics for *unit names* in running text.

Bold capitals denote **REGISTERS**, lower case bold italics their *contents*.² E.g., *register Y* contains the value *y*, and **R45** contains *r45*. *Stack* contents are generally quoted in order [*x*, *y*, *z*, ...].

- This **(KEY)** font (created by *Luiz Vieira* of Brazil) denotes calculator **(KEYS)**, incl. **(softkeys)** in general. For shifted operations like **(GTO)** or **(SF)**, the respective colors are used. **Text** and numeric outputs (like 1.234×10^{-56} or $7,089 \cdot 10^{12}$) are printed as seen on your calculator screen.
- *Courier* is used for `filenames` and describing `numeric formats`.
- Regarding mathematical notation, we follow *ISO 80000-2* to comply with international standards. For the manuals, we chose decimal points, multiplication crosses, and division slashes – you may choose decimal commas, multiplication dots, and division colons on your *WP43* as well, of course.

All this holds unless stated otherwise locally.

The following abbreviations are used throughout this manual. They are listed here for your reference (a few more may be used and explained locally):

OM = Owner's Manual (typically referring to the *WP43 OM*);
OMS *n* refers to its *Section n*, **OMA *n*** to its *Appendix n*.

² We kept the term *register* for the space where a particular object is stored, although the actual size of such a *register* varies widely according to the size of the object stored therein.

2LS - 8LS	= 2- through 8-level stack
1D, 2D, 3D	= one-dimensional, two-dimensional, three-dimensional.
ADM	= angular display mode (→ OMS 3).
AIM	= alpha input mode (→ OMS 3).
App.	= Appendix.
ASL	= automatic stack lift (→ OMS 1).
CDF	= cumulated distribution function (→ OMS 2).
DT	= data type (→ App. B).
FD	= the USB FAT disk (→ App. A).
FM	= flash memory (incl. the FD, → App. A and OMS 4).
f.	= footnote.
GP	= general purpose (→ OMS 1).
HP	= Hewlett-Packard.
IOI	= Index of Items (→ 20ff).
L.R.	= linear regression (covering OLS and TLS, → App. I).
MoHPC	= Museum of HP Calculators at https://www.hpmuseum.org .
MSA	= measuring system analysis (→ App. I).
OH	= Owner's Handbook.
OLS	= ordinary least squares regression (compare TLS = total ~).
PC	= computer of arbitrary kind and brand.
PCB	= printed circuit board.
PDF, PMF	= probability density (or mass) function (→ OMS 2).
PEM	= program-entry mode (→ OMS 4 and RUM).
PET	= Programming Examples and Techniques.
PP	= program pointer (→ OMS 4).
px	= pixels.
RAM	= random access memory, allowing read and write operations.
RPN	= reverse Polish notation (→ OMS 1).
RoV	= register or variable.
RUM	= run mode (→ OMS 1 and PEM).
SD	= standard deviation (→ App. I).
SDC	= startup default configuration.
SM	= SwissMicros.
SRS	= subroutine return stack (→ App. B).
TN	= translator's note (TNG = TN for readers speaking German).
TVM	= Time Value of Money – a preprogrammed financial application featured by HP business calculators since 1972 (→ OMS 6).

SECTION 1: THE INDEX OF ITEMS (IOI)

All the *items* provided on your WP43 (as defined in [OMS 1](#)) are explained in this *Section*. Almost all of them are listed in the table starting on p. 20 with their *names* (as they are displayed in programs) and the way to call them. Most of them are picked from *menus*; then we recorded the respective *menu*, its *view*, and the *label* shown therein (printed on gold or blue if  or  is required for accessing it); we are confident you will find the corresponding *softkey*.  abbreviates .

After this table, the *constants* and *system flags* provided are explained in separate chapters comprehensively. *Menus* are covered in *Section 2*.

Each *item* on your WP43 is identified by its unique *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). Such *labels* are not required to be unique; so one particular *label* may point to or call various *items*, depending on its context.

Since there is no agreed standard for character sorting, we had to invent our own: sorting (*item names*, for instance) works in the following order on your WP43:

0...9 Aa...Zz Aα...Ωω () [] { } + ± - × / ^
, . ! ? : ; ' " « » . @ _ ~ → ← ↑ ↓ ↗
¬ ∧ ∨ & | « < ≤ ≡ := = ≈ ≡ ≐ ≠ ≥ > »
% \$ € £ ¥ √ ∞ ∫ ⊙ ⊕ ⊗ ⊘ # UK US

Accented letters follow their parents, as do superscripts and subscripts.³ The same applies to *Euler's* *e*, *i*, and the *ℓ* for *liter*.

There are a few *labels* printed on the keyboard of your WP43 which will execute differently depending on the context you access them:

³ The two *characters* printed on grey here (used in unit conversions) are only indirectly accessible for the user. Find the sorting table for all the *characters* created in [Background Facts and Considerations](#) near the end of this manual.

Label	Context, condition	Functionality
a b/c	Fraction display off	Displays x as a <i>proper fraction</i> (→ OMS 2)
	Fraction display on	Toggles <i>proper</i> and <i>improper fraction</i> display
#	Input open	Follows a <i>short integer</i> , preceding its base
	Closed x	Followed by a number calls →INT base followed by D calls →INT 10 followed by F calls FP followed by H calls →INT 16 followed by I calls IP
d.ms	Input open	Closes input of an <i>angle</i> in <i>sexagesimal degrees</i>
	Closed x	Calls D.MS
.d	Input open	Closes input of a <i>date</i>
	Closed x	Calls →REAL
h.ms	Input open	Closes input of a <i>sexagesimal time</i>
	Closed x	Calls →H.MS
P/R	In <i>RUM</i>	Opens <i>PEM</i> (→ OMS 4)
	In <i>PEM</i>	Closes <i>PEM</i>
CC , ENTER ↑, ←, ▲, ▼, R/S , EXIT	Various	→ OMA 1 for comprehensive explanation of the functionalities of these seven context sensitive keys

Commands, Menus, and Further Item Names Reserved

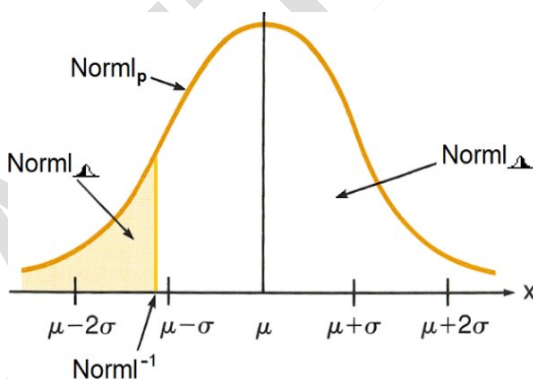
In principle, *WP43* operations work like the corresponding ones on the [WP 34S](#) wherever applicable (→ [App. D](#)). Referring to vintage *HP* calculators, most functions work as they did on the [HP-42S](#), *bit* and *integer* functions as on the [HP-16C](#), unless specified otherwise.

In the first column of the following *IOI* table,

- operations allowed in equations are printed on yellow,
- those whose results are reflected in the *status bar* on grey,
- those not allowed in programming (*PEM*) on violet background.
- Commands asking you for your confirmation are printed red.
- You can UNDO every operation except those printed red or orange.

Some 390 operations listed in the *IOI* are new compared to *HP's RPN* pocket calculators.⁴ For *items* known from prior such calculators but deviating functionally on your *WP43*, their remarks are printed on light red.

Functions operating on the accumulated statistical data are highlighted light blue. For any probability distribution XY , XY_p tags its *PDF*, XY_{Δ} its *CDF*, XY_{Δ} its error probability, and its quantile function is called XY^{-1} (as pictured here for **Norml**). Commands working on the calculator exclusively (i.e. not on the *Simulator*) have their remarks printed on gold.



For the vast majority of operations, their respective remarks start with the number of keystrokes needed to call them (followed by a + for each trailing parameter, all enclosed in brackets []). In most cases then follows a number representing ...

- (0) functions without any effects on the *stack* (like mode setting);
- (1) *monadic*,
- (2) *dyadic*, and
- (3) *triadic functions* as defined in *OMS 1*; and
- (-1) or (-2) functions pushing one or two objects on the *stack*.

⁴ → [App. D](#). We neither compared *HP's RPL* calculators nor the *HP Prime* since they definitely exceed the realm of shirt pocket calculators.

Note that some functions overwrite two *stack* levels: e.g., $\rightarrow\text{POL}$ and $\rightarrow\text{REC}$, as you might have expected.⁵

Operation *parameters* will be taken from the lowest *stack register(s)* unless mentioned explicitly in column 2 of the *IOI* – then they will trail the command (as mentioned above). Most parameters of **probability distributions** shall be given in *registers I, J, and K* as specified.

We show below three **examples** of the parameter notation used in the *IOI*. Let's assume **R12** contains **15.67** here, i.e. $r12 = 15.67$ generally.

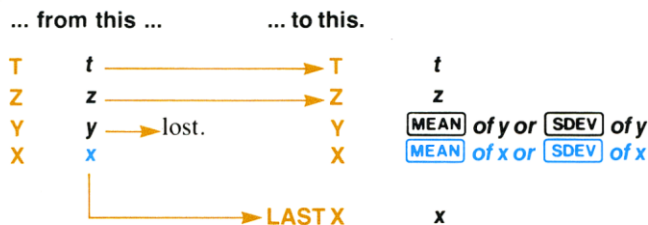
1. ***n*** represents an arbitrary *integer number* which must be keyed in directly, while
n represents such a number which may be specified indirectly via an *RoV* as well (as shown in the addressing tables in **OMS 1**); and
n stands for the respective number itself;

Example: **RSD 12** rounds *x* to 12 significant digits, while
 RSD $\rightarrow 12$ rounds *x* to 15 significant digits.

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 speci-

⁵ Some *HP* calculators do that also with statistical functions returning two values – while the *Spices* (e.g., **HP-34C**) and *Voyagers* (e.g., **HP-15C**) push both results on the *stack* as you expect from *RPN* calculators. The picture shows what the **HP-55**, **-19C/29C**, **-67/97**, **-41C**, and **-42S** do there instead:

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



As far as we know, *HP* did not give any reason for this deviation from plain *RPN* logic until today. In our opinion this is not sensible, so we stick to the paradigm as implemented on the *Spices* and *Voyagers*, then **WP 34S** and **WP 31S** also for **WP43**.

fied indirectly as well; and

r (or *s*) stands for the contents of the *RoV* 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 **OMS 4**); 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 every command **XYZ** requiring one trailing parameter, you can enter **XYZ → X** and it will take its parameter from **X** instead, like a good old traditional *RPN* command.

The **data types (DT)** a particular function operates on are listed in { } in the *IOI* under *Remarks* should there be any restrictions (→ **App. B** for all *DTs* available). Many bit and *integer* functions operate sensibly on **short integers (DT 10)** only – other functions may work with more *DTs*. Functions stating *DT 8* or *9* operate on entire matrices while those stating *DT 8** or *9** operate on the individual matrix elements instead (→ **OMS 3**). Wherever an operation returns a *DT* differing from its input, its output *DT* is listed there as well.⁶

Automatic stack lift (ASL) is enabled after each and every command except **CLX**, **ENTER↑**, **Σ+**, **Σ-**, **M.EDI**, **M.EDIN**, **M.←**, **M.↑**, **M.→**, and **M.↓** – input

⁶ This applies to conversions like °C→°F, for instance: For *real* input (*DT 2*), output will stay *real*. For *integer* input, however, output will become *real*. See there.

Some functions operating on *long integers (DT 1)* will return either such integers or *reals*, depending on input. E.g., **27 $\sqrt[3]{x}$** will return 3, but **28 $\sqrt[3]{x}$** will return a *real number* although input is a *long integer* as well. **$\sqrt[3]{x}$** operating on *short integers (DT 10)* will return *DT 10* always. Compare **OMS 3**.

immediately following one of these ten operations will thus overwrite x instead of pushing it on the *stack*. Reasons are given in [App. H](#).

Labels displayed under **ASSESS**, **PLOT**, **Solver**, **TIMER** () , **TVM**, and **f_{dx}** are printed in the *IOI* but may be not contained in CATALOG'FCNS – some of them denote *system variables*.

If you find some functions in the *IOI* below unknown to you, simply do not bother: you can perfectly enjoy your *WP43* without those. In particular, many of the operations collected in X.FN may well make sense to mathematicians only.

The *IOI* starting overleaf lists the *name* of each and every *item* provided, be it a command, constant, (*sub-*) *menu*, predefined *global label*, *RoV*, or *system flag*.⁷ These are over 1000 *names*. Just single *characters* and digits are not listed below since these do not interfere with other *items*.

The *IOI* (and [App. N](#)) will also support you inventing new names for the *variables* and *menus* you create (avoiding name conflicts).

Each user-created name of a *variable* or *menu* may consist of up to 7 *characters* and **shall be unique** (i.e. it must not match any other *item name* provided or created before – check also [App. N](#)).⁸

Each new name shall start with a letter; it must not contain +, -, ×, /, ^, (,), =, !, |, :, ;, a comma, nor a space. *Names* are case sensitive (e.g., **M** ≠ **m**) but level insensitive (e.g., **B_n*** equals **B_n***, allowing for easy input in equations or catalog searches while the latter notation eases reading). Thus, feel free to create a *variable* **Y_x** but neither **y_x** nor **y_x** since the latter two are in conflict with **y^x**.

Constants taken from CONST are headed by **#** in programs; so you may use *variables* like **a**, **c**, **e**, **h**, **i**, and **k** there, while you must not use these in equations.

Please ensure you have checked these constraints before complaining.

⁷ Functions which **don't work** yet for whatever reasons or were **not checked yet** are marked in column 3. This also applies to the *DTs* stated there.

⁸ *Labels*, on the other hand, do not have to be unique; but multiple use of identical *global program labels* is not recommended since it may carry unexpected consequences.

Item	Keystrokes	Remarks (→ 14ff for general information)
°C→°F	U→ °C → °F	[4] (1) {1, 2} → {2}
°F→°C	etc.	Convert temperatures (→ 152ff).
0.1s	0.1s	[3] (0) Toggles precision of the timer display (1 s / 0.1 s).
10 ^x	(10^x)	[2] (1) {1, 2, 3, 8*, 9*, 10} Returns 10 ^x , the inverse of lg(x).
1COMPL	BIT ▼ 1COMPL INTS ▲ 1COMPL	[4] (0) Sets 1's complement mode for operations on <i>short integers</i> (→ OMS 3).
1/x	(1/x)	[1] (1) {2, 3, 5, 8*, 9*}; {1} → {2} Inverts the number or matrix x.
2COMPL	BIT ▼ 2COMPL INTS ▲ 2COMPL	[4] (0) Sets 2's complement mode for operations on <i>short integers</i> (→ OMS 3).
2 ^x	EXP 2 ^x	[2] (1) {1, 2, 3, 8*, 9*, 10} Returns 2 ^x , the inverse of lb(x).
$\sqrt[3]{x}$	EXP $\sqrt[3]{x}$	[3] (1) {1, 2, 3, 8*, 9*, 10}; ({1} → {2}) Returns the cube root of x.
A	Reserved <i>variable</i> for register A.	
a		Constants (→ 116f).
a ₀		
ABS	CAT. FCNS ABS	Points to x on p. 112. Just for backward compatibility.
ACC	ADV ∫fdx ACC EQN ∫f ACC	Reserved <i>real variable</i> for the relative accuracy of the integrand in integration (→ OMS 5).
ACOS	CAT. FCNS ACOS	Points to arccos. Just for backward compatibility.
ActUSB	USB	[2] (0) Activates the FD (a.k.a. USB disk) for data exchange with your PC. Returning from it, WP43 will continue (→ OMS 4).

Item	Keystrokes	Remarks (→ 14ff for general information)
ac.→ha	  A: ha ← acre etc.	[6] (1) {1, 2} → {2} Convert areas (→ 152ff).
ac.→m ²		
ac _{US} →ha		
ac _{US} →m ²		
ADD	 Σ+	[3] (0) Adds the present timer value to STATS.
ADM		Reserved <i>integer variable</i> for the ADM: 0: DEG, 1: D.MS, 2: RAD, 3: MULπ, 4: GRAD, 5: MIL.
ADV		Menu. → 142.
AGM	 AGM	[3] (2) {2, 3}; {1} → {2} Returns the <i>arithmetic-geometric mean</i> of <i>x</i> and <i>y</i> . Will throw an error for negative <i>x</i> or <i>y</i> . → 326.
AGRAPH	 P.FN2  s	[5+] (0) Alpha graphics. Displays a graphics pattern. The <i>short integer</i> in the source <i>s</i> specifies a 64-pixel column pattern; IP(<i>x</i>) and IP(<i>y</i>) tell the location of the bottom pixel of this column (valid inputs are $0 \leq x \leq 399$, $0 \leq y \leq 175$, pixel (0; 0) is bottom left of the LCD). So one 64 px high graphic row starting in column 0 requires up to 400 such <i>integers</i> to specify – the more blank space is found in this row the less <i>integers</i> are needed for describing it. ⁹ AGRAPH increments <i>x</i> . See examples in OMA 5 . → also GRAMOD on p. 45.
ALL	 ALL n	[3+] (0) Sets the display format to show all digits of a <i>real</i> or <i>complex number</i> as long as displayable (without trailing decimal zeros). For $x \geq 10^{16}$ (or earlier for <i>complex numbers</i>), display will switch to SCI (for ALLENG) or ENG with maximum displayable precision, using the large font; the same will happen a) for n = 0 <u>and</u> $x < 10^{-15}$ or b) for n ≠ 0 <u>and</u> $x < 10^{-n}$. The limits differ in RBR .

⁹ → [HP-42S OM](#), pp. 135 - 140, and [HP-42S PET](#), pp. 195 - 223.

Item	Keystrokes	Remarks (→ 14ff for general information)
		ALL 0 and ALLENG are <i>startup default</i> . ¹⁰ → also ENG, FIX, SCI, and examples in OMS 2.
ALLENG		System flags (→ 129ff)
ALPHA		
ALP.IN		
aMoon		Constant (→ 116f).
AND	AND	[3] (2) {10} Returns the logical AND of the corresponding bits in x and y (→ OMS 3). (2) {1, 2} → {1} Here, x and y are interpreted before executing AND: zero is 'false'; every number $\neq 0$ is 'true'. ¹¹
(ANGLE)		→ $\angle$ on p. 114.
ANGLES	VARS ANGLES	Submenu of tagged angular variables defined at execution time (→ 135f).
arccos	arccos	[2] (1) {3, 8*, 9*}; {1, 2} → {4} Returns the tagged <i>angle</i> $\arccos(x)$. ¹²
arcosh	arcosh	[3] (1) {2, 3, 8*, 9*} Returns $\operatorname{arcosh}(x)$.
	arcosh	
arcsin	arcsin	[2] (1) {3, 8*, 9*}; {1, 2} → {4} Returns the tagged <i>angle</i> $\arcsin(x)$. ¹³

¹⁰ ALL 0 here works almost like ALL does on HP-42S.

¹¹ Input interpretation follows HP-28S in this matter.

¹² Precisely, it returns the principal value of $\arccos(x)$, i.e. a *real* part $\in [0, \pi]$ in $\angle^r$, $\in [0, 1]$ in $\angle\pi$, $\in [0^\circ, 180^\circ]$ in $\angle^\circ$ or $\angle''$, $\in [0^-, 1600^-]$ in $\angle^-$, or $\in [0^g, 200^g]$ in $\angle^g$.
→ ISO/IEC 9899.

¹³ Precisely, it returns the principal value of $\arcsin(x)$, i.e. a *real* part $\in [-\pi/2, \pi/2]$ in $\angle^r$, $\in [-0.5, 0.5]$ in $\angle\pi$, $\in [-90^\circ, 90^\circ]$ in $\angle^\circ$ or $\angle''$, $\in [-800^-, 800^-]$ in $\angle^-$, or $\in [-100^g, 100^g]$ in $\angle^g$.
→ ISO/IEC 9899.

Item	Keystrokes	Remarks (→ 14ff for general information)
arctan	(TRI) arctan	[2] (1) {3, 8*, 9*}; {1, 2} → {4} Returns the tagged <i>angle</i> $\arctan(x)$. ¹⁴
arsinh	(EXP) arsinh	[3] (1) {2, 3, 8*, 9*} Returns $\operatorname{arsinh}(x)$.
	(TRI) arsinh	
artanh	(EXP) artanh	[3] (1) {2, 3, 8*, 9*} Returns $\operatorname{artanh}(x)$.
	(TRI) artanh	
ASIN	(CAT.) FCNS ASIN	Points to arcsin . Just for backward compatibility.
ASLIFT		System flag (→ 129ff)
ASR	(BIT)  ASR <u>n</u>	[5+] (1) {10}  Shifts all bits in x right n times. ¹⁵ Corresponds to a signed division of x by 2^n (with $n \leq 63$). ASR 0 executes as NOP but loads L . Compare SR on p. 87 and OMS 3 .
ASSESS	(FIT)  ASSESS	[4] (0) Opens a new screen, plots the data points collected in STATS together with the regression curve fitted, and writes its parameters and r^2 next to the plot (→ 296ff, OMS 2 and OMA 4). If BestF (→ 25) was used to select ≥ 2 fit models then all allowed and calculable models can be displayed sequentially within ASSESS : Use (NXTFIT) or  to plot the next model,  to plot previous model, and (ZOOM) for zooming out. Press (EXIT) to leave the ASSESS screen.
ASSIGN	(ASN) <i>item, location</i>	[2++] (0) Assigns an <i>item</i> (like a function, <i>menu</i> , program <i>label</i> , or <i>character</i>) to a specific target location on the keyboard or in a <i>user menu</i> . Your assignments will become effective in <i>user mode</i> . Note ASSIGN works in <i>PEM</i> and <i>user mode</i> as well. → examples in OMS 7 .

¹⁴ Precisely, it returns the principal value of $\arctan(x)$, i.e. a *real* part $\in [-\pi/2, \pi/2]$ in $\mathbb{R}$, for example (→ **ARCSIN**), if **SPCRES** is set. Else the result interval for **ARCTAN** becomes, e.g., $(-\pi/2, \pi/2)$ in $\mathbb{R}$. → **ISO/IEC 9899**.

Item	Keystrokes	Remarks (→ 14ff for general information)
ATAN	CAT. FCNS ATAN	Points to arctan . Just for backward compatibility.
atan2	TRI atan2	[3] (2) {3}; {1, 2} → {4} Returns $\arctan2(x, y)$, → 326. Watch the order of parameters. Example: 1 ENTER +/− TRI atan2 returns 135°.
atm→Pa	U→ F&p: ▼ Pa ← atm	[6,4] (1) {1, 2} → {2} Convert pressures and distances (→ 152ff).
au→m	U→ x: m ← au	
AUTOFF		System flags (→ 129ff)
AUTXEQ		
AVIEW	P.FN AVIEW s	[4+] (0) Displays <i>s</i> left-adjusted in the top numeric row (and below if needing more space) until the next key is pressed (in programs until the next AVIEW, VIEW, INPUT, PAUSE, PROMPT, STATUS, or STOP occurs). The name of the source will <u>not</u> be shown (compare VIEW and PROMPT).
A:	U→ A:	Submenu for conversions of areas (→ 152).
a_⊕		Constant (→ 116f).
B	Reserved variable for register B .	
BACK	P.FN P.FN2 BACK n	[5+] (0) Jumps <i>n</i> steps backwards ($0 \leq n \leq 255$) in a program. E.g., BACK 1 goes to the previous program step. If BACK attempts to cross an END, an error will be thrown. Reaching step 0000 stops program execution. Compare SKIP on p. 84.¹⁶

¹⁵ Works like *n* consecutive executions of ASR on [HP-16C](#).

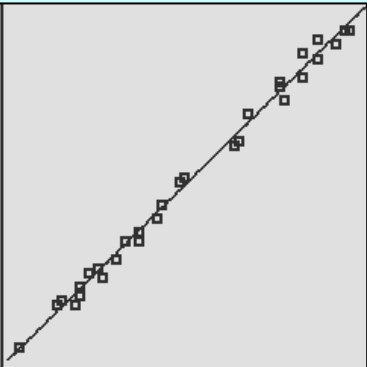
¹⁶ **ATTENTION:** If you edit a section of your routine crossed by one or more **BACK**, **CASE**, or **SKIP** jumps, this may well result in a need to **manually maintain all those statements**. Actually, these 3 commands are for quick and dirty routines.

Item	Keystrokes	Remarks (→ 14ff for general information)																		
bar→Pa	   ← 	  {1, 2} → {2} Converts pressures (→ 152ff).																		
bbl→m ³	    ← 	  {1, 2} → {2} Converts volumes (→ 152ff).																		
BC?	  	  {10} Tests if the specified bit in <i>x</i> is clear.																		
BEEP	 	  Sounds a sequence of four tones (compare TONE).																		
BeginP	  	  Sets “Begin” mode in <i>TVM</i> : then payments occur at the begin of each period. Typical for saving plans and leasing. Clears ENDPMT . Compare EndP . → OMS 6 and OMA 3 .																		
BestF	   	  Instructs your <i>WP43</i> to select the ‘best’ curve fit model for the accumulated statistical data (i.e. the one returning the maximum <i>coefficient of determination</i> (r^2) out of the models allowed, → CORR). ¹⁷ Relevant for L.R. , $\hat{x}$, and $\hat{y}$. You can accelerate computation significantly by excluding curve fit models making no sense for your data (e.g., for physical or technical reasons). This information is carried by <i>n</i> . Each fit model corresponds to a number as listed: <table><tr><td>LinF</td><td>1</td><td>ExpF</td><td>2</td><td>LogF</td><td>4</td></tr><tr><td>PowerF</td><td>8</td><td>RootF</td><td>16</td><td>HypF</td><td>32</td></tr><tr><td>ParabF</td><td>64</td><td>CauchF</td><td>128</td><td>GaussF</td><td>256</td></tr></table> Take the numbers of all models you can exclude and add them up – the result is <i>n</i> . → OMS 2 . We recommend using ASSESS for checking the fit(s) (→ 296ff).	LinF	1	ExpF	2	LogF	4	PowerF	8	RootF	16	HypF	32	ParabF	64	CauchF	128	GaussF	256
LinF	1	ExpF	2	LogF	4															
PowerF	8	RootF	16	HypF	32															
ParabF	64	CauchF	128	GaussF	256															

¹⁷ Works almost like **BEST** on **HP-42S**.

Item	Keystrokes	Remarks (→ 14ff for general information)
Binom _p	PROB  Binom: Binom_p etc.	[5] (1) {1, 2} → {2}
Binom _Δ		<i>Binomial distribution</i> with the number of successes g in X , the probability of a success p₀ in I , and the sample size n in J (→ 273).
Binom _Δ		Binom⁻¹ returns the maximum number of successes m for a given probability p in X , p₀ in I and n in J .
Binom ⁻¹		
Binom:	PROB  Binom:	Submenu, → 146.
BIT	BIT	Menu, → 138.
B _n	X.FN B_n	[3] (1) {1, 2}
B _n [*]	X.FN B_n[*]	B_n and B_n[*] return the <i>Bernoulli</i> number for an integer n > 0 given in X , working with different definitions (→ 271).
BS?	BIT BS? n	[4+] (0) {10}
		Tests if the specified bit in <i>x</i> is set.
Btu→J	 E: J ← Btu	[4] (1) {1, 2} → {2}
		Converts energies (→ 152ff).
BUZZ	I/O BUZZ	[4] (0) Plays a tone with a frequency <i>y</i> (in Hz) and duration <i>x</i> (in ms).
C	Reserved <i>variable</i> for register C.	
c		Constants (→ 116f).
c ₁		
c ₂		
cal→J	 E: J ← cal	[4] (1) {1, 2} → {2}
		Converts energies (→ 152ff).
CALC	EQN Solver Calc	[4] (1) Computes the value of the current equation at position <i>x</i> (→ <i>OMS</i> 5). ATTENTION: You shall store <i>x</i> in the corresponding <i>variable</i> using the respective <i>softkey</i> before calling CALC .

Item	Keystrokes	Remarks (→ 14ff for general information)
CARRY		System flag (→ 129ff). Set by SF C .
CASE	P.FN P.FN2 CASE s	[5+] (0) Works like SKIP (→ 84) but takes the number of steps to skip ($0 \leq s \leq 255$) 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> Executing this program, r12 will be checked in step 100: if r12 ≤ 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 $1 \leq r12 \leq 4$. ATTENTION: CASE might surprise you for r12 > 4 here. Take care of the input you provide! And watch f. 16 on p. 24
CATALOG	CATALOG	Catalog of everything, → 135ff.
CauchF	FIT  CauchF	[5] (0) Selects the <i>Cauchy</i> (a.k.a. <i>Lorentz</i> or <i>Breit-Wigner</i>) peak fit model for L.R. Relevant also for CORR, $\hat{x}$, and $\hat{y}$. → 281ff.
Cauch _p	PROB  Cauch:	[5] (1) {1, 2} → {2} <i>Cauchy</i> (a.k.a. <i>Lorentz</i> or <i>Breit-Wigner</i>) distribution with the peak location x₀ specified in I and the shape γ in J (→ 276). Cauch⁻¹ returns x for a given probability p in X, with x₀ in I and γ in J.
Cauch _Δ	Cauch_p	
Cauch _Δ	etc.	
Cauch ⁻¹		

Item	Keystrokes	Remarks (→ 14ff for general information)
Cauch:	PROB  Cauch:	Submenu, → 146.
CB	BIT CB <u>n</u>	[4+] (1) {10} Clears the specified bit in x .
ceil	INTS ceil	[4] (1) {8*}; {1, 2} → {1} Returns the smallest <i>integer</i> $\geq x$. Compare FLOOR .
CENTRL	FIT  PLOT CENTRL	[5] (0) Fits an <i>orthogonal regression</i> center line through the data points plotted by PLOT (→ 70) for an <i>MSA</i> (→ 299ff); displays also the two fit parameters (y-intercept and slope) with their errors and (for ≥ 30 data points) the measured precision of the measuring instrument. → 85 for s_{mi} , $s(a)$ on p. 89, also pp. 299ff, and OMS 2 for an example. 2022-03-14 22:02 Orthogonal $y = a_0 + a_1 x$ $a_0 = 0.422$ ± 0.581 $a_1 = 0.989$ $\pm 1.520e-02$ $s_{mi} = 1.29$  CENTRL s_{mi}
CF	CF <u>n</u> CLR CF <u>n</u> FLAG CF <u>n</u> MODE CF <u>n</u>	[2+] (0) Clears the <i>flag</i> specified, i.e. sets it to 0 .

Item	Keystrokes	Remarks (→ 14ff for general information)
CHARS	CAT. CHARS	Submenu of characters, → 135ff.
chī→m	U→ x: ▲ m ← chī	[6] (1) {1, 2} → {2} Converts distances (→ 152ff).
CLall	CLR CLall	[4] (0) Clears <u>all</u> RoV's, numbered <i>user flags</i> , user assignments, and programs in <i>RAM</i> . Modes will stay as they are. → also CLCVAR , CLFall , CLMall , CLPall , CLVall , CLREGS , CLSTK , RESET , and OMS 1 .
CLBKUP	CLR CLBKUP	[4] (0) Deletes the backup file created by SAVE .
CLCVAR	CLR CLCVAR	[3] (0) Clears all <i>variables used</i> in <i>current program</i> : it creates the <i>variables</i> not defined yet and sets all such <i>real</i> , <i>complex</i> , and <i>integer variables</i> to zero, all elements of all <i>matrix variables</i> to 0. , all <i>angles</i> to 0. , all <i>times</i> to 0:00:00 , all <i>dates</i> to January 1st of year 0 , and all <i>text strings</i> to zero length.
CLFall	FLAG CLFall	[4] (0) Clears all <u>numbered</u> global and local <i>user flags</i> (in programs without confirmation; compare CF). Lettered <i>flags</i> are not cleared since they may coincide with <i>system flags</i> .
	CLR CLFall	
CLK	CLK	Menu, → 135.
CLKEYS	CLR ▲ CLKEYS	[4] (0) Clears <u>all</u> assignments (→ ASSIGN).
CLLCD	CLR CLLCD	[4] (0) Clears all pixels $\geq IP(x)$ and $\geq IP(y)$ on the <i>LCD</i> . Pixel (0; 0) is bottom left.
CLMall	CLR CLMall	[4] (0) Clears <u>all</u> user defined <i>menus</i> in <i>RAM</i> .
CLMENU	CLR ▲ CLMENU	[4] (0) Clears all <i>menu</i> key definitions for the programmable <i>menu</i> (→ MENU on p. 59).
	P.FN P.FN2 ...	
CLMy...	CLR CLMy...	Submenu, → 143.
CLMyM	CLR CLMy... CLMyM etc.	[4] (0) Clears all user assignments in <u>MyMENU</u> , <u>MyPFN</u> , or <u>Myα</u> , i.e. resets the respective menu to its <i>SDC</i> .
CLMyP		
CLMyα		

Item	Keystrokes	Remarks (→ 14ff for general information)
CLP	CLR CLP	[3] (0) Clears the <i>current program</i> . Freed memory is returned to the pool of free space.
CLPall	CLR CLPall	[4] (0) Clears <u>all</u> programs in <i>RAM</i> . Compare CLP.
CLR	CLR	Menu, → 143.
CLREGS	CLR CLREGS (needs confirmation in <i>RUM</i>)	[4] (0) Clears all global and local <i>GP registers</i> allocated, i.e. sets them to zero. The contents of the <i>stack</i> and L are kept; compare CLSTK.
CLSTK	CLR CLSTK	[3] Clears all <i>stack registers</i> currently allocated (either X ... T or X ... D). All other <i>register</i> contents are kept. Compare CLREGS.
	0 FILL	
CLVall	CLR CLVall	[4] (0) Clears <u>all</u> user defined <i>variables</i> in <i>RAM</i> .
CLX	CLR CLX	[1] (1) Clears <i>stack register X</i> , disabling <i>ASL</i> . Compare CLSTK and CLREGS.
	← for closed <i>x</i>	
CLΣ	CLR CLΣ	[3] (0) Clears STATS and HISTO and releases the memory allocated for them and for the statistical sums. Call CLΣ before accumulating any data for a new statistical analysis.
	FIT CLΣ	
	STAT CLΣ	
cn(u,m)	X.FN Ellipt cn	[4] (2) {2, 3} Returns the <i>elliptic cosine</i> of the argument $u = x$ and parameter $m = y$ (→ 327).
CNST	P.FN CNST <i>n</i>	[4+] (-1) { } → {2} Returns the constant stored at position <i>n</i> in the <i>catalog</i> CNST . Allows for addressing constants indirectly, also beyond ∞ (→ 116ff).

Item	Keystrokes	Remarks (→ 14ff for general information)
COMB	PROB C_yx	[3] (2) {1} Returns the number of possible <u>subsets</u> of x objects taken out of a set of y objects (i.e. choose x out of y). No object occurs more than once in a subset, and <u>different orders</u> of the same x objects are <u>not counted</u> separately. Compare PERM. (2) {2, 3} → 271f for the formula.
conj	CPX conj	[3] (1) {3, 9*} Returns the <i>complex conjugate</i> of x.
CONFIG	CAT. VARS CONFIG	Submenu of all <i>variables</i> storing <i>configurations</i>, → 135f.
CONST	CONST	Catalog of physical, mathematical, astronomical, and surveying constants (→ 116ff and CNST).
CONVG?	TEST CONVG? s	[4+] (0) {2} Checks for convergence by comparing x and y as determined by the lowest five bits of s. a) The very lowest 2 bits set the tolerance limit: $0 = 10^{-14}$, $1 = 10^{-24}$, $2 = 10^{-32}$. b) The next two bits determine the comparison mode using the tolerance limit set: 0 = compare the numbers x and y relatively, 1 = compare them absolutely. c) The top bit tells how special numbers will be treated: 0 = NaN and $\pm\infty$ are considered converged, 1 = they are not considered converged. At the bottom line, $s = a + 4b + 16c$.
CORR	FIT r	[3] (-1) {} → {2} Returns the <i>coefficient of correlation</i> r for the current statistical data and fit model. Note r^2 is the <i>coefficient of determination</i>. → 281ff.

Item	Keystrokes	Remarks (→ 14ff for general information)
cos	TRI cos	[2] (1) {2, 3, 8*, 9*}; {1, 4} → {2} Returns the cosine of the <i>angle</i> x (as tagged, else assumes the current <i>ADM</i> ; → <i>OMS</i> 3).
cosh	EXP cosh	[3] (1) {2, 3, 8*, 9*}
	TRI cosh	Returns the hyperbolic cosine of x .
COV	FIT cov	[3] (-1) { } → {2} Returns the <i>population covariance</i> for the two data sets $\{x\}$ and $\{y\}$ entered via Σ+ . → s_{xy} for the <i>sample covariance</i> and pp. 281ff.
cp _I →mℓ	U→ V: ▲ ▲ mℓ ← cup _I	[7,6] (1) {1, 2} → {2}
cp _{UK} →mℓ	U→ V: ▲ mℓ ← cup _{UK}	Convert kitchen volumes (→ 152ff).
cp _{US} →mℓ	U→ V: ▼ mℓ ← cup _{US}	
CPX	CPX	Menu, → 143.
CPXRES		System flag (→ 129ff), set by SF I (imag.).
CPXS	CAT. VARs CPXS	Submenu of all <i>complex variables</i> defined at execution time, → 135f.
CPXSLV	EQN Solver cp _x Slv	[4] Finds <i>complex</i> (and <i>real</i>) roots of the current equation in EQN (SOLVE , on the other hand, will find only <i>real</i> roots). Two initial guesses of the root may be supplied in X and Y when calling CPXSLV . Its results follow the rules of SOLVE (→ 86). There is no way to force CPXSLV to find a root with <i>imaginary</i> part $\neq 0$ (or $= 0$) – you should try with different (or swapped) first guesses. With a root found, press CALC to verify.
CPX?	TEST ▲ CPX?	[4] (0) Checks if x is <i>complex</i> . Returns true if X contains data of type 3 or 9 with nonzero <i>imaginary</i> part (→ also <i>App. B</i>).

Item	Keystrokes	Remarks (→ 14ff for general information)
cross	MATX cross	[4] (2) {8} With two <i>real</i> 3D vectors x and y , cross returns their cross product $y \times x$, perpendicular to x and y .
	CPX cross	[3] (2) {3} → {2} With either two <i>real</i> 2D vectors or two <i>complex numbers</i> x and y , cross returns a <i>real number</i> equal to the signed <i>magnitude</i> of the resulting moment vector (→ OMS 3).
crt→g	U → m: ▾ g ← carat	[5,6,4] (1) {1, 2} → {2} Convert masses and distances (→ 152ff).
crt→kg	U → m: ▲ kg ← carat	
cùn→m	U → x: ▲ m ← cùn	
cwt→kg	U → m: kg ← cwt	
CX→RE	CPX CX→RE	[1] (-1) {3} → {2}; {9} → {8}
	CC (this short-cut works in <i>RUM</i> only)	Cuts a closed <i>complex</i> object x , putting either • (for L) its <i>real</i> part in Y and its <i>imaginary</i> part in X or • (for ⊙) its <i>magnitude</i> in Y and its <i>phase</i> in X.
D	Reserved <i>variable</i> for register D .	
DATE	CLK DATE	[3] (-1) {} → {6} Recalls date and time from the real-time clock and displays it in the format selected by you (→ D.MY , M.DY , and Y.MD). Furthermore, DATE shows the day of week (→ OMS 3).
DATES	CAT. VAR DATES	Submenu of all <i>date</i> variables defined at execution time, → 135f.
DATE→	CLK DATE→	[3] (-2) {2, 6} → {1} Takes x as a <i>date</i> in the format selected (or a <i>real number</i> in corresponding format) and pushes its three components as <i>integers</i> on the <i>stack</i> . Reversible by → DATE (→ 109).

Item	Keystrokes	Remarks (→ 14ff for general information)
DAY	CLK DAY	[4] (1) {2, 6} → {1} Takes <i>x</i> as a <i>date</i> in the format selected (or a <i>real number</i> in corresponding format) and extracts the day. Compare WDAY on p. 94.
day→s	U→ Time: s ← day	[4] (1) {1, 2} → {2} Converts time, → 152ff.
DBLR	INTS DBLR	[4] {10} Double <i>word</i> length commands for remainder, multiplication, and division. DBL× takes <i>x</i> and <i>y</i> as factors as usual but returns the product in X and Y (with its most significant bits in X). This result may be up to 128 bits long. DBL/ and DBLR accept a double size dividend in Y and Z (with its most significant bits in Y), the divisor in X as usual, and return the result in X . Thus, DBL/ and DBLR are <i>triadic</i> functions.
DBL×	INTS DBL×	
DBL/	INTS DBL/	
dB→fr	U→ dB → field ratio	[4] (1) {1, 2} → {2} Convert ratios (→ 152ff).
dB→pr	... dB → power ratio	
DEC	LOOP DEC <i>r</i>	[4+] (0) {1, 2, 10} Decrements <i>r</i> by 1. Does not load L even for target address X .
DECIM.		System flag (→ 129ff)
DECOMP	PART DECOMP	[3] (-1) {1, 2} → {1} Decomposes <i>x</i> (after converting it to an <i>improper fraction</i> , if applicable), returning a <i>stack</i> [denominator(x) , numerator(x) , ...]. This works with maximum precision always, regardless of the settings of D.MAX , DENFIX , and DENANY . Can be reverted by division. Example: For <i>x</i> = 2.25 , DECOMP returns <i>y</i> = 9 and <i>x</i> = 4 .
DEG	MODE DEG	[3] (0) Sets the ADM to <i>decimal degrees</i> .

Item	Keystrokes	Remarks (→ 14ff for general information)
DEG→	DEG→	[4] (1) {1, 2, 4} → {4} Converts <i>angles</i> as described on p. 163.
DelITM	DelITM	[4] (0) Deletes a user-defined <i>item</i> from memory.
DelMall	DelMall	[4] (0) Deletes all entirely user-defined <i>menus</i> or <i>variables</i> from memory (compare CLMall and CLVall). ↓ OMS 7
DelVall	DelVall	
DENANY		System flags (→ 129ff)
DENFIX		
DENMAX	DENMAX	[4] (1) Specifies the maximum denominator for fractions ($1 \leq x \leq 9999$). For $x < 1$ or $x > 9999$, D.MAX will be set to 9999 . ¹⁸ For $x = 1$, the current D.MAX setting is recalled, replacing x .
DET	FCNS DET	Equals M on p. 112. Just for backward compatibility.
DISP		Menu, → 143.
DMCP	DMCP	[4] (0) Returns to SM's DMCP menu, → App F.
DMY		System flag (→ 129ff).
dn(u,m)	Ellipt dn	[4] (2) {2, 3} Returns the elliptic <i>delta amplitudinis</i> of the argument $u = x$ and parameter $m = y$ (→ 327).
dot	dot	[4] (2) {8} → {2}; {9} → {3}
	dot	Requires two matrices in X and Y and returns their dot (scalar) product. This product is the sum of the products of the corresponding elements in both matrices. Note both matrices must match in size, else dot will throw an error. → OMS 3. [3] (2) {3} → {2} Returns $Re(x) Re(y) + Im(x) Im(y)$

¹⁸ Works like $\frac{1}{c}$ on the *HP-35S*, but allows for a greater maximum denominator, thus higher precision.

Item	Keystrokes	Remarks (→ 14ff for general information)
DRAW	EQN Solver Draw	[4] (0) Opens a new screen and draws the graph for the current equation within the boundaries specified. → OMS 5 for examples.
	EQN ∫f Draw	[4] (0) Opens a new screen and draws the integrand within the limits specified (↓Lim and ↑Lim).
DROP	DROP↓	[2] Drops x ... from the <i>stack</i> (→ OMS 1).
DROPy	STK DROPy	[3] Drops y
DSE	LOOP DSE r	[3+] (0) {1, 2, 10} Most useful in a program running. Given $r = \text{cccccc.fffii}$, DSE decrements r by ii and will skip next program step if $\text{cccccc} \leq \text{fff}$ then (→ rear side of your <i>WP43</i>). If r features no fractional part then fff is 0. If $ii = 0$, cccccc will be decremented by 1. DSE does not load L even for target address X. Note that neither fff nor ii can be negative, and DSE makes sense with $\text{cccccc} > 0$ only. In <i>RUM</i>, DSE returns true for $\text{cccccc} > \text{fff}$ and false for $\text{cccccc} \leq \text{fff}$.
DSL	LOOP DSL r	[3+] (0) {1, 2, 10} Works like DSE but skips if $\text{cccccc} < \text{fff}$.
DSP	DSP n	[2+] (0) Specifies n quickly for ALL , ENG , FIX , or SCI , whatever is currently chosen.
DSTACK	DISP ▲ DSTACK n	[5+] (0) Sets the number of <i>stack registers</i> to be 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. Regardless of the DSTACK set, expanded views of matrices and dual- or multi-level returns like SUM will work as described; command input will be echoed directly below the <i>status bar</i> and also error messages will be displayed there.

The drawing is based on 41 calculated points.

Item	Keystrokes	Remarks (→ 14ff for general information)
		DSTACK is dedicated to old-school <i>RPN</i> users who may feel distracted by a multitude of <i>stack registers</i> displayed changing simultaneously while only the lowest are really relevant.
DSZ	 DSZ <i>r</i>	[3+] (0) {1, 2, 10} Decrements <i>r</i> by 1 and skips the next step if $-1 < r < +1$ thereafter. Does not load L even for target address X . ¹⁹
DUMP	 DUMP	[3] (0) Dumps <i>x</i> (in full precision) onto the <i>FD</i> in SAVFILES/regx- <i>yyyyymmdd-hhmmssdd</i> .tsv. This is how to export also huge integers (>298 digits) from the calculator (→ App. E).
D.MAX		Reserved <i>integer variable</i> for the maximum denominator in fractions, set by DENMAX .
D.MS	 for closed <i>x</i>	[2] (0) Sets the <i>ADM</i> to <i>sexagesimal degrees</i> .
D.MS→	 D.MS→	[4] (1) {1, 2, 4} → {4}
D.MS→D	 D.MS→D	Convert <i>angles</i> as described on p. 163.
D.MY	  D.MY	[4] (0) Sets the <i>date</i> format dd.mm.yyyy.
D→D.MS	 D→D.MS	[4] (1) {1, 2, 4} → {4} Converts <i>angles</i> as described on pp. 163f.
D→J	 D→J	[4] (1) {2, 6} → {1} Takes <i>x</i> as a <i>date</i> in the format selected (or a <i>real number</i> in corresponding format) and converts it to a <i>Julian day number</i> according to the <i>J/G</i> setting (→ 51).
D→MIL	  D→MIL	[4] (1) {1, 2, 4} → {4}
D→R	 D→R	Convert <i>angles</i> as described on pp. 163f.
e		Constant (→ 116ff).

¹⁹ **DSZ** is known from *HP-65*, *-19C/29C*, *-67*, and *-16C*.

Item	Keystrokes	Remarks (→ 14ff for general information)
EDIT	EDIT (→ 428)	[2] (0) In <i>RUM</i> , opens the object <i>x</i> for editing. In <i>PEM</i> , opens the <i>current program step</i> for editing (i.e. the step the <i>PP</i> is pointing to).
EIGVAL	MATX EIGVAL	[5] (-1) {8, 9} Evaluates the matrix <i>x</i> and pushes the matrix of its eigenvalues on the <i>stack</i> .
EIGVEC	MATX EIGVEC	[5] (-1) {8, 9} Evaluates the matrix <i>x</i> and pushes the matrix of its eigenvectors (as column vectors) on the <i>stack</i> .
Ellipt	X.FN Ellipt	Submenu for elliptic functions, → 148.
END	P.FN END	[3] (0) Last command in a <u>program</u> and terminal for searching <i>local labels</i> (→ OMS 4). Can't be skipped by a test. Works like RTN in all other aspects (→ 79).
EndP	FIN TVM End	[5] (0) Sets “End” mode in <i>TVM</i> : Payments occur at the end of each period. Typical for loans and investments. Sets ENDPMT . Compare BeginP .
ENDPMT		System flag (→ 129ff)
ENG	DIS.P ENG <i>n</i>	[3+] (0) Sets the engineer's display format (→ OMS 2) with $0 \leq n \leq 15$. E.g., 12345.67 will display as 12.35×10^3 for ENG 3 and as 12.3457×10^3 for ENG 5 . 10^0 will not be shown.
ENORM	MATX ENORM	[4] (1) {8, 9} → {2} Calculates the <i>Euclidean</i> norm of the matrix in X , i.e. $\sqrt{\sum x_{ij}^2}$. For a vector, ENORM returns its length. Compare x on p. 112.
ENTER↑	ENTER↑	[1] Separates two inputs. Closes <i>x</i> and copies it into Y . Disables <i>ASL</i> , so X can take a new entry without interfering with the prior one. → 169, App. H , OMS 1 , and OMA 1 . Note that ENTER↑  works like EXIT .

L will not be loaded.
→ [OMS 3](#).

Item	Keystrokes	Remarks (→ 14ff for general information)
ENTRY?	TEST ENTRY?	[4] (0) Checks the (internal) entry <i>flag</i>. It is set if ... - any <i>character</i> is entered in <i>AIM</i>, 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	EQN	Menu, → 143.
EQ.DEL	EQN DELETE	[4] (0) Deletes the current equation.
EQ.EDI	EQN EDIT	[3] (0) Opens the <i>Equation Editor</i> to edit the current equation or to enter a new one. OMS 5
EQ.NEW	EQN NEW	
erf	X.FN erf	[3] (1) {1, 2} → {2}
erfc	X.FN erfc	Returns the error function or its complement, → 326ff.
ERR	P.FN P.FN2 ERR <i>n</i>	[5+] (0) Raises the error code specified (→ App. C). The consequences are the same as if the corresponding error really occurred, so, e.g., a running routine will be stopped and the error message will be thrown. Compare MSG .
EVEN?	TEST EVEN?	[4] (0) Checks if <i>x</i> is <i>integer</i> (→ 49) and even.
(EXIT)	EXIT	→ 171. Note that EXIT  works like NOP but closes <i>x</i> .
EXITall	P.FN P.FN2 EXITall	[4] (0) Exits all <i>menus</i> in <i>RUM</i> . In <i>PEM</i> , enters a step EXITALL .
	EXIT pressed for > 1 s	[1] (0) Exits all <i>menus</i> .
EXP	EXP	Menu, → 143.
ExpF	FIT  ExpF	[4] (0) Selects the exponential fit model for L.R. Relevant also for corr , $\hat{x}$, and $\hat{y}$ (→ 281ff).

Item	Keystrokes	Remarks (→ 14ff for general information)
Expon _p	PROB  Expon:	[5] (1) {1, 2} → {2} <i>Exponential distribution</i> with the rate λ in I (→ 273ff). Expon⁻¹ returns the survival time t_s for a given probability p in X , with λ in I .
Expon _Δ	Expon_p etc.	
Expon _Δ		
Expon ⁻¹		
Expon:	PROB  Expon:	Submenu, → 146.
EXPT	PART EXPT	[3] (1) {1, 2} → {1} Returns the exponent h of $x = m \cdot 10^h$. Compare MANT .
E(m)	X.FN Ellipt E(m)	[4] (1) {2, 3} Returns the <i>complete elliptic integral of the 2nd kind</i> for $m = x$, → 327.
E(φ,m)	X.FN Ellipt E(φ,m)	[5] (1) {2, 3} Returns the <i>incomplete elliptic integral of the 2nd kind</i> for $\varphi = x$ for $m = y$, → 327.
E:	U→ E:	Submenu for energy conversions, → 152.
e		Constant (→ 116ff).
e ^x		[1] (1) {2, 3, 8*, 9*, 10}; {1} → {2} Returns the exponential e^x , the inverse of $\ln(x)$. Note $e^{i\pi} = -1$.
e ^{x-1}	EXP e^{x-1}	[3] (1) {2, 8*} For $x \approx 0$, this returns a more accurate result for the fractional part than e^x does.
F		Constant (→ 116ff).
FACTOR	X.FN FACTOR	[4] (1) {1, 10} → {8} Factorizes x . Returns a matrix with the prime factors of x in row 1 and the corresponding exponents in row 2. → 272 for an example and more, → 197f for limitations.

Item	Keystrokes	Remarks (→ 14ff for general information)
FASTFN		System flag (→ 129ff)
FB	[BIT] FB <u><i>n</i></u>	[4+] (1) {10} Inverts ('flips') the specified bit in <i>x</i> .
FBR	[α.FN] FBR	[4] (0) Font browser. Shows all <i>characters</i> designed for your WP43.
FCNS	[CATALOG] FCNS	Catalog of all functions provided, → 135ff.
FC?	[FLAG] FC? <u><i>n</i></u>	[3+] (0) Tests if the specified <i>flag</i> is clear.
FC?C	[FLAG] FC?C <u><i>n</i></u> etc.	[4+] (0) Test if the specified <i>flag</i> is clear. Will clear, flip, or set this <i>flag</i> after testing, respectively.
FC?F		
FC?S		
fēn→m	[U→] x: [▼] m ← fēn	[6] (1) {1, 2} → {2} Converts distances, → 152ff.
FF	[FLAG] FF <u><i>n</i></u>	[3] (0) Flips the <i>flag</i> specified.
FIB	[X.FN] FIB	[4] (1) {1} Returns the <i>Fibonacci number</i> (max. input is 4791, returning a 1001-digit <i>integer</i>). → 271f. [1] {2, 3} Returns the extended <i>Fibonacci number</i> .
FILL	[FILL]	[2] Fills the entire <i>stack</i> with <i>x</i> .
FIN	[FIN]	Menus, → 144.
FIT	[FIT] (→ 428)	
FIX	[DISP] FIX <u><i>n</i></u>	[3+] (0) Sets fixed point display format (→ OMS 2); $0 \leq n \leq 15$ is the number of decimals of the mantissa to be displayed.
FLAG	[FLAG]	Menu, → 144.
floor	[INTS] floor	[4] (1) {8*}; {1, 2} → {1} Returns the greatest <i>integer</i> ≤ <i>x</i> . Compare CEIL .

Item	Keystrokes	Remarks (→ 14ff for general information)
FP	[PART] FP	[3] (1) {1, 2, 8*, 10}
	[#] [F] for closed x	Returns the fractional part of x . Compare IP .
FP?	[TEST] FP?	[4] (0) Tests x for having a fractional part $\neq 0$.
$F_p(x)$	[PROB] [F:] $F_p(x)$ etc.	[4] (1) {1, 2} → {2} <i>Fisher's F distribution</i> . The <i>degrees of freedom</i> shall be specified in I and J (→ 273ff). $F^{-1}(p)$ corresponds to F_p on <i>HP-21S</i> .
$F_{\Delta}(x)$		
$F_{\Delta}(x)$		
$F^{-1}(p)$		
FRACT		System flags (→ 129ff)
FRCSRN		
fr→dB	[U→] [▲] db ← field ratio	[4] (1) {1, 2} → {2} Converts ratios, → 152ff.
FS?	[FLAG] FS? <u>n</u>	[3+] (0) Tests if the specified <i>flag</i> is set.
FS?C	[FLAG] FS?C <u>n</u> etc.	[4+] (0) Test if the specified <i>flag</i> is set. Will clear, flip, or set this <i>flag</i> after testing, respectively.
FS?F		
FS?S		
ftm.→m	[U→] [x:] [▲] m ← fathm	[6,5] (1) {1, 2} → {2} Convert distances (→ 152ff).
ft.→m	[U→] [x:] m ← foot	
ft _{US} →m	[...] [▼] m ← surv. foot _{US}	
FV	[FIN] TVM FV	Reserved <i>real variable</i> for the future value of your investment or loan in <i>TVM</i> (→ <i>OMS 6</i>).
fz _{UK} →m ³	[U→] [■] V: m³ ← floz _{UK}	[6,7] (1) {1, 2} → {2} Convert volumes (→ 152ff).
fz _{UK} →mℓ	[...] V: [▲] mℓ ← floz _{UK}	
fz _{US} →m ³	[...] V: m³ ← floz _{US}	
fz _{US} →mℓ	[...] V: [▼] mℓ ← floz _{US}	

Item	Keystrokes	Remarks (→ 14ff for general information)
F_{α}		Constants (→ 116ff).
F_{δ}		
$F(\varphi, m)$	X.FN Ellipt $F(\varphi, m)$	[5] (1) {2, 3} Returns the <i>incomplete elliptic integral of the 1st kind</i> for $\varphi = x$ for $m = y$, → 327.
F:	PROB F:	Submenu for the <i>F distribution</i> , → 146.
f'	EQN f'	Submenus for interactively computing the 1 st or 2 nd derivative of a given equation, → OMS 5 .
f''	EQN f''	
$f'(x)$	ADV $f'(x)$ label	[3+, 4+] {1, 2} → {2} $f'(x)$ [f''(x)] returns the 1 st [2nd] derivative of the function $f(x)$ at position x . This $f(x)$ must be specified in a routine starting with LBL label . On return, Y , Z , and T will be cleared and the position x will be in L . → OMS 5 . ATTENTION: $f'(x)$ and $f''(x)$ fill the <i>stack</i> with x before calling the routine specified.
$f''(x)$	ADV $f''(x)$ label	
F&p:	U→ F&p:	Submenu for force and pressure conversions, → 152.
G		Constants (→ 116ff).
G_0		
GAP	DISP ▲ GAP n	[5+] (0) Defines the interval for inserting narrow blank digit group separators in <i>DT 1, 2, 3, 4, 5, 8, and 9</i> (for <i>DT 10</i> , intervals are fixed: 4 digits for binary, 2 for bases 4, 8, and 16, and 3 for all other bases; separators in input will always be inserted like for <i>reals</i>). → OMS 2 and 3 . Actually, $n = 4$ formats the Chinese way and $n = 3$ for the rest of the world – except South Asia, which needs a more complex format (to be selected by SETIND). For $n < 3$, <u>no</u> group separators will be displayed in any number at all.

Item	Keystrokes	Remarks (→ 14ff for general information)
gal _{UK} →ℓ	V: ℓ ← gal _{UK}	[7,5] (1) {1, 2} → {2} Convert volumes (→ 152ff).
gal _{US} →ℓ	V: ℓ ← gal _{US}	
gal _{UK} →m ³	V: m ³ ← gal _{UK}	
gal _{US} →m ³	etc.	
GaussF	GaussF	[5] (0) Selects the <i>Gauss</i> peak fit model for L.R. Relevant also for CORR, $\hat{x}$, and $\hat{y}$ (→ 281ff).
G _C		Constant (→ 116ff).
GCD	GCD	[4] (2) {1; 10} Returns the <i>greatest common divisor</i> of x and y. This will always be positive.²⁰
g _d	g _d	[4] (1) {2, 3}; {1} → {2} Returns the <i>Gudermannian</i> function or its inverse, → 329.
g _d ⁻¹	g _d ⁻¹	
g _e		Constant (→ 116ff).
Geom _p	Geom: Geom _p etc.	[5] (1) {1, 2} → {2} <i>Geometric distribution</i>: The <i>CDF</i> returns the probability for a 1st success after x <i>Bernoulli</i> experiments. The probability p_0 for a success in each such experiment must be supplied in I. Geom⁻¹ returns the number of failures f before 1st success for given probability p in X, → 273ff.
Geom _Δ		
Geom _Δ		
Geom ⁻¹		
Geom:	Geom:	Submenu, → 146.
GM _⊕		Constant (→ 116ff).
GRAD	GRAD	[4] (0) Sets the ADM to <i>gradians</i> , a.k.a. <i>gon</i> .

²⁰ $\text{GCD}(x; y) = \left| x^y / \text{LCM}(x; y) \right|$. Compare *LCM*.

TN: *GCD* correspond à *PGCD* en français.
GCD entspricht *ggT* auf Deutsch.

Item	Keystrokes	Remarks (→ 14ff for general information)
GRAD→	GRAD→	[4] (1) {1, 2, 4} → {4} Converts <i>angles</i> as described on pp. 163f.
GRAMOD		Reserved <i>short integer variable</i> determining how the AGRAPH pattern (→ 21) is displayed: ²¹ 0: It is merged (OR-ed) with the existing display (<i>startup default</i>). 1: It overwrites all pixels in that part of the LCD. 2: Duplicate “on” pixels get turned “off”. 3: It is XOR-ed with the existing display.
GROW		System flag (→ 129ff)
GTO	<u>label</u>	[2+] (0) In RUM, positions the PP to <i>label</i> . In PEM, inserts an unconditional branch to <i>label</i> . Executed automatically, branches to <i>label</i> .
GTO.	. nnn	[3+,4] (0) ... local label <i>n</i> (specify ≤ 2 digits). Puts the PP to step <i>nnn</i> in <i>current program</i> – specify ≥ 3 digits until becoming definite.
	. <u>label</u>	... the <i>global label</i> specified (shall be picked from PROG).
	. .	... the end of used program memory, i.e. right to the final END, where you can start entering a new program.
GTO↑	▲	[3] (0) ... <u>previous END</u> , i.e. to the top of <i>current program</i> (→ OMS 4); if being there already, jumps to the top of previous program.
GTO↓	▼	<u>after</u> ... next END, i.e. to the top of next program.
g→crt	m: ▼ g→carat	[5] (1) {1, 2} → {2} Convert masses (→ 152ff).
g→oz.	... g→oz.	
g→troz	... g→troz oz.	

²¹ Working like *flags* 34 and 35 in *HP-42S*.

Item	Keystrokes	Remarks (→ 14ff for general information)
g		Constant (→ 116ff).
h		Constant (→ 116ff).
ha→ac	U→ A: ha→acre	[6] (1) {1, 2} → {2} Convert areas (→ 152ff).
ha→ac_{US}	... ha→acre_{US}	
ha→m²	... m²←ha	
HIDE	DISP ▲ HIDE	[5] (0) Hides all <i>real numbers</i> or parts with absolute values $< 10^{-n}$ with $n = IP(x)$, $12 \leq n \leq 99$ (i.e. displays '0.' instead of those). Useful, e.g., in matrices to suppress meaningless small elements. HIDE 00 resets, so everything allowed by RANGE will be displayed again; → RANGE on p. 72 and pp. 197ff.
HISTO		Reserved <i>matrix variable</i> for histograms.
HISTOG	FIT HISTOG	Submenu, → 147.
	STAT HISTOG	
HISTOX	... HISTOG HISTOX	[4] (0) Evaluates column 1 (or 2) of STATS and stores the results in HISTO .
HISTOY	... HISTOG HISTOY	
H_n	X.FN Orthog H_n	[4,5] (2) {1, 2} → {2} <i>Hermite polynomials</i> for probability (H_n) and physics (H_{np}), → 306.
H_{np}	... H_{np}	
HNORM	FIT HISTOG HPLT HNORM	Fits a normal (<i>Gaussian</i>) curve to your histogram plot.
HPLT	... HISTOG HPLT	Submenu for your histogram plot, → 147.
hp_E→W	U→ P: W←hp_E etc.	[4] (1) {1, 2} → {2} Convert powers (→ 152ff).
hp_M→W		
hp_{UK}→W		

Item	Keystrokes	Remarks (→ 14ff for general information)
Hyper _p	PROB  Hyper:	[5] (1) {1, 2} → {2} Hypergeometric distribution with the number of successes g in X. The probability of a success p₀ shall be in I, the sample size n in J, and the batch size n₀ in K (→ 273ff and OMS 2). Then Hyper⁻¹ returns the maximum number of successes m for a given probability p in X.
Hyper _Δ	Hyper_p etc.	
Hyper _Δ		
Hyper ⁻¹		
Hyper:	PROB  Hyper:	Submenu, → 146.
HypF	FIT  HypF	[5] (0) Selects the hyperbolic fit model for L.R. Relevant also for CORR , ŷ , and ŷ . → 281ff.
ħ		Constant (→ 116ff).
I	Reserved variable for register I .	
IDIV	INTS IDIV	[4] (2) {1, 10}; {2} → {1} Integer division, working like $\lfloor \frac{\text{I}}{\text{IP}} \rfloor$.²²
IDIVR	CAT. FCNS IDIVR	{1, 2, 10} Like IDIV but returns also the remainder in Y.²²
IGN1ER		System flag (→ 129ff)
iHg→Pa	U→ F&p: Pa ← in.Hg	[5] (1) {1, 2} → {2} Converts pressures, → 152ff.
Im	CPX Im	[4] (1) {2, 3} → {2}; {9} → {8} Returns the <i>imaginary</i> part of <i>x</i>. Compare Re.
	PART Im	
INC	LOOP INC r	[4+] (0) {1, 2, 10} Increments r by 1. Does not load L even for target address X.

²² → **OMS 3** for the *data types* of quotient and remainder.

Item	Keystrokes	Remarks (→ 14ff for general information)
INDEX	MATX INDEX <u>name</u>	[4+] (1) Indexes the matrix ²³ specified. A matrix shall be indexed before I+ , I- , J+ , J- , M.GET , M.PUT , M.R↔R , M.C↔C , RCLEL , STOEL , RCLIJ , or STOIJ can be used. You can also index a matrix by editing it (via M.EDIT or M.EDIN); after exiting the <i>Matrix Editor</i> , this matrix is no longer indexed. ²⁴
INFO	INFO	Menu, → 144.
INPUT	P.FN INPUT <i>r</i>	[3+] 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 new value for it. Pressing (R/S) then stores <i>x</i> into said destination and continues program execution. If you use an input <i>variable</i> undefined at execution time, INPUT creates this <i>variable</i> automatically with an initial value of zero. ATTENTION 1: While single-stepping through a routine, close input with ↵ instead of (R/S) . ATTENTION 2: After INPUT , Y may not contain what was in X before INPUT was called. ATTENTION 3: Take care of the <i>stack</i> when you calculate while INPUT is open – if your routine uses higher <i>stack</i> levels, your calculations may have side effects.
INTING		System flag (→ 129ff)
INTS	INTS	Menu, → 144.

²³ Your *WP43* allows for matrices in *variables* but also in *registers*. A named matrix was explicitly required on the *HP-42S* only.

²⁴ → also *Matrix Utility Functions* in the *HP-42S OM*, pp. 223ff.

Item	Keystrokes	Remarks (→ 14ff for general information)
INT?	TEST INT?	[4] (0) Tests x for being <i>integer</i>. Also <i>reals</i> with $FP(x) = 0$ are rated as <i>integer</i> here. Compare FP?. Example: For 5., both INT? and REAL? will return true. For 5, only INT? will return true.
INVRT	CAT. FCNS INVRT	Equals $[M]^{-1}$ (→ 112). Just for backward compatibility.
in.→m	U→ x: m ← inch	[5,6] (1) {1, 2} → {2}
in.→mm	... ▼ mm ← inch	Convert distances (→ 152ff).
IP	PART IP	[3] (1) {1, 8*}; {2, 10} → {1}
	# I for closed x	Returns the integer part of x . Compare FP.
ISE	LOOP ISE r	[3+] (0) {1, 2, 10} Given ccccc.ffffi in r, ISE increments r by ii, skipping next program step if $cccccc \geq fff$ then (→ rear side of your WP43 and DSE on p. 36). If r features no fractional part then fff is 0. If $ii = 0$, ccccc will be incremented by 1. ISE does not load L even for target address X. Note that neither fff nor ii can be negative, but ccccc can. In <i>RUM</i>, ISE returns true for $cccccc < fff$ and false for $cccccc \geq fff$.
ISG	LOOP ISG r	[3+] (0) {1, 2, 10} Works like ISE but skips if $cccccc > fff$.
ISM		Reserved <i>integer</i> variable for the <i>integer sign mode</i> of <i>short integers</i>: -1: sign and mantissa mode, 0: unsigned, 1: 1's complement, and 2: 2's complement (= SDC).

Item	Keystrokes	Remarks (→ 14ff for general information)
ISZ	LOOP ISZ r	[3+] (0) {1, 2, 10} Increments r by 1, skipping next program step if $-1 < s < +1$ thereafter. ISZ does not load L even for target address X . ²⁵
I _{xyz}	X.FN I _{xyz}	[4] (3) {1, 2} Returns the <i>regularized Beta function</i> ; → 329.
I+	MATX I+	[4] (0) Increments or decrements the row index i of the indexed matrix by 1. → INDEX and also J+, J-, RCLEL, STOEL, RCLIJ, and STOIJ.
I-	MATX I-	
I/O	I/O	Menu, → 144.
i%/a	FIN TVM i%/a	Reserved <i>real variable</i> for the annual interest rate of your investment or loan in TVM (→ OMS 6 and OMA 3).
i		Constant (→ 116ff).
J	Reserved <i>variable</i> for register J.	
jīn→kg	U→ m: kg ← jīn	[6] (1) {1, 2} → {2} Converts masses, → 152ff.
J _y (x)	X.FN J _y (x)	[4] (2) {1, 2} → {2} Returns the <i>Bessel function of first kind</i> and order y , → 329.
J+	MATX J+	[4] (0) Increments or decrements the column index j of the indexed matrix by 1. If GROW is set and i and j are pointing to the last element of the matrix, executing J+ creates a new row at the end of the matrix. → INDEX, I+, I-, RCLEL, STOEL, RCLIJ, and STOIJ.
J-	MATX J-	

²⁵ ISZ is known from HP-19C/29C, -67, and -16C.

Item	Keystrokes	Remarks (→ 14ff for general information)
J/G	CLK ▲ J/G	[5] (0) {2} Takes <i>x</i> specifying the date the <i>Gregorian</i> calendar became valid in the region you are interested in. This shall be entered as a <i>real number</i> reflecting the crucial <i>Gregorian</i> date, formatted according to the <i>date display mode</i> set (DMY, MDY, or YMD).
J→Btu	U→ E: J → Btu etc.	[4] (1) {1, 2} → {2} Convert energies (→ 152ff).
J→cal		
J→D	CLK J→D	[4] (1) {1} → {6} Takes <i>x</i> as a <i>Julian day number</i> ²⁶ and converts it to a common <i>date</i> according to J/G and the date format selected.
J→Wh	U→ E: J → Wh	[4] (1) {1, 2} → {2} Converts energies, → 152ff.
K	Reserved <i>variable</i> for register K .	
k		Constant (→ 116ff).
KEY		→ KEYG and KEYX.
KEYG	P.FN P.FN2 KEYG <i>key#</i> , <u>label</u>	[4++] Specifies the <i>label</i> to be branched to (KEYG) or called (KEYX) when a particular <i>softkey</i> is pressed. KEYG and KEYX work in <i>PEM</i> only and will be translated to a program step (on) KEY <i>key#</i> GT0 <i>label</i> or (on) KEY <i>key#</i> XEQ <i>label</i> , respectively. Key numbers go from 1 to 21 with 1 corresponding to (F1) , 9 to (F3) , and 14 to (F2) , for example. 19 - 21 represent ▲ , ▼ , and (EXIT) . → OMS 4.
KEYX	P.FN P.FN2 KEYX <i>key#</i> , <u>label</u>	

²⁶ Note a *Julian day number* differs from a *Julian date*. See corresponding articles in *Wikipedia* for more information about those counting numbers.

TN: *Julian day number* correspond à «jour Julien» en français.

Julian day number = „Julianische Tageszahl“ auf Deutsch.

Item	Keystrokes	Remarks (→ 14ff for general information)
KEY?	TEST KEY? <i>r</i>	[4+] (0) Tests if a key was pressed while a routine was running or paused. If <u>no</u> key was pressed then the next program step after KEY? will be executed; else it will be skipped and the code of said key will be stored in <i>r</i> . ²⁷ Key codes reflect the rows and columns on the keyboard (→ <i>OMS 4</i>). ²⁸ Compare PUTK.
kg→crt	U→ m: ▲ kg → carat	[5,4,6] (1) {1, 2} → {2} Convert masses (→ 152ff).
kg→cwt	... kg → cwt	
kg→jīn	... ▲ kg → jīn	
kg→lb.	... kg → lb.	
kg→liǎ	... ▲ kg → liǎng	
kg→oz.	... ▲ kg → oz.	
kg→scw	... kg → short cwt	
kg→sto	... kg → stone	
kg→sh.t	... ▲ kg → short ton	
kg→ton	... ▲ kg → ton	
kg→trz	... ▲ kg → troy oz.	
K _J		Constant (→ 116ff).
km→mi.	U→ x: ▼	[6] (1) {1, 2} → {2} Convert distances (→ 152ff).
km→nmi	km → mile etc.	
K(m)	X.FN Ellipt K(m)	[4] (1) {2, 3} Returns the <i>complete elliptic integral of the 1st kind</i> for <i>m</i> = <i>x</i> , → 327.

²⁷ This one and only exception of the rule “do if true” is taken over from *HP-42S*.

²⁸ → also GETKEY on *HP-42S*.

Item	Keystrokes	Remarks (→ 14ff for general information)
L		Reserved <i>variable</i> for register L .
(LASTx)	RCL L	[2] (-1) Recalls the last <i>x</i> used (→ OMS 1). Note this command will be recorded as RCL L in routines. Compare UNDO (→ 93).
lb	EXP lb <i>x</i>	[2] (1) {1, 2, 3, 8*, 9*, 10} ({1} → {2, 3}) Returns the logarithm of <i>x</i> for base 2.
lbft→Nm	U→ ▲ Nm ← lbft	[4] (1) {1, 2} → {2} Convert torques and forces (→ 152ff).
lbft→N	U→ F&p: N ← lbft	
LBL	LBL <i>label</i>	[2+] (0) Identifies programs and routines for executing and branching. → OMS 4 .
LBL?	TEST LBL? <i>label</i>	[4+] (0) Tests for existence of the <i>label</i> specified, anywhere in program memory. Compare LBL.
lb.→kg	U→ m: kg ← lb.	[4] (1) {1, 2} → {2} Converts masses, → 152ff.
LCM	INTS LCM	[4] (2) {1; 10} Returns the <i>Least Common Multiple</i> of <i>x</i> and <i>y</i> . ²⁹ This will always be positive.
LEAD.0	SF L	System flag (→ 129ff)
LEAP?	TEST LEAP?	[4] (0) {2, 6} Takes <i>x</i> as a <i>date</i> in the format selected (or a <i>real number</i> in corresponding format), extracts the year, and tests for a leap year.
lg	Ig	[2] (1) {1, 2, 3, 8*, 9*, 10} ({1} → {2, 3}) Returns the logarithm of <i>x</i> for base 10.

²⁹ $\text{LCM}(x; y) = \left| \frac{x \cdot y}{\text{GCD}(x; y)} \right|$. Compare *GCD*.

TN: LCM correspond à PPCM en français.

LCM entspricht kgV auf Deutsch.

Item	Keystrokes	Remarks (→ 14ff for general information)
LgNrm _p	PROB LgNrm: LgNrm _▲ LgNrm _▲ etc.	[5] (1) {1, 2} → {2} Log-normal distribution with $\mu = \ln \bar{x}_G$ specified in I and $\sigma = \ln \varepsilon$ in J (→ $\bar{x}_G$, ε , and pp. 273ff). LgNrm ⁻¹ returns x for a given probability p in X , with μ in I and σ in J .
LgNrm _▲		
LgNrm _▲		
LgNrm ⁻¹		
LgNrm:	PROB LgNrm:	Submenu, → 146.
li↔kg	U→ m: kg ← liäng	[6] (1) {1, 2} → {2} Converts masses, → 152ff.
LinF	FIT LinF	[4] (0) Selects the linear fit model for L.R. . Relevant also for CORR , $\hat{x}$, and $\hat{y}$. → 281ff.
lī→m	U→ x: m ← lī	[5] (1) {1, 2} → {2} Converts distances, → 152ff.
LJ	BIT LJ	[5] (1) {10} Left justifies the bit pattern x within its <i>word</i> size: The <i>stack</i> will lift, placing the left-justified <i>word</i> in Y and the count of bit-shifts necessary to justify the <i>word</i> in X . ³⁰ Example for <i>word</i> size 8: 1 0110 ₂ LJ returns $x = 3$ and $y = 1011\ 0000_2$
L _m	X.FN Orthog L _m etc.	[4] (2) {1, 2} → {2} ³⁰ <i>Laguerre</i> polynomials and <i>Laguerre's</i> generalized polynomials, → 290f.
L _{mα}		
ln	In	[1] (1) {2, 3, 8*, 9*, 10}; {1} → {2, 3} Returns the natural logarithm of x .
lnβ	X.FN lnβ	[4] (2) {2, 3}; {1} → {2} Returns the natural logarithm of <i>Euler's</i> Beta function (→ 101).

³⁰ LJ works as on *HP-16C*.

Item	Keystrokes	Remarks (→ 14ff for general information)
ln Γ	X.FN ln Γ	[4] (1) {2, 3}; {1} → {2}
	PROB  ln Γ	Returns the natural logarithm of $\Gamma(x)$ (→ 101). Allows also for calculating really great factorials (→ example in OMS 2).
ln(1+x)	EXP ln 1+x	[3] (1) {2, 8*} For $x \approx 0$, this function returns a more accurate result for the fractional part than $\ln(x)$ does.
LoadAll	I/O LoadAll	[3] Restores either the entire backup from the <i>FD SAVFILES/wp43.sav</i> whereto it was written by SAVE , or the entire <i>state file</i> selected (→ SAVEST on p. 81; pick the file from a <i>DMCP</i> menu on your calculator or a corresponding menu on your <i>PC</i>). LoadAll calls LoadPr , LoadEq , LoadR , LoadV , LoadΣ , LoadSS , and signals either Backup restored or State file restored . ³¹
LoadEq	I/O LoadEq	[3] (0) Loads all equations, ³² overwriting identical equations present in <i>RAM</i> , and signals Saved equations restored . ³¹
LoadPr	I/O LoadPr	[3] (0) Loads all programs. ³² Stores them behind the programs already in <i>RAM</i> and signals Saved programs appended . ³¹
LoadR	I/O LoadR	[4] Recalls the <i>registers</i> ³² (incl. the <i>stack</i> and <i>local registers</i>), and signals Saved global and local registers (with local flags) restored . ³¹ All <i>registers</i> saved in the file will be recalled.
LoadSS	I/O LoadSS	[4] (0) Recovers the system settings, ³² i.e. the entire calculator <i>configuration</i> (user assignments, <i>system variables</i> and <i>flags</i> covered by RESET , → 75 and 192) plus all global <i>user flags</i> , <u>MyMENU</u> , <u>MyPFN</u> , <u>Myα</u> , and all <i>user menus</i> . Signals Saved system settings restored . ³¹

³¹ → [App. B](#). If space in *RAM* does not suffice for restoring, an error will be thrown.

³² Source is here either the *backup file* or the *state file* selected (→ **LoadAll**).

Item	Keystrokes	Remarks (→ 14ff for general information)
LoadV	(I/O) LoadV	[4] (0) Recalls all user-defined <i>variables</i> ³² and signals Saved user variables restored .
LoadΣ	(I/O) LoadΣ	[4] (0) Recovers the statistics matrices STATS and HISTO as well as the <i>summation registers</i> ³² and signals Saved statistic data restored . Throws an error if no statistic data are allocated. ³¹
LocR	(P.FN) LocR <i>n</i>	[4+] (0) Allocates <i>n</i> <i>local registers</i> (≤ 99) and 32 <i>local flags</i> for the <i>current routine</i> . All start cleared. Any subsequent LocR in the same routine will set the amount of <i>local registers</i> to the new number. → OMS 4.
LogF	(FIT)  LogF	[4] (0) Selects the logarithmic fit model for L.R. Relevant also for CORR, $\hat{x}$, and $\hat{y}$, → 281ff.
Logis _p	(PROB)  Logis: Logis _p etc.	[5] (1) {1, 2} → {2} <i>Logistic distribution</i> with μ given in I and s in J (→ 273ff).
Logis _Δ		
Logis _Δ		
Logis ⁻¹		
Logis:	(PROB)  Logis:	Submenu, → 146.
log _x y	(EXP) log _x y	[2] (2) {1, 2, 3, 8*, 9*, 10}; ({1} → {2, 3}) Returns the logarithm of <i>y</i> for the base <i>x</i> .
LOOP	(LOOP)	Menu, → 145.
LOWBAT		System flag (→ 129ff)
l _{PL}		Constant (→ 116ff).
L.INTS	(CATALOG) VARS L.INTS	Submenu of all <i>long integer variables</i> defined at execution time, → 135ff.

Item	Keystrokes	Remarks (→ 14ff for general information)
L.R.	FIT L.R.	[3] (-2) or (-3) { } → {2} Pushes the parameters a_2 (in Z), a_1 (in Y), and a_0 (in X) of the fit curve through the data points accumulated in STATS on the <i>stack</i>, according to the fit model chosen (→ LinF, OrthoF, ExpF, LogF, PowerF, HypF, RootF, ParabF, CauchF, and GaussF). If BestF is set, L.R. picks the fit model returning maximum r^2 out of the models allowed (→ 25 and examples in OMS 2). For a straight line (in LinF or OrthoF), a_0 is its y-intercept and a_1 its slope. For forecasting, → $\hat{x}$ and $\hat{y}$ (note $\hat{x}$ may return 2 results). → 281ff for more.
l.y.→m	U→ x: m ← l.y.	[4,7] (1) {1, 2} → {2}
ℓ→gal _{UK}	U→ V: Δ ℓ → gal _{UK}	Convert distances and kitchen volumes (→ 152ff).
ℓ→gal _{US}	U→ V: ▽ ℓ → gal _{US}	
ℓ→qt.	U→ V: ▽ ℓ → quart	
m ² →ac	U→ A: m ² → acre	[5,6] (1) {1, 2} → {2}
m ² →ac _{US}	etc.	
m ² →ha		
m ² →m ²		
m ³ →bbl	U→ V: m ³ → barrel	[5,6] (1) {1, 2} → {2}
m ³ →fz _{UK}	... m ³ → floz _{UK}	
m ³ →fz _{US}	etc.	
m ³ →gl _{UK}	... m ³ → gal _{UK}	
m ³ →gl _{US}	etc.	

Item	Keystrokes	Remarks (→ 14ff for general information)
MAD	STAT ▲ MAD	[4] (-2) {} → {2} Pushes the <u>median absolute deviations</u> of the <i>y</i> - and <i>x</i> -data accumulated on the <i>stack</i> . ³³ $M_{AD}(x) := \text{median} x_i - \text{median}(x_i) $
MANT	PART MANT	[3] (1) {1, 2} → {2} Returns the mantissa <i>m</i> of $x = m \cdot 10^h$. Compare EXPT .
MASKL	BIT MASKL <i>n</i>	[3+] (-1) {} → {10} Create a bit mask. ³⁴ The mask length (or its address) follows the command. Thus, the mask is pushed on the <i>stack</i> . MASKL 0 and MASKR 0 return 0 .
MASKR	BIT MASKR <i>n</i>	
MATRS	CATALOG VARS MATRS	Submenu of all matrix variables defined at execution time, → 135f.
MATR?	TEST ▲ MATR?	[4] (0) Checks if <i>x</i> is a <i>real</i> or <i>complex</i> matrix.
MATX	MATX	Menu, → 145.
Mat.A	Reserved matrix variables for solving systems of linear equations (→ OMS 3).	
Mat.B		
Mat.X		
Mat_A	MATX M.SIMQ Mat A etc.	[4] (-1) Creates, if necessary, and opens the coefficient matrix Mat.A (or the constant vector Mat.B) for editing in a system of linear equations (→ example in OMS 3).
Mat_B		
Mat_X	MATX M.SIMQ Mat X	[4] (-1) Returns the solution vector Mat.X of a system of linear equations and exits M.SIMQ .
max	X.FN max	[4] (2) {1, 2, 4, 5, 6, 7, 10} Returns the maximum of <i>x</i> and <i>y</i> .

³³ Compare $s^2 = E([x_i - E(x_i)]^2)$.

³⁴ Works like **MASKL** or **MASKR** on **HP-16C**, but there the mask length is taken from **X**.

Item	Keystrokes	Remarks (→ 14ff for general information)
MDY		System flag (→ 129ff)
m _e		Constant (→ 116ff).
MEDIAN	STAT ▲ MEDIAN	[4] (-2) {} → {2} Returns the <i>medians</i> of the <i>y</i> - and <i>x</i> -data accumulated, pushing those on the <i>stack</i> . The <i>median</i> cuts the data set in equal halves.
MENU	P.FN P.FN2 MENU	Displays the programmable <i>menu</i> (→ OMS 4).
MENUS	CATALOG MENUS	Submenu of all <i>menus</i> defined at execution time, → 135ff.
MIL	MODE MIL	[3] (0) Sets the <i>ADM</i> to <i>mil</i> , → OMS 3.
MIL→	↔ MIL→	[4] (1) {1, 2, 4} → {4}
MIL→D	↔ ▲ MIL→D	Convert <i>angles</i> as described on pp. 131f.
MIL→R	↔ ▲ MIL→R	
min	X.FN min	[4] (2) {1, 2, 4, 5, 6, 7, 10} Returns the minimum of <i>x</i> and <i>y</i> .
MIRROR	BIT MIRROR	[4] (1) {10} Reflects the bit pattern in <i>x</i> (e.g., for <i>word</i> size 8, 0001 0111 ₂ MIRROR will return 1110 1000 ₂).
mi.→km	U→ x: ▼ km ← mile	[6,5] (1) {1, 2} → {2} Convert distances (→ 152ff).
mi.→m	U→ x: m ← mile	
mℓ→cp _I	U→ ■ V: ▲ ▲ mℓ → cup _I	[7,6] (1) {1, 2} → {2} Convert kitchen volumes (→ 152ff).
mℓ→cp _{UK}	... ▲ mℓ → cup _{UK}	
mℓ→cp _{US}	... ▼ mℓ → cup _{US}	
mℓ→fz _{UK}	... ▲ mℓ → floz _{UK}	
mℓ→fz _{US}	... ▼ mℓ → floz _{US}	

Item	Keystrokes	Remarks (→ 14ff for general information)
mℓ→pt _{UK}	... mℓ → pint _{UK}	
mℓ→pt _{US}	... mℓ → pint _{US}	
mℓ→tb _I	... mℓ → table spoon _I	
mℓ→tb _{UK}	... mℓ → tb _{sp} _{UK}	
mℓ→Tb _{US}	... mℓ → Tb _{sp} _{US}	
mℓ→ts _I	... mℓ → tea spoon _I	
mℓ→ts _{UK}	... mℓ → tsp _{UK}	
mℓ→ts _{US}	... mℓ → tsp _{US}	
mmHg→Pa	F&p: Pa mmHg	[5] (1) {1, 2} → {2}
mm→in.	x: mm → inch etc.	Convert pressures and heights (→ 152ff).
mm→pt.		
m _n		Constants (→ 116ff).
m _n /m _p		
MOD	MOD	[2] (2) {1, 2, 10}
	MOD	Returns $y \bmod x$. Compare RMD. → OMS 3 for examples
MODE	MODE	Menu, → 145.
MONTH	MONTH	[4] (1) {2, 6} → {1} Takes x as a <i>date</i> in the format selected (or a <i>real number</i> in corresponding format) and extracts the month.
m _p		Constants (→ 116ff).
m _p /m _e		
m _{pL}		
MSG	P.FN2 n	[5+] (0) {1, 2} Throws just the (<i>temporary</i>) <i>error message</i> specified by the error code n (→ 208ff). Compare ERR.

Item	Keystrokes	Remarks (→ 14ff for general information)
m_u		Constants (→ 116ff).
$m_u c^2$		
MULT×		System flag (→ 129ff)
MUL π	MODE MUL π	[3] (0) Sets the ADM to <i>multiples of π</i> .
MUL $\pi \rightarrow$	$\rightleftarrows$ MUL $\pi \rightarrow$	[4] (1) {1, 2, 4} → {4} Converts <i>angles</i> as described on pp. 163f.
$m\ddot{u} \rightarrow m^2$	U→ A: $m^2 \leftarrow m\ddot{u}$	[5] (1) {1, 2} → {2} Converts areas, → 152ff.
MVAR	P.FN MVAR <i>name</i>	[3+] (0) Defines a <i>menu variable</i> (such <i>variables</i> are required for VARMENU). Works in PEM and running programs only. → OMS 4.
MyMENU	CAT. MENUS MyMENU	Default user <i>menu</i> in RUM.
MyPFN	CAT. MENUS MyPFN	Default user <i>menu</i> in PEM. → OMS 7.
My α	CAT. MENUS My α	Default user <i>menu</i> in AIM.
m_p		Constant (→ 116ff).
M $\pi \rightarrow R$	$\rightleftarrows$ M $\pi \rightarrow R$	[4] (1) {1, 2, 4} → {4} Converts <i>angles</i> as described on pp. 163f.
M.CMAX	MATX CMAX	[5] (-2) {8} → {1 or 2 in X, 1 in Y} Finds the greatest (or smallest) element in column <i>j</i> of the <i>indexed matrix</i> (→ INDEX); pushes its row number <i>i</i> and its value a_{ij} on the <i>stack</i> . If two elements a_{ij} are equal, the one with the greater <i>i</i> will be returned. If you want to search more than one column (e.g., to find the smallest element of an entire matrix) you may re-dimension the matrix, search it, then re-dimension it back again (→ M.DIM).
M.CMIN	etc.	
M.C $\rightleftarrows$ C	MATX C $\rightleftarrows$ C	[5] (0) {1, 2} Swaps columns IP(<i>x</i>) and IP(<i>y</i>) of the <i>indexed matrix</i> .

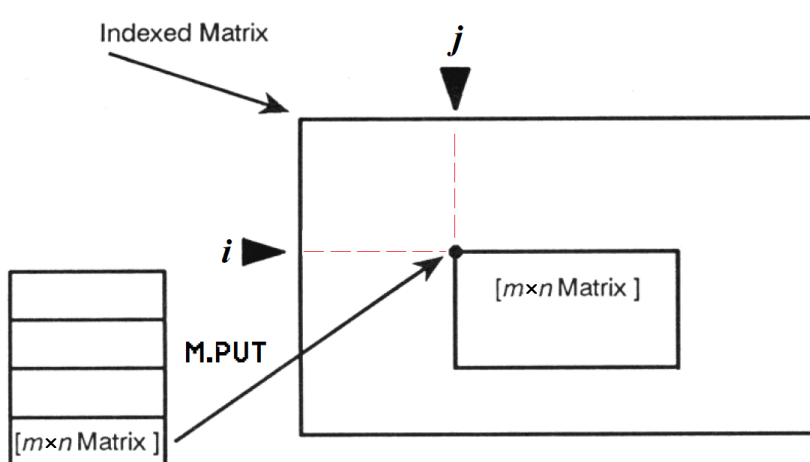
Item	Keystrokes	Remarks (→ 14ff for general information)
M.DELR	DEL R with M.EDIT displayed	[2] (0) {8, 9} Deletes the row of matrix elements where the cursor is. Will not work for a matrix with only 1 row.
M.DIM	MATX DIM <u>name</u>	[4+] (0) {1, 2} Creates a new matrix ³⁵ or re-dimensions an existing matrix to IP(y) rows and IP(x) columns (size limited by memory available).
M.DY	CLK ▲ M.DY	[4] (0) Selects the format mm/dd/yyyy for <i>dates</i> .
M.EDI	MATX EDIT	[3] {8, 9} Opens <i>x</i> using the <i>Matrix Editor</i> . ³⁶
M.EDIN	MATX EDITN <u>name</u>	[4+] (2) Opens a matrix ³⁵ using the <i>Matrix Editor</i> . ³⁶
M.EDIT		<i>Submenu</i> for matrix editing, called by M.EDI and M.EDIN, → 145.
M.FIND	MATX ▲ FIND	[5] (-1) {8, 9} Searches the <i>indexed matrix</i> for the value <i>x</i> . If it is found, <i>i</i> and <i>j</i> will point to the element (else they stay unchanged) and M.FIND will ... • return Yes (else No) in manual <i>RUM</i>; • execute the next step in a program running, else skip it.
M.GET	MATX GETM	[4] (0) {1, 2} → {8, 9} Gets a sub-matrix with IP(y) rows and IP(x) columns out of the <i>indexed matrix</i> , starting with its element <i>ij</i> , into X . See the picture and M.PUT.

³⁵ Your WP43 allows for matrices in *variables* but also in *registers*. A named matrix was explicitly required on the *HP-42S* only (→ DIM in the *HP-42S OM*, p. 217).

³⁶ → *OMS 3*. M.EDI (labeled EDIT in MATX) and M.EDIN (labeled EDITN) disable ASL.

In *HP-42S*, both do not actually disable ASL; they preserve the *stack lift* state – you can observe this if you execute ENTER vs. a *stack-lift*-enabling operation (e.g., $x \div y$) just before invoking them. This behavior is not really useful; it was dropped here since *HP-42S* compatibility is not required.

Item	Keystrokes	Remarks (→ 14ff for general information)
Indexed Matrix i j x columns y rows t t z $[.xxy Matrix]$ M.GET		
M.GOTO	GOTO	with M.EDIT displayed
M.GROW	GROW	
M.INSR	INSR	
M.LU	MATX M.LU	[2] (0) Asks for target row and column and goes to this matrix element, updating the index pointers i and j. [2] (0) Sets GROW to allow the <i>indexed matrix</i> to grow automatically (→ J+ and OMS 3). Compare M.WRAP. [2] (0) Inserts a new row of elements containing 0., above the current cursor position in the matrix. [5] (-2) {8, 9} Computes the <i>LU</i> (a.k.a. <i>LR</i>) <i>decomposition</i> of the matrix x and returns the pivot matrix in Z, the lower (left) triangular matrix L in Y, and the upper (right) triangular matrix U in X. L is computed such that all its diagonal elements are 1. Useful for matrix inversion. Principle:³⁷ $x = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 3 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & -2 \\ 0 & 0 & -2 \end{bmatrix} = L \cdot U$ You can press after M.LU to get the original matrix back or recall it from L.

Item	Keystrokes	Remarks (→ 14ff for general information)
M.NEW	MATX NEW	[3] (2) {1, 2} → {8} Creates a new matrix (size limited by memory available). Its number of rows is supplied as IP(y) and its number of columns as IP(x). M.NEW will return a matrix x with all elements set to zero.
M.PUT	MATX PUTM	[4] (0) {8, 9} Puts the matrix x as it is into the <i>indexed matrix</i> , starting with the target element a_{ij} . The size of the target matrix will not change. Compare M.GET .
		
M.QR	MATX QR	[5] (-1) {8, 9} Computes the <i>QR decomposition</i> of the matrix x and returns the unitary matrix Q in Y and the upper right triangular matrix R in X . Useful in the quest for eigenvalues and -vectors.
M.R↔R	MATX R↔R	[5] (0) {1, 2} Swaps rows IP(x) and IP(y) of the <i>indexed matrix</i> .
M.SIMQ		Submenu of MATX , called by SIM_EQ .
M.SQR?	TEST M.SQR?	[5] (0) Returns true if x is a square matrix.

37 → https://en.wikipedia.org/wiki/LU_decomposition (or equivalent articles in other languages), also about pivoting.

Item	Keystrokes	Remarks (→ 14ff for general information)
M.WRAP	WRAP	[2] (0) Controls the matrix index pointers (→ OMS 3). Clears GROW. Compare M.GROW . [1] (0) Browse the matrix x. Disable ASL.
M.→	→	
M.←	←	
M.↑	↑	
M.↓	↓	
with M.EDIT displayed		
m:	U→ m:	Submenu of mass conversions, → 147.
m→au	U→ x: m → au	[4,6,5] (1) {1, 2} → {2}
m→chĩ	U→ x: ▲ m → chĩ	Convert distances (→ 152ff).
m→cũn	etc.	
m→fēn		
m→ftm.	... ▲ m → fathm	
m→ft.	... m → foot	
m→in.	... m → inch	
m→ft_{US}	... ▼ m → surv. foot _{US}	
m→lĩ	... ▲ m → lĩ	
m→l.y.	... m → l.y.	
m→mi.	... m → mile etc.	
m→nmi		
m→pc	... m → pc	
m→pt.	... m → point	
m→yd.	... m → yard	
m→yĩn	... ▲ m → yĩn	
m→zhàn	... ▲ m → zhàng	

Item	Keystrokes	Remarks (→ 14ff for general information)
$M_{\odot}$		Constants (→ 116ff).
$M_{\oplus}$		
N_A		Constants (→ 116ff).
NaN		
NAND	[BIT] NAND	[4] (2) Works in analogy to AND (→ 22).
NaN?	[TEST] ▲ NaN?	[5] (0) Returns true if x is <i>Not a Number</i> . For complex x , $\text{Re}(x)$ or $\text{Im}(x)$ must be NaN..
nBINS	[FIT] HISTOG nBINS	[4] (1) Sets the number of bins for a histogram to be drawn.
	[STAT] HISTOG nBINS	
NBin _p	[PROB]  NBin: NBin _p etc.	[5] (1) {1, 2} → {2} Negative binomial distribution with the number of successes k in X , gross probability p_0 of a success in a single draw in I , and the number of failures n until the experiment is stopped in J . → 273ff.
NBin _▲		
NBin _▲		
NBin ⁻¹		
NBin:	[PROB]  NBin:	Submenu, → 146.
NEIGHB	[INFO] NEIGHB	[4] (2) {1} Returns • $x + 1$ for $x < y$, • x for $x = y$, and • $x - 1$ for $x > y$.
		(2) {2} Returns the nearest machine-representable number to x in the direction towards y in the mode set. For ... • $x < y$, it will be the machine successor of x; • $x = y$, it will be y; • $x > y$, it will be the machine predecessor of x. It may be useful investigating numeric stability. ³⁸

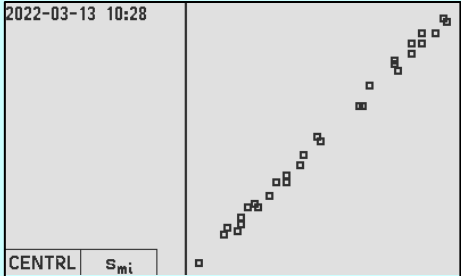
Item	Keystrokes	Remarks (→ 14ff for general information)
NEXTP	X.FN NEXTP	[4] (1) {1, 10}; {2} → {1} Returns the next prime number greater than $IP(x)$. Compare PRIME? on p. 71. Searching for the next prime number may take long and thus can be aborted by EXIT. ATTENTION: → 197 for limitations.
nmi.→km	U→ x: ▼ km ← nmi.	[6,5,4] (1) {1, 2} → {2} Convert distances and torques (→ 152ff).
nmi.→m	U→ x: m ← nmi.	
Nm→lbf·ft	U→ ▲ Nm → lbf·ft	
NOP	P.FN P.FN2 NOP	'Empty' program step (for historical reasons only).
NOR	BIT NOR	[4] (2) Works in analogy to AND (→ 22).
Norml _p	PROB Norml: Norml _p etc.	[4] (1) {1, 2} → {2} Normal distribution with an arbitrary mean μ given in I and an SD σ in J (→ 273ff and OMS 2). Note you get the <u>standard</u> normal (<i>Gaussian</i>) distribution for $\mu = 0$ and $\sigma = 1$. Norml⁻¹ returns x for a given probability p in X, with μ in I and σ in J.³⁹
Norml _▲		
Norml _▲		
Norml ⁻¹		
Norml:	PROB Norml:	Submenu, → 146.
NOT	BIT NOT	[3] (1) {10} Inverts x bit-wise (as on HP-16C). (1) {1, 2} → {1} Returns 1 for $x = 0$, and 0 for $x \neq 0$.
nPER	FIN TVM nPER	Reserved <i>integer variable</i> for the <u>total</u> number of • payment periods for your loan or • compounding periods for your investment. → OMS 6 and OMA 3.

³⁸ **NEIGHB** here corresponds to NEIGHBOR in the **HP-71 Math Pac**.

³⁹ With $i = 0$ and $j = 1$, Norml⁻¹ equals $-Zp$ on **HP-21S**.

Item	Keystrokes	Remarks (→ 14ff for general information)
NUM.IN		System flag (→ 129ff).
NXTFIT	FIT ASSESS NXTFIT	[5] (0) Plots the next fit model allowed together with the accumulated data points (→ 25 for BestF).
nΣ	Σ n	[3] (-1) {} → {1} Recalls the number of accumulated data points.
N→lbf	U→ F&p: N → lbf	[4] (1) {1, 2} → {2} Converts forces, → 152ff.
ODD?	TEST ODD?	[4] (0) Checks if x is integer and odd (→ INT? on p. 49). Note <i>reals</i> with $FP(x) = 0$ count as integers as well.
OFF	OFF	[2] (0) Turns your WP43 off. In PEM, inserts a step to turn it off under program control. Note a timer running in background will continue!
OR	BIT OR	[3] (2) Works in analogy to AND (→ 22).
OrthoF	FIT OrthoF	[4] (0) Selects the linear fit model for orthogonal regression. Relevant for CENTRL, CORR, L.R., s_{mi} , $\hat{x}$, and $\hat{y}$. → 281ff.
ORTHOG	X.FN Orthog	Submenu for orthogonal polynomials, → 148.
OVERFL		System flag (→ 129ff), set by SF B (big).
oz.→g	U→ m: kg ← oz. etc.	[4] (1) {1, 2} → {2} Convert masses (→ 152ff).
oz.→kg		
PAUSE	P.FN PAUSE <u>n</u>	[3+] (0) Within a routine running, refreshes the LCD and pauses program execution for n ticks (→ TICKS on p. 90), with $0 \leq n \leq 99$. This pause will terminate early when you press a key (→ also KEY?).

Item	Keystrokes	Remarks (→ 14ff for general information)
Pa→atm	(U→) F&p: Pa → atm	[5,4] (1) {1, 2} → {2} Convert pressures and distances (→ 152ff).
Pa→bar	... Pa → bar	
Pa→iHg	... Pa → in.Hg	
Pa→mmHg	... Pa → mmHg	
Pa→psi	... Pa → psi	
Pa→torr	... Pa → torr	
pc→m	(U→) x: m ← pc	
PERM	(PROB) P _y x	[3] (2) {1} Returns the number of possible <u>arrangements</u> (a.k.a. <i>permutations</i>) of <i>x</i> objects taken out of a set of <i>y</i> objects. No <i>object</i> shall occur more than once in an arrangement, and <u>different orders</u> of the same <i>x</i> objects <u>are counted</u> separately. Compare COMB. (2) {2, 3} → 271f for the formula.
PER/a	(FIN) TVM per/a	Reserved <i>integer variable</i> for the <u>annual</u> number of • payments for your loan or • compounding periods of your investment. → OMS 6 and OMA 3.
PGMINT	(ADV) PGMINT <u>label</u>	[4+] (0) Specifies the address of the routine containing the expression to be integrated or solved, respectively (→ OMS 5).
PGMSLV	(ADV) PGMSLV <u>label</u>	
PIXEL	(P.FN) P.FN2 PIXEL	[5] (0) Turns on a single pixel (dot) on the <i>LCD</i> . The location of the pixel is given by IP(<i>x</i>) and IP(<i>y</i>). Compare AGRAPH on p. 21.
PLAY	(I/O) PLAY <u>s</u>	[4+] (0) Plays a <i>n</i> ×2 or <i>n</i> ×3 matrix <i>s</i> of buzzer tones: each row contains one tone with its frequency in Hz, duration in ms, and (optionally) the volume level (1...11).

Item	Keystrokes	Remarks (→ 14ff for general information)
PLOT	[FIT] [] PLOT	[4] (0) Opens a new screen and plots the data points stored in STATS, typically for an MSA (→ 298ff and CENTRL on p. 28). Find examples on pp. 299ff and in OMS 2. PLOT can be used for other purposes as well – compare OMA 4. 
PMT	[FIN] TVM PMT	Reserved <i>real variable</i> for the payment per period of your investment or loan in TVM (→ OMS 6 and OMA 3).
P _n	[X.FN] Orthog P_n	[4] (1) {1, 2} → {2} <i>Legendre polynomials</i>, → 290f.
POINT	[P.FN] P.FN2 POINT	[5] (1) {1, 2} Turns on a square point (3×3 <i>pixels</i>) on the LCD. The position of its center is given by IP(x) and IP(y). Compare AGRAPH on p. 21.
Poiss _p	[PROB] [] Poiss:	[5] (1) {1, 2} → {2} <i>Poisson distribution</i> with the number of successes g in X and the <i>Poisson</i> parameter λ in I (→ 273ff). Poiss⁻¹ returns the maximum number of successes m for given probability p in X and λ in I.
Poiss _▲	Poiss_p etc.	
Poiss _▲		
Poiss ⁻¹		
Poiss:	[PROB] [] Poiss:	<i>Submenu</i> , → 146.
POLAR		<i>System flag</i> (→ 129ff), set by [SF] [X] .
PopLR	[P.FN] PopLR	[4] (0) Pops the <i>local registers</i> allocated to the <i>current routine</i> (→ OMS 4) <u>without returning to the caller</u>. → LocR (on p. 56) and RTN (on p. 79).

Item	Keystrokes	Remarks (→ 14ff for general information)
PowerF	(FIT) ▼ PowerF	[4] (0) Selects the power curve fit model for L.R. Relevant also for CORR , $\hat{x}$, and $\hat{y}$ (→ 281ff).
PRIME?	(TEST) PRIME?	[4] (0) {1, 2, 10} Checks if $ IP(x) $ is a prime number. Returns true for prime and false for composite. ATTENTION: → 197 for limitations.
PROB	(PROB)	Menu, → 146.
PROG		Submenu of global labels defined when calling XEQ etc., → OMS 4.
PROGS	(CATALOG) PROGS	Submenu of global labels defined at execution time, → 135f.
PROMPT	(P.FN) PROMPT s	[4+] (0) Displays the name of the source and its content <i>s</i> left-adjusted in the top numeric row (and below if it needs more space) and stops program execution, lighting ⓘ , and waiting for your input. Compare AVIEW .
PROPF		System flag (→ 129ff)
pr→dB	(U→) ▲ dB ← power ratio	[4,5,7] (1) {1, 2} → {2} Convert ratios, pressures, print heights, and kitchen volumes (→ 152ff).
psi→Pa	(U→) F&p: Pa ← psi	
pt.→m	(U→) x: m ← point etc.	
pt.→mm		
pt_{UK}→mℓ	(U→) ■ V: ▲ mℓ ← pint_{UK}	
pt_{US}→mℓ	(U→) ■ V: ▼ mℓ ← pint_{US}	
PUTK	(P.FN) PUTK s	[4+] (0) Assumes a key code in <i>s</i> . 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. ⁴⁰ May help in user interaction with routines (→ OMS 4).

Item	Keystrokes	Remarks (→ 14ff for general information)
PV	(FIN) TVM PV	Reserved <i>real variable</i> for the present value of your investment or loan in <i>TVM</i> (→ OMS 6 and OMA 3).
P.FN	(P.FN)	Menu of functions for programming, → 146.
P.FN2	(P.FN) P.FN2	Submenu, → 146. Hence, P.FN is no multi-view menu.
P:	(U→) P:	Submenu of power conversions → 152.
Q ₃ -Q ₁	(STAT) (▲) Q₃-Q₁	[4] (-2) {} → {2} Pushes the distance between the 75%- and 25%-percentile (a.k.a. 3 rd and 1 st quartile) of the <i>y</i> - and <i>x</i> -data accumulated on the <i>stack</i> . Note this may only make sense for data sets containing at least 12 samples.
QUIET		System flag (→ 129ff)
qt.→ℓ	(U→) () V: (▲) (▲) ℓ ← quart	[8] (1) {1, 2} → {2} Converts volumes, → 152ff.
R		Constant (→ 116ff).
(r)		→ CORR on p. 31.
RAD	(MODE) RAD	[3] (0) Sets the <i>ADM</i> to <i>radians</i> .
RAD→	(↔) RAD→	[4] (1) {1, 2, 4} → {4} Converts <i>angles</i> as described on pp. 163f.
RANGE	(DISP) (▲) RANGE	[5] (0) Limits the range of <i>real numbers</i> or parts to be displayed or entered to $\pm 10^{\pm n}$ with $n = \text{IP}(x)$, $99 \leq n \leq 6145$. For greater <i>x</i> , <i>n</i> will be set to 6145; for less it will be set to 99. <i>Startup default</i> is 999 . In consequence, '0.' will be displayed for smaller values and '∞' for greater, saving screen space for sensible data. Compare HIDE (p. 46), also → 197ff.

⁴⁰ PUTK was invented for the **WP 34S**.

Item	Keystrokes	Remarks (→ 14ff for general information)
RANI#	PROB  RANI#	[4] (-1) {} → {1} Returns a pseudo random <i>integer</i> n with $IP[\min(x, y)] \leq n \leq IP[\max(x, y)]$. ⁴¹ After executing RANI#, the <i>stack</i> will look like [$n, x, y, \dots$], so you can roll down or drop n and call RANI# again to get another random <i>integer</i> within the same limits. Use RANI# , e.g., for throwing dice.
RAN#	PROB  RAN#	[4] (-1) {} → {2} Returns a pseudo random <i>real</i> $x \in [0; 1)$. ⁴¹ → also SEED (on p. 82) and p. 343.
RBR	RBR	[2] Calls the <i>register browser</i> (→ OMS 6). RBR works also in PEM but it is not programmable. In RBR, numbers are generally displayed in their format chosen. Since it uses the small font always, more digits can be shown than in a standard numeric row. For <i>reals</i> and ALL 0, RBR display will turn to SCI or ENG at 33 digits in worst case. Extended display precision may be observed for <i>complex numbers</i> as well.
RCL	RCL <u>s</u>	[1+] (-1) Recalls (copies) the content of the RoV specified and pushes it on the <i>stack</i> .
RCLCFG	RCL Config <u>s</u>	[3+] (0) Recalls a calculator <i>configuration</i> stored by STOCFG (→ OMS 2 and 7) from the RoV specified. → also RESET.

⁴¹ For equal seeds, you will get equal sequences of 'random' numbers. But → SEED.

Note you can easily produce pseudo random numbers following a *normal distribution* with mean 5 and SD 1.826 by adding ten returns of RAN# for each such number. This is brute force. More elegant may be using the *Box-Muller transform* (→ https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform) or the *Marsaglia polar method* (https://en.wikipedia.org/wiki/Marsaglia_polar_method).

ATTENTION: UNDOing RAN# or RANI# will return previous *stack* but not reset current seed.

Item	Keystrokes	Remarks (→ 14ff for general information)
RCLEL	MATX RCLEL	[4,3,1] (-1) {} → {2, 3}
	RCL ...EL	Recalls a copy of the current element a_{ij} of the <i>indexed matrix</i> . Compare STOEL .
	OLD with M.EDIT displayed	
RCLIJ	MATX  RCLIJ	[4,3] (-2) {} → {1}
	RCL ...IJ	Recalls the current values of the matrix index pointers into X (= column number) and Y (= row number). If no matrix is indexed, an error is thrown. Compare STOIJ .
RCLMAX	RCL max <u>s</u>	[2+] (1) {1, 2, 4, 5, 6, 10}
	RCL  <u>s</u>	Replaces x with the maximum of s and x . ⁴²
RCLMIN	RCL min <u>s</u>	[2+] (1) {1, 2, 4, 5, 6, 10}
	RCL  <u>s</u>	Replaces x with the minimum of s and x . ⁴²
RCLSTK	RCL Stack <u>s</u>	[3+] Recalls 4 or 8 values from a set of <i>registers</i> starting at address s , and pushes them on the <i>stack</i> . This is the inverse command of STOSTK .
RCL+	RCL + <u>s</u>	[2+] (1) Recalls the content of the <i>RoV</i> addressed, executes the operation specified, and puts the result on the <i>stack</i> like a <i>monadic</i> function. ⁴² Example: RCL- 12 replaces x with $x - r12$.
RCL-	RCL - <u>s</u>	
RCL×	RCL x <u>s</u>	
RCL/	RCL / <u>s</u>	
RDP	DISP RDP <u>n</u>	[4+] (1) {2, 3, 4, 5, 8*, 9*} Rounds x to n decimal places ($0 \leq n \leq 99$, think of FIX format), taking the RM setting into account. → RM and compare RSD .

⁴² Only legal operations according to the *DT* matrices in **OMS 3** will work. See also the examples given there.

Item	Keystrokes	Remarks (→ 14ff for general information)
		Examples: 123.456 789 012 RDP 5 returns 123.456 79 (5 <u>decimals</u> survive). 1.234 56 $\times 10^{-50}$ RDP 53 returns 1.235 $\times 10^{-50}$ (53 <u>decimals</u> survive).
Re	CPX Re	[3] (1) {2, 3} → {2}; {9} → {8} Returns the <i>real</i> part of <i>x</i> . Compare Im .
	PART Re	
r _e		Constant (→ 116ff).
READP	I/O READP <i>name</i>	[4+] (0) Reads a named program file from the <i>FD</i> and puts it in program memory (if a program carrying an identical <i>name</i> is found there, it will be overwritten). → WRITEP (p. 95) and OMS 4 .
REALDF		Reserved <i>integer variable</i> for <i>real number display format</i> : 0: ALL, 1: FIX, 2: SCI, 3: ENG.
REALS	CATALOG VARS REALS	<i>Submenu</i> of all <i>real variables</i> defined at execution time, → 135f.
REAL?	TEST ▲ REAL?	[4] (0) Checks if <i>x</i> is of <i>DT 2</i> or <i>8</i> (compare INT? on p. 49).
REDRAW	EQN Solver reDraw	[4] (0) Redraws the graph for the current equation as it was looking immediately after DRAW (→ 36), undoing previous ZOOMS this way.
REM	P.FN REM	[4] In <i>PEM</i> , inserts a step containing a comment. Each comment may take up to 4 lines or 196 characters, whatever is reached first. In a running program, this step is skipped. Does nothing in <i>RUM</i> .
RESET	CLR ▲ RESET	[4] Executes CLALL and resets your <i>WP43</i> to <i>SDC</i> , i.e. to 2COMPL , ALL 0 , DEG , DENMAX 0 , DSTACK 4 , GAP 3 , J/G 1752.0914 , LinF , LocR 0 , RM 0 , RSD 34 , TDISP 0 , and WSIZE 64 . It also sets RANGE to 999 , and sets ASLIFT , DECIM. , DENANY , ENDPMT , MULTX , PROPFR , SSIZE8 ,

Item	Keystrokes	Remarks (→ 14ff for general information)
		TDM24, and YMD; it clears all other <i>system flags</i> and resets the random number generator. → said commands, <i>system flags</i> , and p. 192.
(RESET)	in TIMER	→ T.RES.
RE→CX	CC (works in RUM only, does nothing in PEM)	[1,4] (2) {2} → {3} Composes a <i>complex number</i> out of two <i>reals</i> or <i>integers</i> x and y , setting $\mathbb{C}$ and taking either... • (for $\underline{\mathbb{L}}$) the <i>real</i> part from $\mathbf{Y}$ and <i>imaginary</i> part from $\mathbf{X}$, or • (for $\odot$) magnitude from $\mathbf{Y}$ and phase from $\mathbf{X}$.
	CPX RE→CX	(2) {8} → {9} Works in analogy for two <i>real</i> matrices x and y .
Re↔Im	CPX Re↔Im	[3] (1) {3, 9*} Swaps <i>real</i> and <i>imaginary</i> parts of a <i>complex</i> x .
RJ	BIT  RJ	[5] (1) {10} Right justifies the bit pattern x within its word size, in analogy to LJ (→ 54). The <i>stack</i> will lift, placing the right-justified <i>word</i> in $\mathbf{Y}$ and the count of bit-shifts needed to justify the <i>word</i> in $\mathbf{X}$. Example: 10 1100 ₂ RJ results in $y = 1011_2$ and $x = 2$.
R _k		Constant (→ 116ff).
RL	BIT  RL <u>n</u>	[5+] (1) {10}  Rotates the bit pattern in $\mathbf{X}$ left n (≤ 63) bits. ⁴³
RLC	BIT  RLC <u>n</u>	[5+] (1) {10}  Rotates the bit pattern in $\mathbf{X}$ left n (≤ 63) bits through CARRY. ⁴³

⁴³ RL 0 and RLC 0 execute as NOP, but load L. → OMS 3.

RL and RLC work like RL n and RLC n on the HP-16C, respectively.

Item	Keystrokes	Remarks (→ 14ff for general information)
RM	(MODE) RM n	[4+] (0) Sets floating point rounding mode. This mode is used only for RSD or when converting from the extended precision internal format (typically 39 digits) to packed <i>reals</i>. It will <u>not</u> alter the display nor change the behavior of ROUND. The following seven modes are supported: 0: round half even: $\frac{1}{2}\mathbf{E}$ 0.5 rounds to next even number (default, used in science). 1: round half up: $\frac{1}{2}\mathbf{\uparrow}$ 0.5 rounds up ('business-man's rounding' ⁴⁴). 2: round half down: $\frac{1}{2}\mathbf{\downarrow}$ 0.5 rounds down. 3: round up: $\mathbf{\leftarrow 0}$ rounds away from 0. 4: round down: $\mathbf{\rightarrow 0 \leftarrow}$ rounds towards 0. 5: ceiling: $\mathbf{\lceil x \rceil}$ rounds towards $+\infty$. 6: floor: $\mathbf{\lfloor x \rfloor}$ rounds towards $-\infty$. The abbreviations printed on grey here are used in STATUS for indicating the respective rounding mode (→ OMS 6).
RMD	(INTS) RMD	[4] (2) {1, 2, 10} Returns the remainder of the division y / x (→ OMS 3 for examples).⁴⁵ Compare MOD.
R_{Moon}		Constant (→ 116ff).
RNORM	(MATX) ( RNORM	[5] (1) {8, 9} → {2} 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.⁴⁶ For a vector, the row norm is the largest absolute value of any of its elements. Compare ENORM.

⁴⁴ **TN:** Cela correspond à l'arrondi commercial.
Entspricht kaufmännischer Rundung.

⁴⁵ Corresponds to RMD on **HP-16C** but works for *reals* on your **WP43** as well.

⁴⁶ This works like RNRN on **HP-42S**.

Item	Keystrokes	Remarks (→ 14ff for general information)
RootF	FIT RootF	[5] (0) Selects the root fit model for L.R. ; relevant also for CORR , $\hat{x}$, and $\hat{y}$. → 281ff.
ROUND	DISP ROUND	[3] (1) {2, 3, 4, 5, 6, 8*, 9*} Rounds x using the current display format; i.e. all digits not currently displayed will be lost for further calculations, just the rounded value survives. ⁴⁷
ROUNDI	DISP ROUNDI	[3] (1) {1, 8*}; {2} → {1} Rounds x to next <i>integer</i> . $\frac{1}{2}$ rounds to 1. Compare IP.
RR	BIT RR <u>n</u>	[5+] (1) {10}  Rotates the bit pattern in X right n (≤ 63) bits. ⁴⁸
RRC	BIT RRC <u>n</u>	[5+] (1) {10}  Rotates the bit pattern in X right n (≤ 63) bits through CARRY. ⁴⁸
RSD	DISP RSD <u>n</u>	[4+] (1) {2, 3, 4, 5, 8*, 9*} Rounds x to n significant digits ($1 \leq n \leq 34$), taking the RM setting into account. Think of SCI . → RM and RDP . Example: 123.456 789 012 RSD 7 returns 123.456 8 (7 <u>digits</u> survive).
RSUM	MATX RSUM	[5] (1) {8, 9} Calculates the row sum of the matrix x , returning an $m \times 1$ matrix filled with the signed row sums of the $m \times n$ input matrix.

⁴⁷ This works like RND on [HP-42S](#).

⁴⁸ **RR 0** and **RRC 0** execute as **NOP**, but load **L**. → [OMS 3](#).

RR and **RRC** work like **RRn** or **RRCn** on [HP-16C](#), respectively.

Item	Keystrokes	Remarks (→ 14ff for general information)
RTN	RTN	[2] (0) In <i>PEM</i>, RTN shall be the logically last command in a routine (→ <i>OMS 4</i>). Executed automatically (e.g., in a routine Y), RTN pops <i>local</i> data (→ POPLR) and returns to the caller, moving the <i>PP</i> to the step just behind XEQ 'Y'. If there is no caller (i.e. the routine is top level), program execution will halt and the <i>PP</i> will be set to step 0000. If pressed in <i>RUM</i> with no routine executing, RTN resets the <i>PP</i> to the first step of the <i>current program</i> (→ <i>OMS 4</i>).
RTN.SKIP	P.FN2 RTN.SKIP	[4] (0) Works like RTN but in a routine executing, RTN.SKIP moves the <i>PP</i> <u>two</u> steps behind the XEQ that called said routine.
RUNIO		System flags (→ 129ff)
RUNTIM		
R-CLR	P.FN R-CLR	[4] (0) {2} Interprets <i>x</i> in the form <i>sss.nn</i>. Clears <i>nn</i> registers starting with address <i>sss</i>. Example: For <i>x</i> = 34.56, R-CLR will clear R34 - R89. ATTENTION: For <i>nn</i> = 0, clearing will cover the maximum available: - For <i>sss</i> ∈ [0; 99], it will stop at R99. - For <i>sss</i> ∈ [100; 111], it will stop at K. - For <i>sss</i> ≥ 112, it will stop at the highest local <i>register</i> currently allocated.
R-COPY	P.FN R-COPY	[4] (0) {2} Interprets <i>x</i> in the form <i>sss.nnddd</i>. Takes <i>nn</i> registers starting with address <i>sss</i> and copies their contents to <i>ddd</i> etc. Example: For <i>x</i> = 7.03045, R-COPY will copy r07, r08, and r09 into R45, R46, and R47, respectively.

Item	Keystrokes	Remarks (→ 14ff for general information)
		For $x < 0$, R-COPY will take <i>nn registers</i> from the backup file instead, starting with <i>register</i> number sss. Destination will be in <i>RAM</i> always. ATTENTION: For $nn = 0$, copying will cover the maximum available as explained with R-CLR. Then x must be negative.
R-SORT	P.FN R-SORT	[4] (0) {2} Interprets x in the form <i>sss.nn</i>. Sorts the contents of <i>nn registers</i> starting with address sss. Example: Assume $x = 49.03$, $r49 = 1.2$, $r50 = -3.4$, and $r51 = 0$; then R-SORT will return $r49 = -3.4$, $r50 = 0$, and $r51 = 1.2$. ATTENTION: For $nn = 0$, sorting will cover the maximum available as explained with R-CLR.
R-SWAP	P.FN R-SWAP	[4] (0) {2} Works like R-COPY but <u>swaps</u> the contents of corresponding source and destination <i>registers</i>.⁴⁹
R→D	 R→D	[4] (1) {1, 2, 4} → {4} Convert <i>angles</i> as described on 163f.
R→MIL	  R→MIL	
R→Mπ	 R→Mπ	
R↑		[2,1] Rotates the <i>stack</i> contents one level up or down, respectively (→ <i>OMS 1</i>). → also $x \rightleftharpoons y$ for swapping arguments (→ 98).
R↓		
R_∞		Constants (→ 116ff).
R_⊙		
R_⊕		

⁴⁹ Corresponds to the historic command REGSWAP.

Item	Keystrokes	Remarks (→ 14ff for general information)
s	STAT s	[3] (-2) { } → {2} Takes the statistical data accumulated and returns the <i>sample SD</i> s_y and s_x (→ OMS 2 for the output format and p. 286 for the formula).
Sa		Constant (→ 116ff).
SAVE	SAVE	[2] (0) Saves your <i>WP43 state</i> (program space, <i>RoV's</i> , <i>system flags</i> , user assignments, and <i>configuration</i> , → OMS 4) in <code>SAVEFILES/wp43.sav</code> (the backup file) on the <i>FD</i> and returns Saved . Recall your backup or parts of it using the different flavors of LOAD (→ 55).
SAVEST	I/O SAVEST	[3] (0) Works like SAVE but saves your <i>WP43 state</i> in a <u>named</u> file <code>STATE/name.s43</code> . On your calculator, you will be prompted by <i>DMCP</i> for the file name (choose either <New File> ⁵⁰ or an existing file to overwrite) – on your <i>PC</i> , a corresponding prompt will appear. Recall this entire <i>state</i> or parts of it using the different flavors of LOAD (→ 55)..
SB	BIT SB n	[4+] (1) {10} Sets the specified bit in <i>x</i> .
Sb		Constant (→ 116ff).
SCI	DISP SCI n	[3+] (0) Sets scientific display format (→ OMS 2); $0 \leq n \leq 15$ is the number of decimals of the mantissa (ideally, n should be the number of significant digits minus 1).
scw→kg	U→ m: kg ← short cwt	[5] (1) {1, 2} → {2} Converts masses, → 152ff.

⁵⁰ You will be in a kind of *AIM*, set to lower case. Use  to toggle upper case or digits.

Item	Keystrokes	Remarks (→ 14ff for general information)
SDL	P.FN P.FN2	[5+] (1) {1, 2}
SDR	SDL <i>n</i> etc.	Shifts digits left (right) by <i>n</i> decimal positions, equivalent to multiplying (dividing) <i>x</i> by 10^n . SDR for <i>long integers</i> will end with zero. Compare ASR , SL , and SR for binary <i>integers</i> .
Se ²		Constant (→ 116ff).
SEED	PROB SEED	[4] (0) {2} Stores a seed ($0 < x \leq 1$) for RAN# . For $x \leq 0$, the seed is taken from the real-time clock; this is a way to generate <u>more random</u> numbers (→ RAN#).
SETCHN	DISP CHINA	[4] (0) Sets regional format preferences. ⁵¹
SETDAT	CLK SETDAT	[4] (0) {2, 6} Sets the date for the real-time clock. ⁵²
SETEUR	DISP EUROPE	[4] (0) Set regional format preferences. ⁵¹
SETIND	DISP INDIA	
SETJPN	DISP JAPAN	
SETSIG	MODE SETSIG	[4] (0) {1} Sets the number of significant digits (1 ... 34) for rounding after <u>each</u> operation. SETSIG 0 sets maximum precision. SETSIG is also relevant for the precision of the <i>Solver</i> , → 86.
SETTIM	CLK SETTIM	[4] (0) {2, 5} Sets the time for the real-time clock. Input may be a <i>real number</i> formatted <i>hh.mm.ss</i> or a <i>time</i> . ⁵²
SETUK	DISP UK	[4] (0) Set regional format preferences. ⁵¹
SETUSA	etc.	
Se' ²		Constant (→ 116ff).

⁵¹ → **OMS 2** about localizing numeric output.

⁵² The *Simulator* reads this information from the *PC* clock.

Item	Keystrokes	Remarks (→ 14ff for general information)
SF	[SF] <u>n</u>	[2+] (0) Sets the <i>flag</i> specified.
	[FLAG] SF <u>n</u>	
	[MODE] SF <u>n</u>	
Sf ⁻¹		Constant (→ 116ff).
SHOW	[SHOW]	[2] (0) {1, 2, 3, 4, 7} Shows all <i>characters</i> of <i>x</i> in top numeric row until next keystroke, using small font. For a <i>real number</i> up to 34 digits will be displayed, for a <i>time</i> up to 35, for a <i>date</i> (incl. time) up to 36. For a <i>complex number</i>, either part of it will take one <i>LCD</i> row if needed. For a <i>long integer</i>, up to 298 digits can be shown using 7 rows on the <i>LCD</i>; for any longer <i>integer</i>, just its most significant 294 digits may be displayed by SHOW, followed by ellipses. For extracting even greater <i>integers</i> use DUMP (→ 37) and look up App. E. A text <i>string</i> may contain up to 196 <i>characters</i>. <i>Short integers</i> are always displayed with their full precision (switching fonts automatically), so SHOW is obsolete there.
SHUFFLE	→ ↻	
sh.t→kg	[U→] m: ▲ kg ← short ton	[5] (1) {1, 2} → {2} Converts masses, → 152ff.
sign	[PART] sign	[3] (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 number</i> <i>x</i> (→ UNITV). Just for backward compatibility.
SIGNMT	[BIT] ▼ SIGNMT	[4] (0) Sets <i>sign-and-mantissa</i> mode for operations on <i>short integers</i> , → OMS 3.
	[INTS] ▲ SIGNMT	

Item	Keystrokes	Remarks (→ 14ff for general information)
SIM_EQ	MATX SIM EQ $\underline{n}$	[3+] (0) Opens a window for solving a system of n linear equations $(MATA) \cdot \overrightarrow{MATX} = \overrightarrow{MATB}$. If these matrices and vectors are not defined before, they will be created automatically at execution time. → OMS 3 .
sin	TRI sin	[2] (1) {2, 3, 8*, 9*}; {1, 4} → {2} Returns the sine of the angle in X (takes x as tagged, else assumes the current ADM ; → OMS 3).
sinc	TRI sinc	[3] (1) {2, 3, 4, 8*, 9*} Returns $\sin(x)/x$ for $x \neq 0$ and 1 for $x = 0$. Tagged DT 4 input will be converted in radians before computing. Untagged input is taken as radians .
sincπ	TRI sincπ	[3] (1) {2, 3, 4, 8*, 9*} Returns $\sin(\pi x)/\pi x$...
sinh	TRI sinh EXP sinh	[3] (1) {2, 3, 8*, 9*} Returns the hyperbolic sine of x .
SKIP	P.FN P.FN2 SKIP $\underline{n}$	[5+] (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 program section crossed by one or more BACK , CASE , or SKIP jumps, this may well make <u>manual maintenance of all those individual statements</u> necessary.
SL	BIT ▲ SL $\underline{n}$	[5+] (1) {10}  Shifts all bits of x to the left n (≤ 63) bits. ⁵³ Each bit shifted out of the word is shifted into CARRY , and the previous state of CARRY is lost. New bits generated at the right end of x are all zeros. SL 0 executes as NOP , but loads L . → OMS 3 .

⁵³ **SL** on your **WP43** works like n consecutive calls of **SL** on the **HP-16C**. This applies in analogy to **SR** as well.

Item	Keystrokes	Remarks (→ 14ff for general information)
SLVC	ADV SLVC	[4] {1, 2, 3} → {1 or 2 or 3} Solves the cubic equation $ax^3 + bx^2 + cx + d = 0$ and tests the result. Its parameters shall be on the input <i>stack</i> [d, c, b, a, ...]. For <i>real</i> parameters, SLVC returns the three roots in X, Y, Z, and the discriminant $\Delta = b^2c^2 - 4ac^3 - 4b^3d - 27a^2d^2 + 18abcd$ in T. Higher <i>stack registers</i> are kept unchanged.
SLVQ	ADV SLVQ	[3] {1, 2, 3} → {1 or 2 or 3} Solves the quadratic equation $ax^2 + bx + c = 0$ with its parameters on the input <i>stack</i> [c, b, a, ...], and tests the result. The following holds for <i>real</i> parameters: • 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 it returns the 1st <i>complex</i> root in X and the 2nd in Y (the <i>complex conjugate</i> of the 1st). In a routine, the step after SLVQ will be skipped. In either case (and also for <i>complex</i> parameters), SLVQ returns <i>r</i> in Z. Higher <i>stack registers</i> are kept unchanged. L will contain parameter c.
s_m	STAT s_m	[3] (-2) {} → {2} Returns the <i>standard errors</i> (i.e. <i>SDs</i> of the means $\bar{y}$ and $\bar{x}$, → 286 for the formula). Output format will be like with s (→ OMS 2).
s_{mi}	FIT PLLOT s_{mi}	[5] (-1) Pushes the measured precision of the measuring instrument (or system) on the <i>stack</i> (determined according to the <i>MSA</i> explained on pp. 299ff and demonstrated in OMS 2). Note s_{mi} will be only computed for ≥ 30 measured data points. → also CENTRL on p. 28.

Item	Keystrokes	Remarks (→ 14ff for general information)
s_{mw}	STAT s_{mw}	[4] (-1) {} → {2} Returns the <i>standard error</i> for weighted data, i.e. the <i>SD</i> of the mean $\bar{x}_w$ (→ 286 for the formula).
$sn(u,m)$	X.FN Ellipt sn	[4] (2) {2, 3} Returns the <i>elliptic sine</i> of the argument $u = x$ and the parameter $m = y$, → 327.
SNAP	SNAP " DISP "	[2] (0) Stores a snapshot of the calculator screen (a.k.a. screenshot) in a .bmp file in the directory SCREENS on the <i>FD</i>. The file (its name comprises date and time of storage, → OMS 4) can be transferred to your <i>PC</i> and dealt with there. Entered in <i>PEM</i>, inserts a SNAP step to take a snapshot in automatic program execution. If you want to store a snapshot in <i>PEM</i> instead, press ▢ + E <u>simultaneously</u>; this works also if you want to take a snapshot of a <i>temporary information</i>.
SOLVE	ADV SOLVE label (called in <i>RUM</i>)	[3+] {2, 3} Solves the equation $f(v) = 0$, with f as specified by the routine label. → below for more.
	ADV SOLVE v (called in <i>PEM</i>)	[3+] {2, 3} Inserts a SOLVE step. Executed automatically, this will solve the equation $f(v) = 0$ with f as specified by PGMSLV; additionally acts as binary test, so the next program step will be skipped if SOLVE fails to find a root.
Common remarks: ATTENTION: SOLVE fills the <i>stack</i> with x before calling the routine specified. Two initial guesses of the root may be supplied in X and Y when calling SOLVE. It returns... • v_{root} in X with accuracy determined by SETSIG, • the penultimate v-value tested in Y, • $f(v_{root})$ in Z, and		

Item	Keystrokes	Remarks (→ 14ff for general information)
		• T contains ... 0 if SOLVE found a root. 1 if it found a sign reversal but $f(v)$ isn't 'small'. 2 if it found a local or asymptotic extremum. 3 if ≥ 1 start value is out of function domain. 4 if SOLVE found $f(v)$ looking constant. Higher <i>stack registers</i> are not affected. → 335f and OMS 5 for examples. SOLVE may be aborted by EXIT.
Solver	EQN Solver	Submenu for solving a given <u>equation</u> you stored. → OMS 5 for examples.
SOLVING		System flags (→ 129ff)
SPCRES		
SPEC?	TEST ▲ SPEC?	[5] (0) True if x is 'special' (i.e. $\pm\infty$ or NaN).
SQRT	CAT. FCNS SQRT	Points to $\sqrt{x}$ on p. 113. Just for backward compatibility.
SR	BIT ▲ SR $\underline{n}$	[5+] (1) {10}  Shifts all bits in x to the right n (≤ 63) bits in analogy to SL. SR 0 executes as NOP, but loads L. Compare ASR on p. 23 and → OMS 3 for examples.
SSIZE8		System flag (→ 129ff)
STAT	STAT	Menu, → 147.
STATS	Reserved <i>variable</i> for the statistics matrix (→ OMS 2).	
STATUS	FLAG STATUS	[3] Displays the number of free bytes in <i>RAM</i> , all <i>flags</i> set, number of local <i>registers</i> allocated, and <i>system variables</i> (→ OMS 6).
STK	STK	Menu, → 147.
STO	STO $\underline{r}$	[1+] (0) Stores a copy of x into the <i>RoV</i> specified. ⁵⁴

⁵⁴ **L** will not be loaded. This is sensible since x stays unchanged.

Item	Keystrokes	Remarks (→ 14ff for general information)
STOCFG	[STO] Config $\underline{r}$	[3+] (0) Stores the current calculator <i>configuration</i> in an RoV for later use as described in OMS 2 and 7 . RCLCFG recalls such data. Compare also RESET on p. 75. ⁵⁴
STOEL	[MATX] STOEL	[3] (1) {1, 2, 3}
	[STO] ...EL	Stores a copy of x into the current element a_{ij} of the <i>indexed matrix</i> . Compare RCLEL . ⁵⁴
STOIJ	[MATX] [Δ] STOIJ	[3] (1) {1}
	[STO] ...IJ	Sets the index pointers to IP(x) (column number) and IP(y) (row number). Compare RCLIJ . ⁵⁴
STOMAX	[STO] max $\underline{r}$	[2+] (0) {1, 2, 4, 5, 6, 10} Stores the maximum (or minimum) of r and x into the RoV specified. ⁵⁵
	[STO] [Δ] $\underline{r}$	
STOMIN	[STO] min $\underline{r}$	
	[STO] [▽] $\underline{r}$	
STOP	[R/S]	[1] (0) Stops program execution. May be inserted in a program to wait for input, for instance.
stopwatch	→ TIMER	
STOSTK	[STO] Stack $\underline{r}$	[3+] (0) Stores the entire <i>stack</i> in a set of 4 or 8 <i>registers</i> , starting at the destination address specified. Compare RCLSTK . ⁵⁴
STO+	[STO] + $\underline{r}$	[2+] (0) Executes the specified operation on r and stores the result (e.g., $r - x$) in the RoV specified. ⁵⁵ Note every RoV can be a target, so STO× X is legal as well.
STO-	[STO] - $\underline{r}$	
STO×	[STO] × $\underline{r}$	
STO/	[STO] / $\underline{r}$	
sto→kg	[U→] m: kg ← stone	[5] (1) {1, 2} → {2} Converts masses, → 152ff.

⁵⁵ Only legal operations according to the *DT* matrices in [OMS 3](#) will work (→ also the examples given there). **L** will not be loaded (→ previous f.).

⁵⁶ This works almost like STR? in [HP-42S](#).

Item	Keystrokes	Remarks (→ 14ff for general information)
STRI?	TEST STRI?	[4] (0) Returns true if x is a <i>text string</i> . ⁵⁶
STRING	CAT. VAR STRING	Submenu of all <i>text string variables</i> defined at execution time, → 135f.
SUM	STAT SUM	[3] (-2) {} → {2} Recalls the linear sums Σy and Σx . Useful in basic 2D vector algebra. Output labeled in analogy to s .
s_w	STAT s_w	[4] (-1) {} → {2} Returns the sample <i>SD</i> for weighted data (where the weight y of each data point x was entered via $\Sigma+$); → 273ff for the formula.
s_{xy}	FIT s_{xy}	[4] (-1) {} → {2} Returns the <i>sample covariance</i> for the two data sets $\{x\}$ and $\{y\}$ entered via $\Sigma+$. → 281ff for the formula and COV for the <i>population covariance</i> .
SYS.FL		Submenu of all <i>system flags</i> accessible by CF , FC? , FF , FS? , SF , etc.
$s(a)$	FIT Δ $s(a)$	[4] (-2) {} → {2} Returns the standard errors $s(a_1)$ (in Y), and $s(a_0)$ (in X) for the two parameters of the line fitted through the data points accumulated in STATS following the fit model chosen. Works for ExpF , LinF , LogF , OrthoF , and PowerF only. → CENTRL (p. 28), s_{mi} (p. 85), and pp. 290ff.
S.INTS	CAT. VAR S.INTS	Submenu of all <i>short integer variables</i> defined at execution time, → 135f.
$s \rightarrow \text{day}$ $s \rightarrow \text{year}$	U→ Time: s → day etc.	[4] (1) {1, 2} → {2} Convert times (→ 152ff).
T		Reserved <i>variable</i> for register T .
T_0		Constant (→ 116ff).

Item	Keystrokes	Remarks (→ 14ff for general information)
tan	TRI tan	[2] (1) {2, 3, 8*, 9*}; {1, 4} → {2} Returns the tangent of the <i>angle</i> in X (takes <i>x</i> as tagged, else assumes the current ADM ; → OMS 3). May return “ <i>Not a Number</i> ” for $x = \pm 90^\circ$ or equivalents if SPCRES is set.
tanh	EXP tanh	[3] (1) {2, 3, 8*, 9*} Returns the hyperbolic tangent of <i>x</i> .
	TRI tanh	
tb _I →mℓ	U→ V: ▲ ▲ mℓ ← table spoon _I	[6,5] (1) {1, 2} → {2} Convert kitchen volumes (→ 152ff).
tb _{UK} →mℓ	U→ V: ▲ mℓ ← tbs _{UK}	
Tb _{US} →mℓ	U→ V: ▼ mℓ ← Tbsp _{US}	
TDISP	CLK ▲ TDISP <i>n</i>	[4+] (0) Sets time display format. TDISP 0 allows for the full precision from <i>hours</i> to arbitrary fractions of <i>seconds</i> (trailing decimal zeros will not be shown), TDISP 1 or 2 truncates it to <i>hours : minutes</i> only, TDISP 3 to <i>hours : minutes : seconds</i> , and for $4 \leq n \leq 6$, TDISP displays also <i>n</i> – 3 digits for decimal fractions of <i>seconds</i> . Example: With TDISP 5 , a time string might look like 157:26:38.49
TDM24		System flag (→ 129ff)
TEST	TEST	Menu, → 147.
TICKS	P.FN TICKS	[4] (–1) {} → {1} Returns the number of ticks from the real-time clock at execution time. 1 <i>tick</i> = 0.1 s. Counting starts when the calculator is turned on. You can use TICKS for measuring execution times (→ example in OMS 5).

Item	Keystrokes	Remarks (→ 14ff for general information)
TIME	TIME	[3] (-1) {} → {5} Recalls the time from the real-time clock (or the <i>PC</i> clock for the <i>Simulator</i>) at execution (→ OMS 3 for the output format).
TIMER		[2] Opens the timer (a.k.a. stopwatch) application based on the calculator real-time clock. The time counted can be indicated up to 99:59:59.9 (i.e. almost 100 hours – care for reliable power supply), thereafter an overflow will occur but the timer will continue. → 0.1s, ADD, T.RES, and in particular OMS 6 for more information.
TIMES	VARS 	Submenu of all <i>time variables</i> defined at execution time, → 135f.
Time:	Time:	Submenu for time conversions, → 152ff.
T_n	Orthog T_n	[4] (2) {1, 2} → {2} Čebyšev polynomials of 1 st kind, → 290f.
TONE	TONE $\underline{n}$	[4+] (0) Sounds a tone according to n ($= 0 \dots 9$).
ton→kg	m: kg ← ton	[5] (1) {1, 2} → {2} Converts masses, → 152ff.
TOP?	TOP?	[4] (0) Returns ... - false if called with the <i>PP</i> in a <u>subroutine</u>; - true if called in the top routine (i.e. if the program-running <i>flag</i> is set and the <i>SRS</i> pointer is clear).
torr→Pa	F&p: Pa ← torr	[5] (1) {1, 2} → {2} Converts pressures, → 152ff.
T_{PL}		Constants (→ 116ff).
t_{PL}		

Item	Keystrokes	Remarks (→ 14ff for general information)
$t_p(x)$	PROB t:	[4] (1) {1, 2} → {2} <i>Student's t distribution</i> . The degrees of freedom are stored in I (→ 273ff and OMS 2). ⁵⁷
$t_{\Delta}(x)$	$t_p(x)$ etc.	
$t_{\Delta}(x)$		
$t^{-1}(p)$		
TRACE	SF T	System flag (→ 129ff)
TRANS	CAT. FCNS TRANS	Equals [M]^T (→ 112). Just for backw. compatibility.
TRI	TRI	Menu, → 147.
$\text{troz} \rightarrow \text{g}$	U → m: kg ← troy oz. etc.	[5] (1) {1, 2} → {2} Convert masses and volumes (→ 152ff).
$\text{trz} \rightarrow \text{kg}$		
$\text{ts}_I \rightarrow \text{m}\ell$	U → V: ▲ ▲ mℓ ← tea spoon_I	
$\text{ts}_{UK} \rightarrow \text{m}\ell$	U → V: ▲ mℓ ← tsp_{UK}	
$\text{ts}_{US} \rightarrow \text{m}\ell$	U → V: ▼ mℓ ← tsp_{US}	
TVM	FIN TVM	Submenu for the Time Value of Money (→ OMS 6).
T.RES	RESET ⁵⁸	[3] (0) Resets the timer to zero without changing its status (running or stopped). Deletes total time if applicable.
t:	PROB t:	Submenu for the <i>t</i> distribution, → 146.
$t \leftrightarrow r$	STK $t \leftrightarrow r$ r	[3] Swaps <i>t</i> and <i>r</i> , in analogy to $x \leftrightarrow r$.
ULP	INFO ULP	[4] (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 WP43 in the mode set. Thus, 1 is returned for <i>integers</i> . Indicated in STATUS .

⁵⁷ $t^{-1}(p)$ equals $-t_p$ on **HP-21S**.

⁵⁸ Note this **RESET** is just a *label* in **TIMER**, not the command **RESET** above.

Item	Keystrokes	Remarks (→ 14ff for general information)
U_n	X.FN Orthog U _n	[4] (2) {1, 2} → {2} <i>Čebyšev</i> polynomials of 2 nd kind, → 290f.
UNDO		[2] Recalls the <i>stack</i> , <i>STATS</i> , summation <i>registers</i> , and all <i>system flags</i> as they were before the last operation executed – unless that operation is printed in red or orange in this <i>IOI</i> .
UNITV	MATX UNITV	[4] (1) {8, 9} Returns the ‘unit vector’ for the matrix <i>x</i> : each element of the matrix is adjusted so its overall <i>Euclidean</i> norm will become 1 (→ ENORM on p. 38); for a vector, its <i>magnitude</i> will become 1.
	CPX UNITV	[3] (1) {3} Returns a <i>complex number</i> with <i>magnitude</i> $ r = 1$ in direction of <i>x</i> .
UNSIGN	BIT  UNSIGN	[4] (0) Sets <i>unsigned</i> mode for operations on <i>short integers</i> (→ OMS 3). ⁵⁹
	INTS  UNSIGN	
USB		System flags (→ 129ff)
USER	USER	
U→	U→	Menu, → 147.
VAR		Submenu of all <i>variables</i> defined at execution time when calling ST0 , RCL , etc.
VARMNU	P.FN VarMNU <i>label</i>	[3+] Creates a <i>variable menu</i> (i.e. a <i>menu</i> of <i>variables</i>) using the MVAR steps following the <i>global label</i> specified here (→ OMS 4).
VARS	CAT. VARS	Submenu of <u>all</u> <i>variables</i> defined at execution time (superset of <u>V.ALL</u> , <u>ANGLES</u> , etc.), → 135f.
VERS	INFO VERS	[4] (0) Shows the firmware version and build number until next keystroke (→ OMS 2).

⁵⁹ This mode corresponds to UNSGN on [HP-16C](#).

Item	Keystrokes	Remarks (→ 14ff for general information)
VIEW	VIEW <u>s</u>	[2+] (0) Displays the name of the source and its content <i>s</i> left-adjusted in the top numeric row (and below if it needs more space) until the next key is pressed (in programs until the next AVIEW, VIEW, INPUT, PAUSE, PROMPT, STATUS, or STOP occurs). Compare AVIEW. Example: If a variable Test12 contains -123.45 then VIEW ⌫ Test12 ENTER↑ will display in top row Test12 = -123.45
VMDISP		System flag (→ 129ff)
V _m		Constant (→ 116ff).
VOL	CAT. FCNS VOL <u>n</u>	[4+] (0) Sets the buzzer volume to level <i>n</i> (1...11).
VOL+	I/O VOL+	[4] (0) Increases the buzzer volume by one level.
VOL-	I/O VOL-	[4] (0) Decreases the buzzer volume by one level.
V.ALL	CAT. VAR ALL	Submenu of <u>all</u> variables defined at execution time, → 135f.
V:	U→ V:	Submenu for conversions of volumes, → 152.
V4	↗	[2] (2) {8} → {4} Returns the <i>angle</i> between two real 2D or 3D vectors $\vartheta = \arccos\left(\frac{\vec{v}_1 \cdot \vec{v}_2}{ \vec{v}_1 \cdot \vec{v}_2 }\right)$
WDAY	CLK WDAY	[3] (1) {2, 6} → {1} Assumes <i>x</i> containing a <i>date</i> in the format selected (or a <i>real number</i> in a corresponding format) and returns the name of the respective day and a corresponding <i>integer</i> (Monday = 1).⁶⁰

Item	Keystrokes	Remarks (→ 14ff for general information)
Weibl _p	PROB  Weibl:	[5] (1) {1, 2} → {2}⁶⁰ <i>Weibull</i> distribution with its shape parameter b in I and its characteristic lifetime T in J (→ 273ff and OMS 2). Weibl⁻¹ returns the survival time t_s for a given probability p in X, with b in I and T in J.
Weibl _Δ	Weibl _p	
Weibl _Δ	etc.	
Weibl ⁻¹		
Weibl:	PROB  Weibl:	Submenu, → 146.
WHO	INFO WHO	[3] (0) Displays credits to the brave men who made this project work (<i>temporary message</i>).
Wh→J	U→ E: J ← Wh	[4] (1) {1, 2} → {2}
		Converts energies, → 152ff.
W _m	X.FN  W _m	[4] (1) {2, 3}; {1} → {2} 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). → 326ff.
W _p	etc.	
W ⁻¹		
WRITEP	I/O WRITEP	[3] (0) Writes the <i>current program</i> on the <i>FD</i> (→ App. E for the <i>Simulator</i>). You will be prompted for a file <i>name</i> . A file carrying an identical <i>name</i> will be overwritten. Compare READP and → OMS 4 .
WSIZE	BIT  WSIZE <u>n</u>	[4+] (0) Sets the <i>word size</i> (a.k.a. bit width, $1 \leq n \leq 64$) for <i>DT 10</i>.⁶¹ WSIZE 0 sets the <i>word size</i> to maximum, i.e. 64 <i>bits</i>. Reducing <i>word size</i> truncates <i>short integer</i> values in L and in the <i>stack registers</i>. All other memory content stays as is (→ 199f). Increasing the <i>word size</i> adds significant empty bits to <i>short integer</i> values in L and in the <i>stack registers</i>. All other memory content stays as is.
	INTS  WSIZE <u>n</u>	

⁶⁰ **TN:** These day numbers correspond to Chinese weekdays 1 to 6 directly.
For Portuguese / Brazilian weekdays ('segunda-feira' etc.), add 1 to days 1 to 5.

⁶¹ Works as it does on **WP 34S**. It does almost like on **HP-16C**, but with the parameter trailing the command instead of taken from **X** there.

Item	Keystrokes	Remarks (→ 14ff for general information)
$W \rightarrow hp_E$	U→ P: $W \rightarrow hp_E$ etc.	[4] (1) {1, 2} → {2} Convert powers (→ 152ff).
$W \rightarrow hp_M$		
$W \rightarrow hp_{UK}$		
X	Reserved <i>variable</i> for <i>stack register X</i> .	
$\hat{x}$	(FIT) $\hat{x}$	[3] (1) {1, 2} → {2} Returns a forecast $\hat{x}$ for a given y (in X) according to the curve fit model chosen (note $\hat{x}$ may return two results for 3-parameter models, i.e. ParabF , CauchF , and GaussF , → L.R. on p. 57).
$\bar{x}$	(STAT) $\bar{x}$	[3] (-2) {} → {2} Calculates the <i>arithmetic means</i> of the y - and x -data accumulated and pushes these means on the <i>stack</i> . → also s , s_m , and σ .
x^2	(EXP) x^2	[2] (1) {1, 2, 3, 8*, 9*, 10} Return the square or cube of x , respectively.
x^3	(EXP) x^3	
XEQ	(XEQ) <i>label</i> or <i>op.name</i>	[1+] (0) Executes the operation specified (→ 165) or the routine with the <i>label</i> specified. In <i>PEM</i> , XEQ inserts a call to the subroutine with the <i>label</i> specified. Executed automatically, calls the subroutine specified.
XEQ.SKP	(P.FN) P.FN2 XEQ.SKP	[5] (0) Works like XEQ but will return from the subroutine called to the step <u>two</u> steps behind the XEQ.SKP instruction (→ also RTN.SKP).
$\bar{x}_G$	(STAT) $\bar{x}_G$	[4] (-2) {} → {2} Calculates the <i>geometric means</i> of the y - and x -data accumulated and pushes them on the <i>stack</i> (→ 287 for the formula). Output format will be similar to the one of $\bar{x}$. → 102 for ε , ε_m , and ε_p .
$\bar{x}_H$	(STAT)  $\bar{x}_H$	[5] (-2) {} → {2} Calculates the <i>harmonic means</i> of the y - and x -data accumulated and pushes them on the <i>stack</i> .

Item	Keystrokes	Remarks (→ 14ff for general information)
$x_{\max}$	$\boxed{\text{STAT}}$ $\blacktriangle$ $x_{\max}$ etc.	$[5] (-2) \{ \} \rightarrow \{2\}$ Returns the maximum (or minimum) of the y - and x -data accumulated and pushes them on the <i>stack</i> .
$x_{\min}$		
XNOR	$\boxed{\text{BIT}}$ XNOR	$[4, 3] (2)$ Boole's operations working in analogy to AND (→ 22).
XOR	$\boxed{\text{BIT}}$ XOR	
$\bar{x}_{\text{RMS}}$	$\boxed{\text{STAT}}$ $\blacktriangle$ $\bar{x}_{\text{RMS}}$	$[5] (-2) \{ \} \rightarrow \{2\}$ Calculates the <i>quadratic means</i> of the y - and x -data accumulated and pushes them on the <i>stack</i> (→ 287 for the formula). Quadratic means are often used in electrical engineering with alternating currents.
$\bar{x}_w$	$\boxed{\text{STAT}}$ $\bar{x}_w$	$[4] (-1) \{ \} \rightarrow \{2\}$ Returns the <i>arithmetic mean</i> for weighted data (where the weight y of each data point x was entered via $\boxed{\Sigma+}$, → 287 for the formula). → also s_w and s_{mw} .
$\sqrt[y]{x}$	$\boxed{\text{EXP}}$ $\sqrt[y]{x}$	$[3] (2)$ Returns the x^{th} root of y . For $y < 0$ and CPXRES set, $\sqrt[y]{x}$ may return <i>complex numbers</i> .
X.FN	$\boxed{\text{X.FN}}$	Menu, → 148.
$x!$	$\boxed{x!}$	$[2] (1) \{1, 10\}$ Returns the <i>factorial</i> $n!$. Note this is only defined for positive <i>integers</i> . 20! is the biggest factorial $< 2^{64}$. 165! returns the biggest <i>long integer</i> result you can display entirely using SHOW . 450! is the maximum for <i>long integers</i> (returning a 1001-digit <i>integer</i>). Inputs > 450 will be converted to <i>reals</i> before calculating $x!$
		$(1) \{2, 3\}$ Returns $\Gamma(x + 1)$. 2 123.549 956 662 463 236 31 is the maximum <i>real</i> input (returning almost 10^{6145}).
$x:$	$\boxed{\text{U} \rightarrow}$ $x:$	Submenu for conversions of distances, → 152.

Item	Keystrokes	Remarks (→ 14ff for general information)
x→DATE	CLK x→DATE	[3] (1) {2} → {6} Takes the <i>real number</i> x as a date coded in the date format selected (YMD, DMY, or MDY) and converts it to a proper <i>date</i> .
x→α	α.FN x→α r	[3+] (1) {1, 2, 10} → {7} Interprets $IP(x)$ as a <i>character code</i> , converts it to the respective <i>character</i> , and appends it to r . For destination X , it returns just the <i>character</i> converted. If X contains a text string already, $x→α$ appends the entire string to r . ⁶²
x↔	x↔ r STK x↔ r	[2+] Swaps x and r , in analogy to $x↔y$. Will be listed like $x↔ J$, $x↔ .12$, $x↔ →12$, etc. in programs.
x↔y	x↔y	[1] Swaps the contents of <i>stack registers</i> X and Y .
x=+0? x=-0?	TEST ▲ x=+0? etc.	[4] (0) {1, 2, 3, 10} These two tests are for comparing <i>short integers</i> in modes 1COMPL and SIGNMT, and for <i>real</i> or <i>complex numbers</i> if SPCRES is set. Then, e.g., 0/(-7) will display -0.
x<? x≤?	TEST x<? s etc.	[3+] (0) {1, 2, 4, 5, 6, 7, 10} Compare x with s . <i>Text strings</i> are compared according to their sorting order. → also <i>Addressing Tables</i> in OMS 1 . 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.
x=?	TEST x=? s	[3+] (0) Compares x with s .
x≈?	... ▲ x≈? s	[4+] (0) {2, 3, 4, 5, 8, 9} Will be true if the <u>rounded</u> values of x and s are equal (compare ROUND on p. 78).

→ x<? for more.

⁶² This works like XTOA on **HP-42S** but with an arbitrary destination instead of the $α$ -register there.

Item	Keystrokes	Remarks (→ 14ff for general information)
$x \neq ?$	TEST $x \neq ?$ S	[3+] (0) Compares x with s .
$x \geq ?$	TEST $x \geq ?$ S etc.	[3+] (0) {1, 2, 4, 5, 6, 7, 10}
$x > ?$		Compare x with s .
Y	Reserved <i>variable</i> for <i>stack register</i> Y.	
$\hat{y}$	FIT $\hat{y}$	[3] (1) {1, 2} → {2} Returns a forecast y (in X) for a given x according to the curve fit model chosen (→ L.R. on p. 57).
yd→m	U→ $x:$ m ← yard	[5] (1) {1, 2} → {2} Converts distances, → 152ff.
YEAR	CLK YEAR	[4] (1) {2, 6} → {1} Assumes x containing a <i>date</i> in the format selected (or a <i>real number</i> in corresponding format) and extracts the year.
year→s	U→ Time: s ← year	[4] (1) {1, 2} → {2} Convert times and distances (→ 152ff).
yīn→m	U→ $x:$ ▲ m ← yīn	
YMD		System flag (→ 129ff)
y^x	EXP y^x	[2] (2) Raises y to the power of x . For $y < 0$ and CPXRES set, it may return <i>complex numbers</i> .
$Y_y(x)$	X.FN $Y_y(x)$	[4] (2) {1, 2} → {2} Returns the <i>Bessel function of second kind</i> and order y , → 329.
Y.MD	CLK ▲ Y.MD	[4] (0) Sets the format yyyy-mm-dd for <i>dates</i> .
$y \rightleftharpoons r$	STK $y \rightleftharpoons r$	[3+] Swaps y and r , in analogy to $x \rightleftharpoons r$.
Z	Reserved <i>variable</i> for <i>stack register</i> Z.	
Z_0		Constant (→ 116ff).
zhàn→m	U→ $x:$ ▲ m ← zhàng	[5] (1) {1, 2} → {2} Converts distances, → 152ff.

Item	Keystrokes	Remarks (→ 14ff for general information)
ZOOM	EQN Solver Draw ZOOM	[5] (0) Zooms the current plot out in the <i>x</i> -axis → OMS 5 .
	FIT ASSESS ZOOM	
z↔r	STK z↔r r	[3+] Swaps <i>z</i> and <i>r</i> , in analogy to x↔z .
α		Constant (→ 116ff).
αCAP		System flag (→ 129ff)
αINTL	CAT. CHARS αINTL	Submenu, → 135f.
	1/x (A)	Catalog in AIM (→ 149).
αLENG	INFO αLENG s	[4+] (-1) {} → {1}
	α.FN αLENG s	Returns the number of <i>characters</i> found in the <i>text string s</i> . ⁶³
αMATH	CAT. CHARS αMATH	Submenu, → 135ff.
	- (~)	Menu in AIM, → 150.
αPOS	INFO αPOS s	[4+] (-1) {} → {1}
	α.FN αPOS s	Looks in the <i>text string s</i> for the <i>target</i> given in X . If a match is found, αPOS returns the position number where the <i>target</i> was found (counting the left-most <i>character</i> as position 0). If a match is not found, αPOS returns -1. ⁶³ The <i>target</i> may be an individual <i>character code</i> or a <i>text string</i> . αPOS saves a copy of the <i>target</i> in L .
αRL	α.FN αRL r	[3+] (0) Rotates the <i>text string r</i> by <i>x</i> (≥ 0) symbols to the left. αRL 0 executes as NOP , but loads L . ⁶³
αRR	α.FN αRR r	[3+] (0) Works like αRL but rotates to the right.
αSL	α.FN αSL r	[3+] (0) Shifts the <i>x</i> leftmost symbols out of the <i>text string r</i> . This allows for deleting the first <i>x</i> symbols in a string. αSL 0 executes as NOP , but loads L . ⁶³

⁶³ This will throw an error if there is no *text string* or an empty string in *r* at execution time. It corresponds to **ALENG**, **POSA**, **ASHF**, or **ATOX** on **HP-42S**, respectively

Item	Keystrokes	Remarks (→ 14ff for general information)
α SR	α .FN α SR r	[3+] (0) Works like α SL but for the x rightmost symbols out of r , deleting the last x symbols in the string. ⁶³
α .FN	α .FN	Menu, → 150.
A...Ω	CAT. CHARS A...Ω	Submenu of Greek letters, → 135ff.
α ·	CAT. CHARS α ·	Submenu for punctuation, → 135ff.
	 	Menu in AIM, → 150.
$\alpha \rightarrow x$	α .FN $\alpha \rightarrow x$ r	[3+] (-1) {} → {10 ₁₆ } Pushes the <i>character code</i> of the leftmost <i>character</i> in the <i>text string</i> r on the <i>stack</i> and removes this <i>character</i> from the string. ⁶³
$\beta(x,y)$	α .FN  $\beta(x,y)$	[4] (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. → $\Gamma(x)$.
γ		Constants (→ 116ff).
γ_{EM}		
γ_p		
Γ_p	α .FN  Γ_p	[4,5] (2) {1, 2}
Γ_q	α .FN  Γ_q	Returns the <i>regularized gamma function</i> (one of two kinds; → 326ff).
Γ_{xy}	α .FN  Γ_{xy}	[5] (2) {1, 2} → {2}
γ_{xy}	α .FN  γ_{xy}	Returns the <i>upper or lower incomplete Gamma function</i> , → 326ff.
$\Gamma(x)$	PROB  $\Gamma(x)$	[4] (1) {2, 3}; {1} → {2} Returns $\Gamma(x)$. Note $\alpha x!$ calls $\Gamma(x+1)$. Compare also $\ln \Gamma$ on p. 55.

Item	Keystrokes	Remarks (→ 14ff for general information)
Δx	STAT ▲ Δx	[5] (-2) { } → {2} Returns the range (i.e. max – min) of the y - and x -data accumulated in STATS and pushes them on the stack . ⁶⁴
Δv_{cs}		Constant (→ 116ff).
$\Delta\%$	△%	[2] (1) {1, 2} → {2} Returns $100^{x-y}/y$ leaving y unchanged. You can use it also for calculating markups or margins as explained in OMS 2 .
ϵ	STAT ε	[4] (-2) { } → {2} Calculates the <i>scattering factors</i> ϵ_y and ϵ_x for <i>log-normally</i> distributed sample data and pushes them on the stack . ε works for the <i>geometric mean</i> $\bar{x}_g$ in analogy to the SD for the <i>arithmetic mean</i> $\bar{x}$ but <u>multiplicative</u> instead of additive. → 278ff.
ϵ_0		Constant (→ 116ff).
ϵ_m	STAT ε_m	[4] (-2) { } → {2} Works like ε but returns the <i>scattering factors</i> of the two <i>geometric means</i> (in analogy to the <i>standard error</i> for <i>arithmetic means</i>). → 278ff.
ϵ_p	STAT ε_p	[4] (-2) { } → {2} Works like ε but returns the <i>scattering factors</i> of the two <i>populations</i> . → 278ff.
$\zeta(x)$	X.FN ▲ $\zeta(x)$	[5] (1) {2, 3} Returns <i>Riemann's Zeta</i> , → 332.
$Z(\varphi, m)$	X.FN Ellipt $Z(\varphi, m)$	[5] (1) {2, 3} Returns <i>Jacobi's Zeta</i> , → 327. Note the <i>name</i> of this function begins with a capital Greek <u>Zeta</u> (not with a Latin Z , watch the sorting order).

⁶⁴ Note Δx is the worst quantity measuring statistical process scattering since it is very sensitive to random individual outliers – **SDs** (and standard errors) are far better.

Item	Keystrokes	Remarks (→ 14ff for general information)
λ_c		Constants (→ 116ff).
λ_{cn}		
λ_{cp}		
μ_0		Constants (→ 116ff).
μ_B		
μ_e		
μ_e/μ_B		
μ_n		
μ_p		
μ_u		
μ_μ		
π		[2] (-1) {} → {2} Recalls a 34-digit approximation of π. May be used in <i>complex</i> input, too.
Π_n	 Π_n <i>label</i>	[3+] Computes a finite product using the routine specified (→ <i>OMS 5</i>). ATTENTION: Π_n fills all <i>stack registers</i> with x before calling the routine specified.
$\Pi(n,m)$	  $\Pi(n,m)$	[4] (2) {2, 3} Returns the <i>complete elliptic integral of the 3rd kind</i> for $n = x$ and <i>real number</i> $m = y$. Note $0 \leq m < 1$. → 327.
Σ		Menu, → 151.
σ	 σ	[3] (-2) {} → {2} Works like <i>s</i> but returns the <i>SD</i> of the two <i>populations</i> instead (→ 273ff).
σ_B		Constant (→ 116ff).

Item	Keystrokes	Remarks (→ 14ff for general information)
Σ^1/x	  Σ^1/x etc.	[4] (-1) {} → {2} Recall the corresponding statistical sums, needed for statistical analyses and regressions beyond the linear model. Calling these sums by names significantly improves program readability. Note they are stored in dedicated summation <i>registers</i> of your WP43 (→ 189ff and OMS 1). For space reasons, two labels are abbreviated: $\Sigma \ln xy$ denotes $\Sigma \ln(x) \ln(y)$ and $\Sigma \ln y/x$ denotes $\Sigma \ln(y)/x$. Note that logarithmic or inverted sums may become infinite or even non-numeric depending on your input data. If this happens no error will be thrown, regardless of the status of SPCRES. Do not bother.
Σ^1/x^2		
Σ^1/y		
Σ^1/y^2		
$\Sigma \ln^2 x$	 $\Sigma \ln^2 x$ etc.	
$\Sigma \ln^2 y$		
$\Sigma \ln x$		
$\Sigma \ln xy$		
$\Sigma \ln y$		
$\Sigma \ln y/x$	 $\Sigma \ln y/x$	
Σ_n	 Σ_n <u>label</u>	[3+] Computes a finite sum using the routine specified (→ OMS 5). ATTENTION: Σ_n fills all <i>stack registers</i> with x before calling the routine specified.
σ_w	 σ_w	[4] (-1) {} → {2} Works like s_w but returns the <i>SD</i> of the <i>population</i> instead (→ 273ff).
Σx	 Σx etc.	[3,4,5] (-1) {} → {2} Recall the corresponding statistical sums, necessary for statistical analyses and regressions (→ Σ^1/x for more).
Σx^2		
$\Sigma x^2 \ln y$	 $\Sigma x^2 \ln y$ etc.	
$\Sigma x^2 y$		
$\Sigma x^2/y$	  $\Sigma x^2/y$	
Σx^3	  Σx^3 etc.	
Σx^4		
$\Sigma x \ln y$	 $\Sigma x \ln y$	
Σxy	 Σxy	

Item	Keystrokes	Remarks (→ 14ff for general information)
$\Sigma x/y$	Σ $\blacktriangle$ $\Sigma x/y$	
Σy	Σ Σy	
Σy^2	etc.	
$\Sigma y \ln x$	Σ $\Sigma y \ln x$	
$\Sigma +$	FIT $\Sigma +$	[3] {8} → {2} ⁶⁵
	STAT $\Sigma +$	If $\mathbf{X}$ contains an $n \times 2$ matrix then $\Sigma +$ copies it to STATS. The LCD will show the last data point added (equivalent to the last row of the input matrix) and said matrix will be saved in $\mathbf{L}$. → OMS 2.
		{1, 2} ⁶⁵ Adds one 2D data point (x, y), i.e. adds one row to STATS. Note both x and y must be of DT 1 or 2.
($\Sigma +$)	in TIMER	→ ADD.
$\Sigma -$	FIT $\Sigma -$	[3] (-2) {1, 2} ⁶⁵
	STAT $\Sigma -$	Reverts $\Sigma +$, deleting the last row of STATS and returning this last data point to the <i>stack</i> . ⁶⁶
Φ		Constants (→ 116ff).
Φ_0		
$\chi^2_p(x)$	PROB $\chi^2:$	[4] (1) {1, 2} → {2} <i>Chi-square distribution</i> with its degrees of freedom given in $\mathbf{I}$ (→ 273ff and OMS 2). ⁶⁷
$\chi^2_{\Delta}(x)$	$\chi^2_p(x)$	
$\chi^2_{\Delta}(x)$	etc.	
$(\chi^2)^{-1}$		
$\chi^2:$	PROB $\chi^2:$	Submenu, → 146.

⁶⁵ $\Sigma +$ and $\Sigma -$ return *temporary information* (as shown in OMS 2) and disable ASL.

⁶⁶ $\Sigma -$ takes no data from the *stack* here; and it does not subtract 2D vectors like on very early HP pocket calculators.

⁶⁷ $(\chi^2)^{-1}$ corresponds to χ_p^2 on HP-21S (note there is no minus here – compare the other distributions).

Item	Keystrokes	Remarks (→ 14ff for general information)									
$\psi(u,m)$	X.FN Ellipt $\psi(u,m)$	[5] (2) {2, 3} Returns the <i>elliptic amplitude</i> for the argument $u = x$ and parameter $m = y$, → 327.									
ω		Constant (→ 116ff).									
$(-1)^x$	X.FN Δ $(-1)^x$	[5] (1) {1, 2, 3, 8*, 9*, 10} If x is non-integer, returns $\cos(\pi x) + i \sin(\pi x)$.									
$[M]^T$	MATX $[M]^T$	[3] (1) {8, 9} Returns the transpose of the matrix x . 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.									
$[M]^{-1}$	MATX $[M]^{-1}$	[3] (0) {8, 9} Takes the square matrix in X and inverts it in-situ.									
$+$	+	[1] (2) Returns $y+x$ for compatible objects. ⁶⁸									
$+/-$	+/- for closed x	[1] (1) 'Unary minus', returns $x \times (-1)$. ⁶⁸									
$\pm\infty?$	INFO ±∞?	[4] (0) {2} → {1} Tests x for infinity. Returns <table> <tr> <td>true</td> <td>1</td> <td>for $x = +\infty$,</td> </tr> <tr> <td>true</td> <td>-1</td> <td>for $x = -\infty$, and</td> </tr> <tr> <td>false</td> <td>0</td> <td>else.</td> </tr> </table>	true	1	for $x = +\infty$,	true	-1	for $x = -\infty$, and	false	0	else.
true	1	for $x = +\infty$,									
true	-1	for $x = -\infty$, and									
false	0	else.									
$-$	-	[1] (2) Returns $y-x$ for compatible numeric objects. ⁶⁸									
$-\infty$		Constant (→ 116ff).									
$\times$	x	[1] (2) Returns $y \times x$ for compatible numeric objects. ⁶⁸									

⁶⁸ → *DT* combination matrices in **OMS 3** for details and compatibility. Note that sizes must match for matrix operations; and for dividing by a matrix x , it must be invertible.

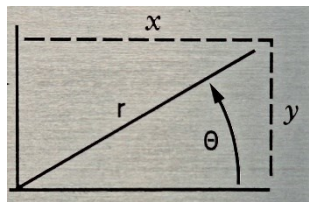
Item	Keystrokes	Remarks (→ 14ff for general information)
×MOD	INTS ×MOD	[4] (3) {1, 2, 10} Returns $(z \times y) \bmod x$ for $x > 1, y > 0, z > 0$. ⁶⁹
/	/	[1] (2) Like × , but returns $y \times x^{-1}$. ⁶⁸ Returns $y \div x$ if both y and x are of <i>DT 1</i> or <i>10</i> ; → IDIV .
^MOD	INTS ^MOD	[4] (3) {1, 10} Returns $(z^y) \bmod x$ for $x > 1, y > 0, z > 0$. ⁶⁹
?BATT	INFO ?BATT	[4] (-1) {} → {2} Measures the battery voltage U between 1.9 V and 3.4 V. Returns U with a resolution of 1 mV. U < 2.5 V is considered low, U < 2.0 V may turn off your <i>WP43</i> automatically – but a timer running in background will continue! Note the value returned by ?BATT is an almost open-circuit voltage (→ App. G).
?BestF	INFO  ?BestF	[4] (-1) {} → {1} Returns the current setting of BestF , encoded as a <i>long integer</i> (as explained for BestF).
?DISK	INFO DISK?	[4] (0) Returns <i>SM's DMCP</i> information about the <i>FD</i> looking like this → Press any key to close. Note for information about free space on the <i>FD</i> you have to connect it to a <i>PC</i> (e.g., by ActUSB) and call disk <i>Properties</i> there.
?HIDE	DISP  ?HIDE	[4] (-1) {} → {1} Returns the current setting of HIDE .
	INFO  ?HIDE	

⁶⁹ Compare **MOD**.

Item	Keystrokes	Remarks (→ 14ff for general information)												
?ISM	 ?ISM	[3] (-1) {} → {1} Returns the <i>integer sign mode</i> set for <i>short integers</i>: <table> <tr> <td>true</td><td>-1</td><td>for sign and mantissa mode,</td></tr> <tr> <td>false</td><td>0</td><td>for unsigned,</td></tr> <tr> <td>true</td><td>1</td><td>for 1's complement, or</td></tr> <tr> <td>true</td><td>2</td><td>for 2's complement.</td></tr> </table>	true	-1	for sign and mantissa mode,	false	0	for unsigned,	true	1	for 1's complement, or	true	2	for 2's complement.
true	-1	for sign and mantissa mode,												
false	0	for unsigned,												
true	1	for 1's complement, or												
true	2	for 2's complement.												
?J/G	  ?J/G	[4] (-1) {} → {6} Returns the current setting of J/G. <i>Startup default</i> is 1752-09-14.												
?KTYP	 ?KTYP 	[3+] (-1) {} → {1} Assumes a key code in the <i>RoV</i> specified (→ KEY?), checks it, and returns its key type: • 0 ... 9 if it corresponds to a digit  ... , • 10 if it corresponds to , , or , • 11 if it corresponds to  or , • 13 if it corresponds to a <i>softkey</i>, and • 12 if it corresponds to any other key. Helps in user interaction with routines (→ OMS 4).												
?LocR	 ?LocR	[4] (-1) {} → {1} Returns the number of <i>local registers</i> currently allocated for the <i>current routine</i>.												
?MEM	 ?MEM	[3] (-1) {} → {1} Returns the number of free <i>bytes</i> in <i>RAM</i> (i.e. the pool size), taking into account all <i>registers</i> currently allocated and their current contents.												
?M.DIM	 ?DIM	[4] (-2) {8, 9} → {1} 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.												
	  ?DIM													

Item	Keystrokes	Remarks (→ 14ff for general information)
?RANGE	?RANGE	[4] (-1) {} → {1}
	?RANGE	Returns the current range setting.
?RM	?RM	[3] (-1) {} → {1} Returns the floating point rounding mode (→ RM).
?SDIGS	?SDIGS	[4] (-1) {} → {1} Returns the number of significant digits set by SETSIG. Also returned by STATUS.
?SSIZE	?SSIZE	[3] (-1) {} → {1} Returns the number of <i>stack registers</i> currently allocated, 4 or 8.
?VOL	?VOL	[4] (-1) Returns the present buzzer volume.
?WSIZE	?WSIZE	[4] (-1) {} → {1} Recalls the <i>word</i> size set.
→		Reserved symbol for indirect addressing.
→BIN	→BIN	[4] (1) {1, 2} → {10} Converts a closed number to a binary <i>short integer</i> .
→DATE	→DATE	[3] (3) {1, 2} → {6} Assumes three components of a date (year, month, and day) supplied on the <i>stack</i> in proper order for the date format selected and converts them to a single <i>date</i> in <i>x</i> . Thus inverts DATE→.
→DEC	→DEC	[4] (1) {1, 2} → {10} Converts a closed number to a decimal <i>short integer</i> .
→DEG	→DEG	[3] (1) {1, 2, 4} → {4} Convert <i>angles</i> as described on pp. 163f.
→D.MS	→D.MS	
→GRAD	→GRAD	

Item	Keystrokes	Remarks (→ 14ff for general information)
→HEX	INTS  →HEX	[4] (1) {1, 2} → {10} Converts a closed number to a hexad. <i>short integer</i> .
→HR	CAT. FCNS →HR	(1) {5} → {2} Operates on <i>times</i> like →REAL below. Just for backward compatibility.
→H.MS	h.ms for closed x	[2] (1) {1, 2, 5} → {5} Converts a decimal x to a sexagesimal <i>time</i> (→ 168). Does not change a sexagesimal <i>time</i> .
→INT	# base for closed x	[2+] (1) {1, 2, 10} → {10} Converts IP(x) to a <i>short integer</i> of the base specified. Conversion to decimal may be abbreviated by #D , to hexadecimal by #H . Note #I returns a <i>long integer</i> instead (→ 49).
→MIL	 →MIL	[3] (1) {1, 2, 4} → {4}
→MUL π	 →MUL π	Converts <i>angles</i> as described on pp. 163f.
→OCT	INTS  →OCT	[4] (1) {1, 2} → {10} Converts a closed number to an octal <i>short integer</i> .
→POL		[2] {1, 2} → {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). Inverted by →REC. For switching the display format of <i>complex numbers</i> , → POLAR on p. 129.
→RAD	 →RAD	[3] (1) {1, 2, 4} → {4} Converts <i>angles</i> as described on pp. 163f.



Item	Keystrokes	Remarks (→ 14ff for general information)
→REAL	for closed x	[2] {1, 2, 4, 5, 6, 10} → {2} Converts x to a <i>real number</i>. Any object displayed sexagesimal (e.g., a <i>time</i>) will be converted in a decimal number and shown in the format set. For <i>dates</i>, the date format chosen is taken into account. Fraction display will return to decimal numbers (→ FRACT and PROPFR). To return the <i>real</i> part of a <i>complex</i> object, take Re (→ 75). To cut a <i>complex</i> object in its two parts, use [CC] or CX→RE (→ 33).
→REC		[2] {1, 2, 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 <i>Cartesian</i> coordinates or components (x, y). Inverted by →POL. For switching the display format of <i>complex</i> objects, → POLAR on p. 129.
↑Lim	↑Lim etc.	[4] Reserved <i>real variables</i> for the upper and lower limit of integration (→ <i>OMS</i> 5).
↓Lim		
↻	r_1, r_2, r_3, r_4	[3++++] Shuffles the contents of the four bottom <i>stack registers</i> X, Y, Z, and T at execution time. Examples: xyyz works like ENTER↑ (but enables <i>ASL!</i>), yxzt works like , yztx works like in a 4LS, txyz works like in a 4LS, but also yytt or zzzx are possible. ATTENTION: This is a very powerful command although it does not look it. Note it will affect the <u>bottom 4 stack registers only</u>; there is <u>no</u> connection to A ... D, regardless of <i>stack size</i>. Playing with , you may lose <i>stack</i> contents and make a mess of the <i>stack</i> easily.

Item	Keystrokes	Remarks (→ 14ff for general information)
M	MATX M	[3] (1) {8} → {2}; {9} → {3} Requires a square matrix in X and returns its determinant. The original matrix is saved in L.
x	(I)x 	[2] (1) {1, 2, 4, 10} Returns the absolute (unsigned) value of x. (1) {3} → {2} Returns the <i>magnitude</i> $\sqrt{\text{Re}(x)^2 + \text{Im}(x)^2}$ in X. (1) {8*}; {9*} → {8} Returns a <i>real</i> matrix with the absolute values or <i>magnitudes</i> of all matrix elements. Compare ENORM.
	(I)	[2] (2) {2, 3}; {1} → {2} Returns $\left(\frac{1}{x} + \frac{1}{y}\right)^{-1}$; useful in electrical engineering especially. Returns 0. for $xy = 0$.
%	(FIN) %	[3] (1) {1, 2} → {2} Returns $xy/100$, leaving y unchanged.
%ile	(STAT) ▲ %ile	[4] (-2) {} → {2} Pushes the <i>x%-percentile</i> of the accumulated y- and x-data on the <i>stack</i>. Note this may only make sense for data sets containing ≥ 50 samples.
%MRR	(FIN) %MRR	[3] (3) {1, 2} → {2} Returns the <i>mean rate of return</i> in % per period,⁷⁰ i.e. $100 \cdot \left(\sqrt[z]{x/y} - 1\right)$ with $x = \text{FV} = \text{future value after } z \text{ periods}$, and $y = \text{PV} = \text{present value}$.
%T	(FIN) %T	[3] (1) {1, 2} → {2} Returns $100 \cdot x/y$, interpreted as % of <u>t</u>otal. Leaves y unchanged.

⁷⁰ **TNG**: *MRR* = mittlere oder durchschnittliche Rendite über mehrere Perioden.

Item	Keystrokes	Remarks (→ 14ff for general information)
%Σ	FIN %Σ	[3] (1) {1, 2} → {2} Returns $100 \ x / \sum x$.
%+MG	FIN %+MG	[3] (2) {1, 2} → {2} Calculates a sales price by adding a margin of x % to the cost y . 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 .
$\sqrt{x}$	$\sqrt{x}$	[1] (1) {1, 2, 3, 8*, 9*, 10}; ({1} → {2}, {1, 2} → {3}) Returns the square root of x .
∞		Constant (→ 116ff).
∫	ADV ∫fdx ∫ <u>label</u> (called in <i>RUM</i>)	[4+] {2} Integrates the expression specified in the routine <u>label</u> . → common remarks for more.
	ADV ∫fdx ∫ <u>var</u> (called in <i>PEM</i>)	[4+] {2} Inserts a step ∫fd followed by <u>var</u> . Executed automatically, it integrates the expression given in the routine specified by PGMINT over the <i>integration variable</i> <u>var</u> . → common remarks for more.
	EQN ∫f ∫	[4] Integrates the current <u>equation</u> .
		Common remarks: ATTENTION: ADV ∫fdx ∫ fills all <i>stack registers</i> with x before calling the routine specified. Lower and upper integration limits must be supplied by the corresponding <i>variables</i> ↓Lim and ↑Lim , accuracy of the integrand by ACC . ∫ returns the (approximate) integral in X and an upper limit of its uncertainty in Y (i.e. the computed value of the integral is $x \pm y$). Z is cleared. Higher <i>stack registers</i> are not affected. → OMS 5 for examples and App. J for details. ∫ may be aborted by EXIT .

Item	Keystrokes	Remarks (→ 14ff for general information)
$\int f$	 $\int f$	Submenus, → 142f.
$\int f dx$	 $\int f dx$	
$\angle$		$[2] \{1\} \{2\} \rightarrow \{4\}$ Returns 180° (or equivalent) for $x < 0$ and 0 else. $(1) \{3\} \rightarrow \{4\}$ Returns the <i>phase</i> or argument (compare x) $\arg(x) = \arctan\left(\frac{\operatorname{Im}(x)}{\operatorname{Re}(x)}\right)$ $(1) \{9^*\} \rightarrow \{8\}$ Returns a matrix with the <i>phases</i> of all elements of the input matrix. Compare x.
$\angle \rightarrow$		Menu of angular conversions, → 151.
 ADV	  ADV	 Prints the current contents of the print buffer and a linefeed. ATTENTION: The printer will print only when a linefeed is sent to it.
 CHAR	  CHAR n	 Sends a single <i>character</i> (with the code specified) to the printer. <i>Character codes</i> $n > 127$ can only be specified indirectly.  MODE setting will be honored. →  ADV.
 DLAY	  DLAY n	 Sets a delay of n ticks (→ TICKS) to be used with each linefeed on the printer.
 MODE	  MODE n	 Sets print mode. Legal print modes are: 0: Use the printer font and <i>character</i> set wherever possible (default). All <i>characters</i> 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 <i>characters</i> in a row. 2: Use the small display font, which allows for packing even more info in a row.

Item	Keystrokes	Remarks (→ 14ff for general information)
		3: Send the output to the serial line. Works for plain ASCII only – no <i>characters</i> will be translated. Line setup is the same as for serial communication: 9600 baud, 8 bits, no parity.
	 	(0) Prints the listing of the <i>current program</i> (→ OMS 4), one row per step.
	  <u>s</u>	(0) Prints the content of the <i>RoV</i> specified, right adjusted, <u>without</u> labeling the output. If you want a heading label, compose the string in X first or use  REGS . →  .
	 	(0) Interprets <i>x</i> in the form <i>sss.nn</i> . Prints the contents of <i>nn registers</i> starting with number <i>sss</i> . Each <i>register</i> takes one row starting with its label. ATTENTION for <i>nn</i> = 0 : • For <i>sss</i> ∈ [0; 99], printing will stop at R99. • For <i>sss</i> ∈ [100; 111], printing stops at K. For <i>sss</i> ≥ 112, printing stops at the highest allocated local <i>register</i> .
	 	(0) Prints the entire <i>stack</i> contents. Each <i>stack register</i> prints in a separate row starting with its label.
	  <u>n</u>	(0) Positions the print head to print column <i>n</i> (0 to 165, where <i>n</i> > 127 can only be specified indirectly). Useful in formatting (in  MODE 1 or 2 in particular). Allows also for printer plots. If <i>n</i> is less than current print head position, a linefeed will be entered to reach the new position. →  .
	 	(0) Prints all <i>variables</i> and <i>global</i> program <i>labels</i> in alphabetic order. The <i>variables</i> are printed first; if you are not interested in the program <i>labels</i> , press (R/S) to stop the listing.
	 	(-1) Returns the number of print columns that <i>x</i> would take in the print mode set. →  MODE . Second use: in  MODE 1 or 2,  WIDTH returns the width of <i>x</i> in px (including the last column being always blank) in the specified font.

Item	Keystrokes	Remarks (→ 14ff for general information)
	  x	(0) Prints <i>x</i> without a heading label. →  ADV .
	  Σ	(0) Prints the summation <i>registers</i> . Each <i>register</i> prints in one row starting with its label.
	  # <i>n</i>	(0) Sends a single <i>byte</i> , without translation, to the printer (e.g., a control code). <i>n</i> > 127 can only be specified indirectly.  MODE setting will not be honored. →  ADV .
#B	 #B	[4] (1) {10} Counts the bits set in <i>x</i> (like on <i>HP-16C</i>).
#DEC		Reserved <i>integer variable</i> for the parameter specified in last call of ALL , FIX , SCI , ENG , or DSP .

Constants

Press  to access a *catalog* of 79 physical, astronomical, and mathematical constants sorted alphabetically. *Names* of **mathematical** and **astronomical** (incl. **surveying**) constants are printed on colored background in the table starting overleaf. Values of physical constants incl. their **relative SDs** below are printed on white if they are exactly defined – the darker grey the background, the less precisely the particular value is known.⁷¹ Constants defining standard conditions are printed on **orange**.

⁷¹ For most physical constants in **CONST**, their precise numeric values and their **relative SDs** are from *CODATA 2022*, copied in April 2025. These are the best values known in the scientific community today, agreed on by the national standards institutes worldwide (e.g., by *NIST* and *PTB*). Note that the values of the six **fundamental constants** (printed **bold** in the table) all feature less than 11 significant digits.

Relative SDs are included in this table here though not contained in **CONST**. They are important for computing the **precision of the results** you obtain using these constants given, through the process of *error propagation* (→ [App. I](#)) going back to *Carl F. Gauss* (1777 - 1855). This is essential for trustworthy results – not only in science (remember each and every scientific result shall include an indication of its uncertainty). Consult, e.g., <http://physics.nist.gov/cgi-bin/cuu/Info/Constants/definitions.html>, giving a nice introduction and further links. There is simply **no way** yardstick measurements

In this chapter, we use commas as radix marks for better visibility and multiplication dots for space reasons. Formulas are printed where applicable.

Name	Numeric value and rel. SD	Remarks
a (0) ⁷²	365,242 5 d (<i>per definition</i>)	<i>Gregorian</i> year
a₀	5,291 772 105 44 · 10 ⁻¹¹ m (1,6 · 10 ⁻¹⁰)	<i>Bohr</i> radius $a_0 = \alpha / 4\pi R_\infty$
a_{Moon}	3,844 · 10 ⁸ m (1 · 10 ⁻³)	Semi-major axis of the Moon's orbit around the Earth ≈ 1,3 <i>light seconds</i> .
a_⊕	1,495 979 · 10 ¹¹ m (1 · 10 ⁻⁶)	Semi-major axis (≈ radius) of the Earth's orbit around the Sun. Within the uncertainty stated here, it equals 1 <i>astronomic unit</i> ≈ 499 <i>light seconds</i> ≈ 8 <i>light minutes</i> .
c	2,997 924 58 · 10 ⁸ m/s	Speed of light in vacuum ≈ 300 000 km/s = 300 km/ms = 300 m/μs = 30 cm/ns = 0,3 mm/ps = 0,3 μm/as ^{.73}

can yield results accurate to four decimals.

Apropos, the terms *resolution*, *precision*, and *accuracy* are confused frequently in measuring (and hence in advertising all the more, sometimes on purpose). In a nutshell, **resolution** is the least significant digit a measuring instrument indicates. Using this instrument to measure the same object or property multiple times under identical conditions, you get an idea of its **repeatability** (or **precision**); this will include your interaction with said instrument and can be no better than its *resolution* but may be significantly worse – a factor of ten or more may be observed in real life. **Accuracy** of a measuring instrument can never be any better than its *repeatability*, however.

Since we can't know anything about any real-life object or process any better than we can measure it, these considerations are fundamental. We recommend watching them to distinguish truth from fake – in your very own interest.

⁷² The numbers in parentheses in this column are to support determination of parameters for CNST (→ IOI). The names of the constants may refer to the pioneers in this area.

⁷³ If you need some 0,2 ... 10 ns fix delays for electronic pulse signals, you can achieve these using coaxial cables easily. At the long end you will need some space.

Name	Numeric value and rel. SD	Remarks
c_1 (5)	$3,741\,771\,85\dots \cdot 10^{-16} \text{ W m}^2$	First radiation constant $c_1 = 2\pi h c^2$
c_2	$0,014\,387\,768\,77\dots \text{ m}\cdot\text{K}$	Second radiation constant $c_2 = hc/k$
e	$1,602\,176\,634 \cdot 10^{-19} \text{ A s}$	Elementary charge $e = \frac{2}{K_J R_K} = \Phi_0 G_0$
e	$2,718\,281\,828\,459\,045\,2\dots$	<i>Euler's</i> e .
F	$96\,485,332\,12\dots \text{ A s/mol}$	<i>Faraday</i> constant $F = e N_A$
F_α (10)	$2,502\,907\,875\,095\,892\,8\dots$	<i>Feigenbaum's</i> α and δ
F_δ	$4,669\,201\,609\,102\,990\,6\dots$	
G	$6,674\,30 \cdot 10^{-11} \text{ m}^3/\text{kg s}^2$ $(2,2 \cdot 10^{-5})$	<i>Newtonian</i> constant of gravitation; also known as γ from other authors. $\rightarrow \mathbf{GM}_\oplus$ for a more precise value.
G_0	$7,748\,091\,729\dots \cdot 10^{-5} / \Omega$	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</i> constant
g_e (15)	$-2,002\,319\,304\,360\,92$ $(1,8 \cdot 10^{-13})$	<i>Landé's</i> electron g-factor
$\mathbf{GM}_\oplus$	$3,986\,004\,418 \cdot 10^{14} \text{ m}^3/\text{s}^2$ (per definition)	<i>Geocentric gravitational constant</i> incl. the mass of the atmosphere (according to the Earth model WGS84) ⁷⁴
$g_\oplus$	$9,806\,65 \text{ m/s}^2$ (per definition)	Standard Earth acceleration (actual values vary from $9,780\,4 \text{ m/s}^2$ at the equator to $9,832\,2 \text{ m/s}^2$ at the poles)
h	$6,626\,070\,15 \cdot 10^{-34} \text{ J s}$	Planck constant
$\hbar$ (19)	$1,054\,571\,817\dots \cdot 10^{-34} \text{ J s}$	Reduced <i>Planck</i> constant $\hbar = h/2\pi$
i		$= \sqrt{-1}$ (not in <u>CONST</u> nor in CNST)

⁷⁴ $\rightarrow$ <https://earth-info.nga.mil/php/download.php?file=coord-wgs84> (207 pages).

Name	Numeric value and rel. SD	Remarks
k (20)	$1,380\,649 \cdot 10^{-23} \text{ J/K}$	Boltzmann constant $k = R/N_A$
K_J	$4,835\,978\,48\dots \cdot 10^{14} \text{ Hz/V}$	Josephson constant $K_J = 2e/h$
l_{PL}	$1,616\,255 \cdot 10^{-35} \text{ m}$ $(1,1 \cdot 10^{-5})$	Planck length $l_{PL} = t_{PL}c$
m_e	$9,109\,383\,713\,9 \cdot 10^{-31} \text{ kg}$ $(3,1 \cdot 10^{-10})$	Electron rest mass $\triangleq 511,00 \text{ keV}^{75}$
M_{Moon}	$7,349 \cdot 10^{22} \text{ kg}$ $(5 \cdot 10^{-4})$	Mass of the Moon
m_n (25)	$1,674\,927\,500\,56 \cdot 10^{-27} \text{ kg}$ $(5,1 \cdot 10^{-10})$	Neutron rest mass $\triangleq 939,57 \text{ MeV}$
m_n/m_p	$1,001\,378\,419\,46$ $(4,0 \cdot 10^{-10})$	Neutron to proton rest mass ratio
m_p	$1,672\,621\,925\,95 \cdot 10^{-27} \text{ kg}$ $(3,1 \cdot 10^{-10})$	Proton rest mass $\triangleq 938,27 \text{ MeV}$
m_{PL}	$2,176\,434 \cdot 10^{-8} \text{ kg}$ $(1,1 \cdot 10^{-5})$	Planck mass $m_{PL} = \sqrt{\hbar c/G} \approx 22 \mu\text{g}$
m_p/m_e	$1\,836,152\,673\,426$ $(1,7 \cdot 10^{-11})$	Proton to electron rest mass ratio
m_u (30)	$1,660\,539\,068\,92 \cdot 10^{-27} \text{ kg}$ $(3,1 \cdot 10^{-10})$	Atomic mass constant $\approx 10^{-3} \text{ kg}/N_A$
m_uc²	$1,492\,418\,087\,68 \cdot 10^{-10} \text{ J}$ $(3,1 \cdot 10^{-10})$	Energy equivalent of the atomic mass constant $\approx 931,49 \text{ MeV}$
m_μ	$1,883\,531\,627 \cdot 10^{-28} \text{ kg}$ $(2,2 \cdot 10^{-8})$	Muon (myon) rest mass $\triangleq 105,66 \text{ MeV}$
M_☉	$1,989\,1 \cdot 10^{30} \text{ kg}$ $(5 \cdot 10^{-5})$	Mass of the Sun (precisely: mass of the Solar system in total, → 126)

⁷⁵ For comparison: the rest mass of the (electron) neutrino is $0,02 \text{ eV} \leq m_\nu \leq 0,45 \text{ eV}$ (2025-04).

Name	Numeric value and <i>rel. SD</i>	Remarks
$M_{\oplus}$	$5,973\,6 \cdot 10^{24} \text{ kg}$ $(5 \cdot 10^{-5})$	Mass of the Earth. → $GM_{\oplus}$ for a more precise value.
N_A (35)	$6,022\,140\,76 \cdot 10^{23} / \text{mol}$	Avogadro's number
NaN	<i>Not a Number</i>	NaN covers poles as well as regions where a function result is not defined at all (infinities, on the other hand, are considered numeric by your <i>WP43</i> – see below). Unless <i>SPCRES</i> is set, non-numeric results lead to an error message thrown – NaN allows that functions written by you can return it instead (→ also p. 196).
p_0	$101\,325 \text{ Pa}$ (<i>per definition</i>)	Standard atmospheric pressure
R	$8,314\,462\,618 \dots \text{ J/mol K}$	Molar gas constant $R = k N_A$
r_e	$2,817\,940\,320\,5 \cdot 10^{-15} \text{ m}$ $(4,7 \cdot 10^{-10})$	Classical electron radius ⁷⁶ $r_e = \alpha^2 a_0$
R_K (40)	$25\,812,807\,45 \dots \Omega$	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_{∞}	$10\,973\,731,568\,157 / \text{m}$ $(1,1 \cdot 10^{-12})$	Rydberg constant $R_{\infty} = \alpha^2 m_e c / 2h$
$R_{\odot}$	$6,96 \cdot 10^8 \text{ m}$ $(5 \cdot 10^{-3})$	Mean radius of the Sun
$R_{\oplus}$	$6,371\,010 \cdot 10^6 \text{ m}$ $(5 \cdot 10^{-7})$	Mean radius of the Earth
Sa (45)	$6,378\,137\,0 \cdot 10^6 \text{ m}$ (<i>per definition</i>)	Semi-major axis ⁷⁷

⁷⁶ Actually, an electron is far smaller. Though specifying its radius makes no sense.

⁷⁷ ... of the Earth according to [WGS84](#). See the publication linked in f. 74 on p. 118, also for the uncertainties of **Sb**, **Se**², and **Se'**² overleaf.

Name	Numeric value and rel. SD	Remarks
Sb	$6,356\,752\,314\,2 \cdot 10^6 \text{ m}$	Semi-minor axis ⁷⁷ $S_b = S_a \left(1 - 1/S_f^{-1}\right)$
Se²	$6,694\,379\,990\,141 \cdot 10^{-3}$	1 st eccentricity squared ⁷⁷ $S_e^2 = 1 - \frac{S_a^2}{S_b^2}$
Se'²	$6,739\,496\,742\,276 \cdot 10^{-3}$	2 nd eccentricity squared ⁷⁷ $S_{e'}^2 = \frac{S_a^2 - S_b^2}{S_b^2}$
Sf⁻¹	298,257 223 563 (per def.)	Flattening factor ⁷⁷
T₀ (50)	273,15 K (per definition)	= 0°C, standard temperature
T_{PL}	$1,416\,784 \cdot 10^{32} \text{ K}$ $(1,1 \cdot 10^{-5})$	<i>Planck</i> temperature $T_P = \frac{c^2}{k} \sqrt{\frac{\hbar c}{G}} = \frac{M_P c^2}{k} = \frac{E_P}{k}$
t_{PL}	$5,391\,247 \cdot 10^{-44} \text{ s}$ $(1,1 \cdot 10^{-5})$	<i>Planck</i> time $t_{PL} = l_{PL}/c$
V_m	$0,022\,413\,969\,5... \text{ m}^3/\text{mol}$	Molar volume of an ideal gas at standard conditions $V_m = \frac{R T_0}{p_0} \approx 22,4 \text{ l/mol}$
Z₀	$376,730\,313\,412 \, \Omega$ $(1,6 \cdot 10^{-10})$	Characteristic impedance of vacuum
α (55)	$7,297\,352\,564\,3 \cdot 10^{-3}$ $(1,6 \cdot 10^{-10})$	Fine-structure constant $\alpha = \frac{e^2}{2\varepsilon_0 \hbar c} \approx \frac{1}{137}$
γ	$6,674\,30 \cdot 10^{-11} \text{ m}^3/\text{kg s}^2$ $(2,2 \cdot 10^{-5})$	<i>Newtonian</i> constant of gravitation; also known as G from other authors. → GM_⊕ for a more precise value.
γ_{EM}	$0,577\,215\,664\,901\,532\,9...$	<i>Euler's</i> (a.k.a. <i>Euler-Mascheroni</i>) constant
γ_p	$2,675\,221\,870\,8 \cdot 10^8 \text{ Hz/T}$ $(4,3 \cdot 10^{-10})$	Proton gyromagnetic ratio $\gamma_p = 4\pi \frac{\mu_p}{\hbar}$

Name	Numeric value and rel. SD	Remarks
$\Delta\nu_{\text{Cs}}$	9 192 631 770 Hz	Hyperfine transition frequency of ^{133}Cs
ϵ_0 (60)	$8,854\,187\,818\,8 \cdot 10^{-12} \frac{\text{As}}{\text{Vm}}$ ($1,6 \cdot 10^{-10}$)	Vacuum electric permittivity $\epsilon_0 = 1/\mu_0 c^2$ (note the so-called <i>Coulomb's</i> constant is just $1/4\pi\epsilon_0$)
λ_c	$2,426\,310\,235\,38 \cdot 10^{-12} \text{ m}$ ($3,1 \cdot 10^{-10}$)	<i>Compton</i> wavelengths of the electron $\lambda_c = h/m_e c$, neutron $\lambda_{cn} = h/m_n c$,
λ_{cn}	$1,319\,590\,903\,82 \cdot 10^{-15} \text{ m}$ ($5,1 \cdot 10^{-10}$)	and proton $\lambda_{cp} = h/m_p c$, respectively
λ_{cp}	$1,321\,409\,853\,60 \cdot 10^{-15} \text{ m}$ ($3,1 \cdot 10^{-10}$)	
μ_0	$1,256\,637\,061\,27 \cdot 10^{-6} \frac{\text{Vs}}{\text{Am}}$ ($1,6 \cdot 10^{-10}$)	Vacuum magnetic permeability
μ_B (65)	$9,274\,010\,065\,7 \cdot 10^{-24} \text{ J/T}$ ($3,1 \cdot 10^{-10}$)	<i>Bohr's</i> magneton $\mu_B = e\hbar/2m_e$
μ_e	$-9,284\,764\,691\,7 \cdot 10^{-24} \text{ J/T}$ ($3,1 \cdot 10^{-10}$)	Electron magnetic moment
μ_e/μ_B	$-1,001\,159\,652\,180\,46$ ($1,8 \cdot 10^{-13}$)	Ratio of μ_e to <i>Bohr's</i> magneton
μ_n	$-9,662\,365\,3 \cdot 10^{-27} \text{ J/T}$ ($2,4 \cdot 10^{-7}$)	Neutron magnetic moment
μ_p	$1,410\,606\,795\,45 \cdot 10^{-26} \text{ J/T}$ ($4,3 \cdot 10^{-10}$)	Proton magnetic moment
μ_u (70)	$5,050\,783\,739\,3 \cdot 10^{-27} \text{ J/T}$ ($3,1 \cdot 10^{-10}$)	Nuclear magneton $\mu_u = e\hbar/2m_p$

Name	Numeric value <i>and rel. SD</i>	Remarks
μ_μ	$-4,490\,448\,30 \cdot 10^{-26} \text{ J/T}$ <i>(2,2 · 10⁻⁸)</i>	Muon (myon) magnetic moment
π	3,141 592 653 589 793 2...	(not in <u>CONST</u> nor in CNST)
σ_B (72)	$5,670\,374\,41... \cdot 10^{-8} \frac{\text{W}}{\text{m}^2 \text{K}^4}$	<i>Stefan-Boltzmann</i> constant $\sigma_B = \frac{2 \pi^5 k^4}{15 h^3 c^2}$
Φ	1,618 033 988 749 894 8...	Golden ratio $\Phi = (1 + \sqrt{5})/2$
Φ_0	$2,067\,833\,848... \cdot 10^{-15} \text{ V s}$	Magnetic flux quantum $\Phi_0 = h/2e$
ω (75)	$7,292\,115 \cdot 10^{-5} \text{ rad/s}$ <i>(per definition)</i>	<i>Nominal mean angular velocity of the rotating Earth according to WGS84 (→ 118, f. 74).</i>
$-\infty$	$-\infty$	Note both are counted as numeric values here. For <u>SPCRES</u> , an error will be thrown for infinite results, → also p. 196.
∞	∞	

This array continues with:

Position	Contents	Remarks
78	0.	
79 – 141	Conversion factors	→ 152ff
142 – 237	Auxiliary constants	like $\frac{1}{e}$, $\ln 2$, $\ln 10$, $\frac{\sqrt{2}}{2}$, $\frac{\pi}{4}$, $\frac{\sqrt{3}}{2}$
238 – 266	Kitchen volume conv. factors	→ 152ff
267 – 296	30 <i>Lanczos</i> coefficients	
297 - 368	Auxiliary constants, also for angular modulo reduction and calculations	like π , $\frac{\pi}{2}$, $\frac{\pi}{4}$, 2π , $\ln 2$, $\ln 10$
369	Max. <i>date</i>	$3,1... \cdot 10^{19}$
370	Max. <i>time</i>	$3,6 \cdot 10^{19}$

In total, 371 constants are stored and may be called by CNST.⁷⁸ Each named constant (0 – 77) will appear in equations with its *name* (→ [OMS 5](#)), in programs with a heading # (e.g., # c, → [OMS 4](#)).

A few constants more (not stored in your WP43) follow for your convenience:

Numeric value	Remarks
343 m/s	Speed of sound in
1,2041 kg/m ³	Density of ... dry air at standard lab conditions (101,325 kPa, 20°C) ⁷⁹
28,965 · 10 ⁻³ kg/mol	Molar mass of
11,2 · 10 ³ m/s	Escape speed from Earth
1361 W/m ²	Solar constant, i.e. average incident power at the mean distance of the Earth to the Sun outside the atmosphere
13,8 · 10 ⁹ a	Age of the universe (= 4,35 · 10 ¹⁷ s)
46,5 · 10 ⁹ l.y.	Radius of the observable universe ⁸⁰ (≈ 4,40 · 10 ²⁶ m)

Some further constants are known to 1 or 2 digits only but may help you in quick order-of-magnitude (a.k.a. *ballpark*) estimates nevertheless:

⁷⁸ Your WP43 calculator holds all constants in its QSPI memory (→ [App. A](#)). CNST recalls 34 digits for each although many trailing digits may be zero.

⁷⁹ → https://en.wikipedia.org/wiki/Density_of_air#Dry_air

⁸⁰ For more information, look up https://en.wikipedia.org/wiki/Observable_universe (or https://de.wikipedia.org/wiki/Beobachtbares_Universum or https://fr.wikipedia.org/wiki/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 46.5×10⁹ *light years* printed above and the naïve assumption of 13.8×10⁹ *light years* for the radius; this picture was not found on the other sites checked).

So the circumference of the observable universe is 292 · 10⁹ l.y. ≈ 3 · 10²⁷ m. Note the uncertainty caused by the finite precision of your WP43 means just a deviation of 3 · 10⁻⁷ m = 0.3 μm in these 292 billion *light years*! Think about its relevance.

Radius of an atomic nucleus⁸¹

$$\sim 10^{-15} \text{ m} = 1 \text{ fm}$$

Radius of an atom⁸²

$$\sim 10^{-10} \text{ m} = 100 \text{ pm}$$

Radius of the star *Stephenson 2-18* comparable to the half-major axis of *Saturn's* orbit around the Sun (this picture of *Saturn* in its natural colors was shot by *Cassini* in 2004-10)

$$\sim 1,5 \times 10^{12} \text{ m} \approx 1,4 \text{ light hours (!)},$$



Radius of our home galaxy

$$\sim 10^5 \text{ l.y.} \approx 10^{21} \text{ m}$$

Hubble 'constant'⁸³

$$\sim 70 \frac{\text{km/s}}{\text{Mpc}} \approx 1/4,4 \cdot 10^{17} \text{ s}$$

Baryonic mass in the observable universe⁸⁴

$$\sim 10^{53} \text{ kg}$$

Number of stars in our home galaxy⁸⁵

$$\sim 4 \cdot 10^{11}$$

⁸¹ E.g., the (charge) radius of an hydrogen nucleus (i.e. a proton) is 0,840 9(4) fm, of helium 1,678 24(83) fm (data of 2025-06). Just in case you are looking for something really small.

Actually, helium atoms are used for *UHV* leak testing – since bare helium nuclei (a.k.a. α -particles) are more difficult to generate and store. Helium is the smallest atom nature provides (its radius is 28 pm), then comes *neon* with 58 pm. A hydrogen atom (32 pm) is slightly bigger than helium, but hydrogen comes as H_2 molecule only.

⁸² So the nucleus takes far less than a billionth of the volume of an atom. Electrons are even smaller. Thus, an atom is almost entirely empty space. On the other hand, our world as we watch and touch it every day is built of atoms. Think about it.

These facts also give some hand-waving arguments why cancer (tumor) therapy using heavy ion beams works at all (and often significantly better than using X-rays). Ions are charged atoms (here: atoms lacking some electrons). 'Heavy' ions for therapeutical use are actually carbon (^{12}C) although that's not really heavy but sufficient and carries some collateral advantages.

⁸³ Two significantly different values are known: 73 and 67,4. Discussions are ongoing in 2026 still. – Note this so-called constant does change over very long times actually.

⁸⁴ This is the mass of all common matter known (i.e. atoms, ions, molecules, etc.). Physicists think this can't be everything (for good reasons) and talk about '*dark matter*' – but this is not identified yet although a dark galaxy was found recently (2025-04).

⁸⁵ Assume our home galaxy fills a disk 1000 l.y. high, what's the average distance between its stars in 0th order approximation? Calculate yourself, then check f. 99.

<i>Number of cells in a human</i>	$\sim 10^{13}$
<i>Number of bacteria living in a human</i>	$\sim 10^{14}$
<i>Number of neuron connections in human brain</i>	$\sim 10^{14}$
Number of stars in observable universe	$\sim 10^{23}$
<i>Number of atoms in a human of 75 kg</i>	$\sim 7,2 \cdot 10^{27}$
<i>Number of legal board positions in chess</i>	$\sim 10^{43}$
Number of atoms in the Earth	$\sim 1,3 \cdot 10^{50}$
Number of atoms in the Sun ⁸⁶	$\sim 10^{57}$
Number of hydrogen atoms in observable universe ⁸⁷	$\sim 10^{80}$
<i>Number of legal board positions in the Japanese game Go</i>	$\sim 2 \cdot 10^{170}$

Note that all these quantities are far within the range of *real numbers* allowed on your WP43 (→ [App. B](#)). Physical constants are seldom known more precisely than twelve digits. Please take both facts into account when assessing very small numeric differences in your calculations as well as when talking about very large numbers (→ f. 80).

Just SI/ base units and SI/ derived units are employed for all these constants, no prefixes. This eases their use in subsequent calculations.

The next two pages are just for visual (and mental) calibration purposes.

⁸⁶ Take this number for the amount of atoms in our solar system as well – the matter beyond the Sun (0,144 8%) is negligible on this scale. So is our influence on ‘world history’, luckily for our solar system, our home galaxy, and our universe all the more. → picture overleaf.

At the bottom line, we should feel happy that we are allowed to gaze and investigate it – when we switch off our lighting. It is absolutely amazing how much mankind has found out about our universe as a whole in some 10 000 years of careful observations confined to our home position orbiting a minor star way out in one of the less important spiral arms of our home galaxy.

⁸⁷ Almost everything is hydrogen there – you can neglect the other material on this scale.



Here we are making 'world history'!

The arrows point to a tiny bright star 'under' (behind? in front of?) the bigger white one. This photomontage was published by [Bruno Bürgel](#) in his book [Der Mensch und die Sterne](#), Aufbau-Verlag, Berlin 1946.⁸⁸

⁸⁸ This is an enlarged clip of the original picture printed in [Bruno's](#) book. Not too much imagination is needed to guess that the year and place of its publication contributed significantly to the message. Still correct and relevant today, nevertheless.

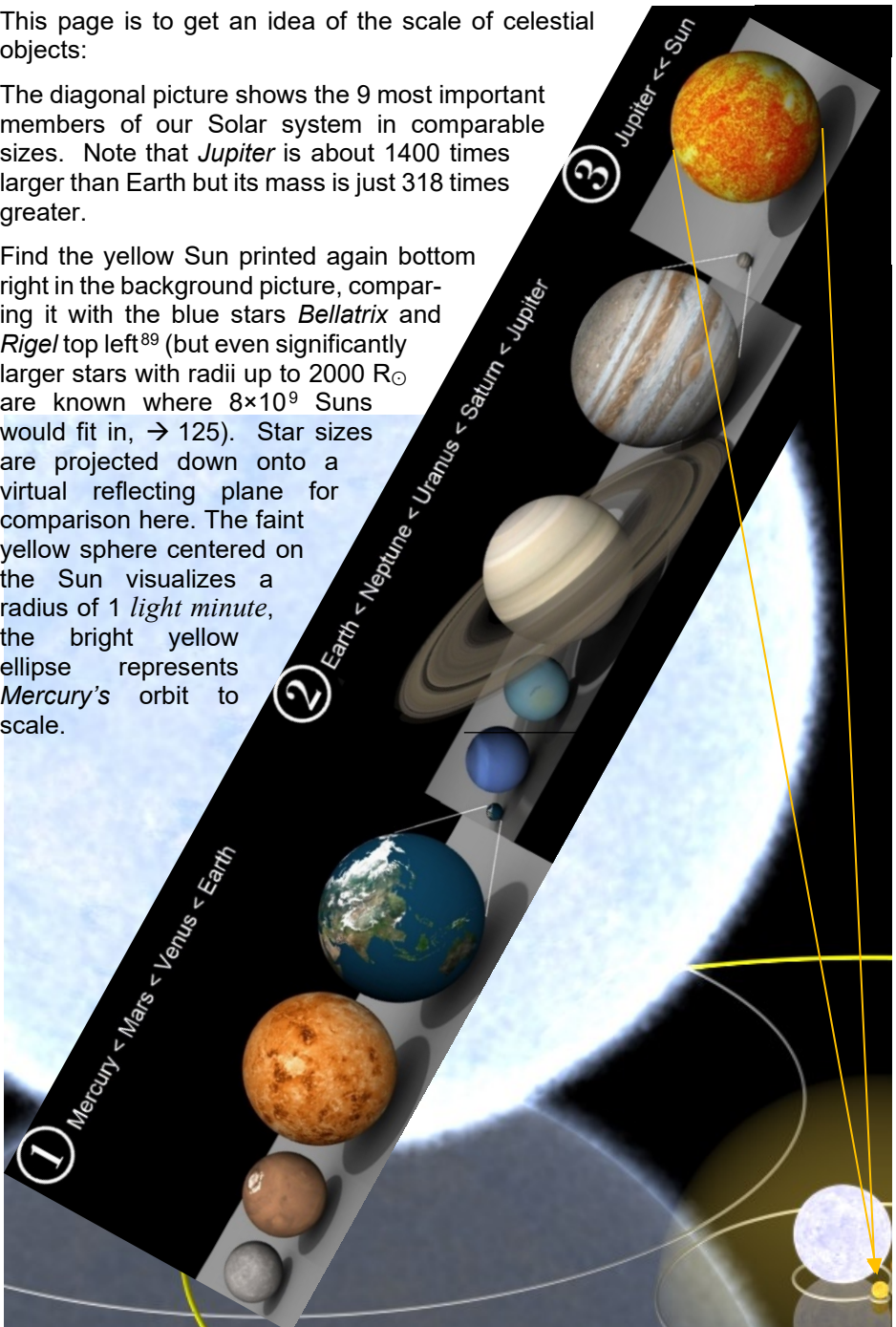
As stated elsewhere in these manuals already, the laws of nature apply inevitably – also to those who don't believe in them. Maybe this could teach us some modesty, though I doubt it will work for everybody – some are just too far off.

⁸⁹ The background of next page is a clip of a work by [Paul Stansifer](#).

This page is to get an idea of the scale of celestial objects:

The diagonal picture shows the 9 most important members of our Solar system in comparable sizes. Note that *Jupiter* is about 1400 times larger than Earth but its mass is just 318 times greater.

Find the yellow Sun printed again bottom right in the background picture, comparing it with the blue stars *Bellatrix* and *Rigel* top left⁸⁹ (but even significantly larger stars with radii up to $2000 R_{\odot}$ are known where 8×10^9 Suns would fit in, $\rightarrow$ 125). Star sizes are projected down onto a virtual reflecting plane for comparison here. The faint yellow sphere centered on the Sun visualizes a radius of 1 *light minute*, the bright yellow ellipse represents *Mercury's* orbit to scale.



System Flags

The *status bar*, especially its indicators (*SBI*), and the command **STATUS** return visible information about the system status of your *WP43* (→ [OMS 2](#) and [6](#)). For program control, machine readable status information is available via 43 named *system flags* as listed below, sorted following their order of appearance in the *status bar*.

If the third column is **green**, the corresponding *flag* may be modified by the user, else it can be written by the system only. *Flags* with their remarks printed on **light blue** are stored by **STOCFG** and may be recalled by **RCLCFG**. Some *system flags* can be also accessed via lettered *user flags*. *Startup default flag* status is indicated.

Purpose	<i>SBI</i>	<i>Flag</i>	Remarks
Date display	Date string	YMD, DMY, MDY	Indicating the date format chosen. The commands Y.MD , D.MY , and M.DY set the respective <i>flag</i> and clear the two others.
Time display	Time string	TDM24	Set for international 24h time display. Shows four digits for the year in the <i>status bar</i> date display. Clear for 12h time display: e.g., 1:23 will become 1:23am, 23:45 will become 11:45pm. Truncates the date display in the <i>status bar</i> to two digits for the year.
Complex results	C / R	GPXRES (I)	Set for allowing <i>complex</i> results for <i>real</i> input (e.g., $\sqrt{-1}$). Else an error will be thrown in such cases.
Polar notation	⊙ / ⊔	POLAR (X)	Set for polar display of <i>complex numbers</i> , clear for rectangular.
Fraction display 1	—	FRACT	Set for fraction display, clear for decimal display (a b/c sets FRACT, .d clears it). → next entries.
Fraction display 2	—	FRGSRN	Set for a verbose fraction display <u>showing register names</u> (e.g., $x = 3/5$), else clear.

Purpose	SBI	Flag	Remarks
Fraction kind 1	—	PROPFR	Set for <i>proper fractions</i> , clear for <i>improper fractions</i> (coming from decimal display, a b/c sets PROPFR, then it toggles it). PROPFR allows only <i>proper fractions</i> in display (e.g., 1 4/5 instead of 9/5). Any <i>reals</i> (with $ x < 10^6$) will be displayed as <i>proper fractions</i> according to the settings of DENANY, DENFIX, and D.MAX.
Fraction kind 2	→ OMS 2	DENANY	Set if <u>any</u> denominator up to D.MAX is legal (D.MAX is indicated in the <i>status bar</i>). ⁹⁰ For D.MAX given, this is the most precise way of displaying a decimal number as a fraction. If clear then DENFIX is evaluated.
Fraction kind 3	→ OMS 2	DENFIX	Set if D.MAX is the one and only denominator allowed. Clear if the denominator may be an integer factor of D.MAX. ⁹¹ Evaluated only for DENANY.
Carry	c or o	CARRY (C)	Reflects the status of the <i>Carry</i> bit).
Overflow	o or o	OVERFL (B)	Reflects the status of the <i>Overflow</i> bit.
Leading zeros	—	LEAD-0 (L)	Set for leading zeros turned on in <i>short integers</i> of bases 2, 4, 8, and 16. ⁹²
Matrix grow mode	grow / wrap	GROW	Set (e.g., by M.GROW) if matrices may grow, else clear (e.g., by M.WRAP). Lit in the <i>status bar</i> as long as a matrix is open for editing. → J+ in the <i>IOI</i> and OMS 3 . ⁹³

⁹⁰ E.g., for D.MAX = 8 and DENANY, denominators 1, 2, 3, 4, 5, 6, 7, and 8 are allowed.

⁹¹ If D.MAX = 60 and DENFIX (and DENANY), this will allow for denominators 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, and 60. Note 60 was a sacred number in ancient *Babylon* – hence our hours are divided in 60 *minutes* even today. Actually, Babylonian astronomers, businessmen, and computers calculated base 60.

⁹² Works like *flag 3* of **HP-16C**.

Purpose	SBI	Flag	Remarks
Payment due (for TVM)	END / BEG	ENDPMT	Set for payments at the end of each period, clear for beginning. Controlled by ENDP and BEGINP , → OMS 6 and OMA 3 . Lit in the <i>status bar</i> as long as <u>TVM</u> is open. ⁹³
Alpha	A or α	ALPHA	Set for <i>A/M</i> , else clear.
Upper case	A / α	αCAP	Set for capital letters, clear for lower case.
Running timer		RUNTIM	Set if the timer is running in background. Note your <i>WP43</i> will <u>not</u> turn off automatically with the timer running – care for reliable power supply.
Running I/O		RUNIO	Set if I/O is in progress.
Tracing	—	TRACE ()	Set if the 'print line' is in tracing mode. Else each print must be triggered explicitly.
User mode		USER	Set if your <i>WP43</i> is in user mode.
Low battery		LOWBAT	Set if battery voltage is low (< 2.5 V, → ?BATT and App. G).
USB power		USB	Set if connected to a USB power source. ⁹⁴
Fast function	—	FASTFN	If set , your <i>WP43</i> calculates faster without using internal float of extended (39-digit) precision. ⁹⁴ This is a trade-off between speed and precision. It affects $\sqrt{x}$, $\sqrt[3]{x}$, $\ln$, $\ln(1+x)$, $\lg$, $\lg_b$, $\log_x y$, e^x , e^{x-1} , 10^x , 2^x , y^x , $\sin$, $\cos$, $\tan$, $\arcsin$, $\arccos$, $\arctan$, $\sinh$, $\cosh$, $\tanh$, arsinh , arcosh , and artanh .
Special results	—	SPGRES ()	Set for allowing 'special results' of calculations (i.e. $\pm\infty$ and NaN).

⁹³ Compare also GROW in the [HP-42S OM](#), p. 213.

⁹⁴ Speed of execution increases in following order and coarse relation: battery power supply and FASTFN (25), battery and FASTFN (70), USB power supply and FASTFN (80), USB and FASTFN (200).

Purpose	SBI	Flag	Remarks
Stack size	8 / 4	SSIZE8	Set for 8, clear for 4 <i>stack registers</i> . Note <i>register</i> contents will remain unchanged if SSIZE8 is modified (also by RCLCFG).
Radix mark	—	DECIM.	Set for a decimal point, clear for a decimal comma.
Multiplication symbol	—	MULT×	Set for multiplication symbol ×, clear for ·. Note the symbols used for matrices and in equations are fixed.
Overflow from ALL	—	ALLENG (A)	Set if <i>real numbers</i> exceeding the range displayable in ALL or FIX shall be shown in ENGINEER's format, clear for SCIENTIFIC .
Automatic execution	—	AUTXEQ	With AUTXEQ set (e.g., from previous session) when you turn on your <i>WP43</i> , program execution begins at the current program line and AUTXEQ is cleared. ⁹⁵
Data entry	—	NUM.IN, ALP.IN	Set for numeric or alphanumeric (text) entry, respectively. ⁹⁶
Error handling	—	IGN1ER	If set , your <i>WP43</i> ignores one arbitrary error and clears IGN1ER the. ⁹⁷ The operation causing this error will not be executed.
Automatic OFF	—	AUTOFF	Set if automatic shutdown after 600 s is enabled. Else your <i>WP43</i> will remain ON until you will turn it off manually or battery voltage will drop below the lower limit (→ ?BATT and App. G). ⁹⁸ Note <i>WP43</i> will <u>not</u> turn off automatically with a program or the timer running.
Beeper	—	QUIET	Set for disabled beeper.

⁹⁵ Works like *flag* 11 of [HP-42S](#).

⁹⁶ These work like *flags* 22 and 23 of [HP-42S](#).

⁹⁷ Works like *flag* 25 of [HP-42S](#).

⁹⁸ Works like *flag* 44 of [HP-42S](#) so far.

Purpose	SBI	Flag	Remarks
Automatic stack lift	—	ASLIFT	Cleared by ENTER↑, CLX, M.EDI, M.EDIN, M.←, M.↑, M.→, M.↓, Σ+, and Σ- (→ App. H and OMS 1). Set by all other functions.
Integrating	—	INTING	Set while the <i>Integrator</i> is running.
Solving	—	SOLVING	Set while the <i>Solver</i> is computing a root.
Variable menu	—	VMDISP	Set while the <i>variable menu</i> is displayed (→ VARMNU and OMS 4).

Note the *status bar* will be updated every minute or after a keystroke. Thus, e.g., settings by SETDAT or SETTIM, although recorded immediately, may take up to one minute to appear in the *status bar* date-and-time string.

Tests

All commands provided under [TEST](#) are compiled and classified below for your convenience (*num. comp.* represents *numeric comparison*):

Name(s)	Legal input data types	Num. comp.	Remarks
CPX?	All	Yes	true for DT 3 or 9 <u>and</u> $\text{Im}(x) \neq 0$
INT?	All	Yes	true for $\text{FP}(x) = 0$
EVEN? ODD? PRIME?	All	Yes	true for $\text{FP}(x) = 0$ <u>and</u> ... (→ 39, 68, and 71)
FP?	All	Yes	true for $\text{FP}(x) \neq 0$
REAL?	All	Yes	true for DT 2 or 8 <u>and</u> $\text{FP}(x) \neq 0$
MATR?	All	—	true for DT 8 or 9
M.SQR?	All	—	true if x is a square matrix
NaN? SPEC?	All	Yes	true if x contains special values (→ 66 and 87)

<i>Name(s)</i>	Legal input <i>data types</i>	Num. comp.	Remarks
STRI?	All	—	true for <i>DT 7</i>
x=? x≠?	All	Yes	→ 98
x<? x≤? etc.	1,2,4,5,6,7,10	Yes	
x≈?	2,3,4,5,8,9	Yes	
x=+0? x=-0?	1, 2, 10	Yes	
LEAP?	2, 6	Yes	→ 53
CONVG? <u>s</u>	—	Yes	→ 31
ENTRY?	—	—	Tests the entry flag, → 39
KEY? <u>d</u>	—	—	Tests for key press, → 52
LBL? <u>label</u>	—	—	Does this <i>label</i> exist (→ 53)?
TOP?	—	—	true in a top level routine, → 91

Nonprogrammable Commands and Keys

Operations whose *names* are printed on **violet** or **black** in the *IOI* can't be programmed (but *menu items* can be called). Also answering questions your *WP43* may ask is impossible automatically.

All other *items* may be used in programs. Operations triggered by **P/R**, **α**, **↶**, **≡Δ**, **▲**, **≡▽**, and **▼**, however, may not be echoed on the screen (neither in *RUM* nor *PEM*) as the other commands are.

⁹⁹ Solution of f. 85: In this very simple model, our galaxy has a volume of $3.14 \times 10^{13} \text{ l.y.}^3$. So each star could take 78.5 l.y.^3 , meaning a sphere of 5.3 l.y. diameter. Actually, star density is far higher in the galactic center and far lower besides the spiral arms.

SECTION 2: MENUS AND CATALOGS

Due to the many *items* your WP43 features, most of them are stored in *menus* (as introduced in [OMS 1](#)). Besides commands, also numeric constants, *characters* (as in the *alpha menus*), digits, *submenus*, *variables*, and program *labels* may be contained in *menus*.

If no specific *menu* was called by you, either MyMENU or (in *PEM*) MyPFN or (in *AIM*) Myq are displayed.

You may switch *menus* (except *catalogs* – see below) easily just calling another *menu* directly from the *menu* you are using – no need to **EXIT** first. The last (*sub-*) *menus* opened are stored on a 4-level *menu stack*. Each **EXIT** of a (*sub-*) *menu* pops the youngest entry from this *stack*; opening a new (*sub-*) *menu* pushes this new entry on it. So wherever you are, the fourth **EXIT** will return to a screen with MyMENU, MyPFN, or Myq displayed at latest.

Catalogs are a special kind of *menus*: their contents are sorted alphabetically. Your WP43 provides 17 *catalogs*:

- CATALOG'FCNS (by far the largest *catalog* in *SDC*),
- CATALOG'MENUS ,
- CATALOG'CHARS'αINTL ,
- the two *submenus* of CATALOG'PROGS ,
- the 11 *submenus* of CATALOG'VARS , and
- CONST (being identical to CATALOG'CONST).

Let's look at the mother of all *catalogs* first.

One to Find and Rule Them All – the CATALOG

Pressing **CATALOG** opens a very special *menu*: it comprises all the user visible *items* defined on your WP43. Many of them are sorted alphabetically: these we call *cataloged*. *Cataloged items* may be accessed quickly as demonstrated in next chapter.

A survey of the various CATALOG branches at *startup* is presented below. Note they appear here in reverse order compared to the *LCD*, taking care of common top-down reading habits:

	F1	F2	F3	F4	F5	F6	Remarks
CATALOG:	FCNS	CONST	CHARS	PROGS	VARS	MENUS	top branches
FCNS:	°C→°F	°F→°C	10 ^x	1COMPL	1/x	2COMPL	catalog of all the >700 functions provided; while in A/M, only ← and → will be shown ¹⁰⁰
	2 ^x	$\sqrt[3]{x}$	ABS	ACOS	ActUSB	ac→ha	
	ac _{US} →ha	AGM	AGRAPH	ALL	AND	arccos	
	...	...	#B				
CONST:	a	a ₀	...				constants, → 116ff
CHARS:	αINTL	A...Ω		αMATH	Myα	α·	character branches
αINTL:	A	Ã	Á	Â	Ã	Ä	catalog of all the 99 Latin letters provided, → 149
	...						
	...	...	Ž				
αMATH:	<	≤	=	≈	≥	>	mathematical operators and symbols, → 150
	...						
A...Ω:	A	B	Γ	Δ	E	Z	all Greek letters, → 150
	...						
α•:	!	;	:	'	“	✓	punctuation marks etc., → 151
	...						
PROGS:	AllOp's	Bairsto	ECA	Fact	FreeF	FreeFp	global labels defined (empty unless test_pgms.bin was copied); to be filled with your creations
	...						
VARS:	L.INTS	S.INTS	REALS	CPXS	ALL	MATRS	see below
	DATES	TIMES	ANGLES	CONFIG		STRING	

¹⁰⁰ The light green background signals *catalogs* in this Section only.

	F1	F2	F3	F4	F5	F6	Remarks
V.ALL:	ACC	ADM	D.MAX	FV	GRAMOD	ISM	catalogs of all variables specified, with ALL pointing to V.ALL being the union of all the other submenus of VARS. The variables are placed according to their DT – most of these submenus are empty in SDC. They will be filled and grow with your creations.
	i%/a	Mat_A	Mat_B	Mat_X	n _{PER}	PER/a	
	PMT	PV	REALDF	↑Lim	↓Lim	#DEC	
ANGLES:							
CPXS:							
CONFIG:							
DATES:							
L.INTS:	ADM	D.MAX	GRAMOD	ISM	REALDF	#DEC	
MATRS:	Mat.A	Mat.B	Mat.X				
REALS:	ACC	FV	i%/a	n _{PER}	PER/a	PMT	
	PV	↑Lim	↓Lim				
STRING:							
S.INTS							
TIMES:							

MENUS:	ADV	ANGLES	A:	Binom:	BIT	Cauch:	(sub-) menus currently defined – see above and below for fix menu contents. User menus will be inserted as new entries here.
	CHARS	CLK	CLMy...	CLR	CONST	CPX	
	CPXS	DATES	DISP	EQN	EXP	Expon:	
	...						
	...	↔					
ADV	...						all existing (sub-) menus unless mentioned above already, → 140ff for fix contents.
ANGLES	...						
A:	...						
Binom:	...						
...							

The branches **FCNS**, **CONST**, and **CHARS** are fixed since all functions, constants, and *characters* on your WP43 are predefined; **MENUS**, **PROGS**, and the submenus of **VARS** are **expandable** since you may create *items* of these kinds (→ [OMS 7](#)).

Opening CATALOG will display these six top level labels, point-

FCNS	CONST	CHARS	PROGS	VARS	MENUS
------	-------	-------	-------	------	-------

ing to the *submenus* containing all the functions, constants, *characters*, programs, *variables*, and *menus* defined at execution time. Choosing one of these *submenus* will display its first *view* of *items* (primary,  - and -shifted, as applicable). Pressing **FCNS**, for instance, will call CATALOG'FCNS showing its first *view* as pictured here:

Select any *function* by pressing the corresponding *softkey* in

	ac _{US} →ha	AGM	AGRAPH	ALL	AND	arccos	
	2 ^x	$\sqrt[3]{x}$	ABS	ACOS	ActUSB	ac→ha	
	°C→°F	°F→°C	10 ^x	1COMPL	1/x	2COMPL	

this *catalog*. – You can as well choose any *menu* via CATALOG'MENUS, an arbitrary *real variable* via CATALOG'VARS'REALS, etc.

Within any *catalog*,  will advance just by 6 *items* per keystroke (and  will go back by 6); but you may as well use the alphabetic search method explained in next chapter. **EXIT** will just leave the open (sub-) menu without doing anything.

Although all the functions provided on your *WP43* are stored in CATALOG'FCNS, most of them can be accessed easier through other *menus* (→ the chapter after next). Remember that each and every command on your *WP43*, as well as the keystrokes calling it are explained for your reference in the *IOI* on pp. 20ff; also the predefined *menus* and *variables*, as well as the constants and *system flags* are listed therein. The *IOI* may point you to further material.

Rapid Access to Cataloged Items

Feel free to browse a *catalog* like any other *menu* just using  and . Though in CONST and major parts of CATALOG (i.e. **FCNS**, **CONST**, **MENUS**, **PROGS**, **CHARS** & **INTL**, and the *submenus* of **VARS**) you may reach your target distinctly faster taking advantage of the *alphabetic search method* demonstrated below.¹⁰¹

¹⁰¹ This method is known from *WP 34S* since 2011.

Example:

Let's assume $x = 1\,237\,572\,753\,017$. You are looking for the next prime number. You know you need the function **NEXTP** but have forgotten in which particular *menu* it is stored. Then simply enter:

1	User input	CATALOG FCNS																		
	Output	See the first <i>view</i> in this <i>catalog</i> ; ¹⁰² <i>AIM</i> is on.																		
		<table><tr><td>$ac_{us} \rightarrow ha$</td><td>$ac_{us} \rightarrow m^2$</td><td>AGM</td><td>AGRAPH</td><td>ALL</td><td>AND</td></tr><tr><td>2^x</td><td>$\sqrt[3]{x}$</td><td>ABS</td><td>ACOS</td><td>$ac \rightarrow ha$</td><td>$ac \rightarrow m^2$</td></tr><tr><td>$^{\circ}C \rightarrow ^{\circ}F$</td><td>$^{\circ}F \rightarrow ^{\circ}C$</td><td>$10^x$</td><td>1COMPL</td><td>$1/x$</td><td>2COMPL</td></tr></table>	$ac_{us} \rightarrow ha$	$ac_{us} \rightarrow m^2$	AGM	AGRAPH	ALL	AND	2^x	$\sqrt[3]{x}$	ABS	ACOS	$ac \rightarrow ha$	$ac \rightarrow m^2$	$^{\circ}C \rightarrow ^{\circ}F$	$^{\circ}F \rightarrow ^{\circ}C$	10^x	1COMPL	$1/x$	2COMPL
$ac_{us} \rightarrow ha$	$ac_{us} \rightarrow m^2$	AGM	AGRAPH	ALL	AND															
2^x	$\sqrt[3]{x}$	ABS	ACOS	$ac \rightarrow ha$	$ac \rightarrow m^2$															
$^{\circ}C \rightarrow ^{\circ}F$	$^{\circ}F \rightarrow ^{\circ}C$	10^x	1COMPL	$1/x$	2COMPL															
2	User input	1 st <i>character</i> of the <i>item</i> desired																		
	Output	e.g., N See the <i>view</i> starting with the first <i>item</i> starting with this <i>character</i> ¹⁰³																		
		<table><tr><td>NOR</td><td>Norml_p</td><td>Norml_Δ</td><td>Norml_Δ</td><td>Norml⁻¹</td><td>NOT</td></tr><tr><td>NEIGHB</td><td>NEXTP</td><td>nmi$\rightarrow$km</td><td>nmi$\rightarrow$m</td><td>Nm$\rightarrow$lbft</td><td>NOP</td></tr><tr><td>NAND</td><td>NaN?</td><td>NBin_p</td><td>NBin_Δ</td><td>NBin_Δ</td><td>NBin⁻¹</td></tr></table>	NOR	Norml _p	Norml _Δ	Norml _Δ	Norml ⁻¹	NOT	NEIGHB	NEXTP	nmi $\rightarrow$ km	nmi $\rightarrow$ m	Nm $\rightarrow$ lbft	NOP	NAND	NaN?	NBin _p	NBin _Δ	NBin _Δ	NBin ⁻¹
NOR	Norml _p	Norml _Δ	Norml _Δ	Norml ⁻¹	NOT															
NEIGHB	NEXTP	nmi $\rightarrow$ km	nmi $\rightarrow$ m	Nm $\rightarrow$ lbft	NOP															
NAND	NaN?	NBin _p	NBin _Δ	NBin _Δ	NBin ⁻¹															
3	User input	Here it is already! ¹⁰⁴ Press the corresponding <i>softkey</i> to call it e.g., F2 .																		

¹⁰² ... unless you visited the same *catalog* before – then it will open showing its last view you looked at. The remaining procedure will stay unchanged though.

¹⁰³ This search is case independent (i.e. specifying **A** will find **a** as well). Nevertheless **A** and **a** remain different letters. You can search for Greek letters via prefix **☐**, e.g., **☐** + **A** for **α**. Also other *characters* can be specified ($\rightarrow$ *virtual keyboard* on p. 168). Watch the sorting order printed on p. 14.

Note the *items* in the *catalog* you search may be displayed at other positions in the *menu* view than the ones you see when browsing using **▼** or **▲** only.

¹⁰⁴ You may have entered more *characters* in step 2 – but after 3 *seconds* or after pressing **▼** or **▲**, whatever occurs first, the search string will be reset. Then you may continue browsing or start a new search string.

If the *character* or string specified is not found in this *catalog* then the first *item* following in sorting order ($\rightarrow$ 14) will be shown. If there is no such *item* then the last *item* in this *catalog* will be displayed.

Output	If no parameter input is required, your <i>WP43</i> jumps directly to step 4. Else it turns to <i>TAM</i> and waits for your input:
3a User input	Enter the necessary parameter(s).
4 Output	Your <i>WP43</i> executes the command, calls the program, recalls the constant, <i>RoV</i>, inserts the command, digit or <i>character</i> selected. It displays the result then e.g., 1 237 572 153 089 ... and exits this <i>catalog</i> returning to the mode set before entering it. 

At the bottom line, this means that you can ...

- call any function of over 700 by **CATALOG** **FCNS** + 4 keystrokes max. if you know its first two *characters*;
- recall any constant of over 80 by **CONST** + 3 keystrokes max. if you know its first *character*;
- insert any Latin letter provided by **CATALOG** **CHARS** **αINTL** (or **α** **A** in *AIM*) + 3 keystrokes max.

αINTL is the one and only *catalog* staying open until you **EXIT** it explicitly. Picking an *item* from any other *catalog* (e.g. from **CONST**) will exit this automatically.

Further Menus and Their Contents

In the table following, all the 38 *menus* provided on the keyboard (incl. those accessible in *AIM*, three user *menus*, and two *catalogs*) are listed in alphabetical order with their number of *views* and *submenus* contained:

Menu	Views	Submenus	Views	Remarks
ADV	1	fidx	1	
BIT	3	—		
CLK	2	—		
CLR	1	1	1	
CONST	5	—		Submenu of CATALOG , → 116ff for its content
CPX	1	—		
DISP	2	—		
EQN	1	5	1 each	
EXP	1	—		
FIN	1	TVM	1	
FIT	2	ASSESS , HISTOG , PLOT	1 each	
FLAG	1	—		
INFO	2	—		
INTS	3	—		
I/O	1	—		
LOOP	1	—		
MODE	1	—		
MATX	2	M.EDIT , M.SIMQ	1 each	
MyMENU	1	—		3 <i>menus</i> to be filled by you – feel free to create more (→ OMS 7)
MyPFN	1	4	1 - 2	
Mya	1	4	1 - 6	
PART	1	—		
PRINT 	1	—		
PROB	2	14	1 each	Probability distribu- tions
P.FN	1	P.FN2	1	

Menu	Views	Submenus	Views	Remarks
STAT	2	FIT and HISTOG	2	→ FIT above
STK	1	—		
TEST	2	—		
TRI	1	—		
U→	2	8	1 - 3	Conversion kinds
X.FN	2	Ellipt and Orthog	1 each	Function kinds
αINTL 	6	—		Submenu of CATALOG'CHARS
αMATH 	3	—		
α.FN	1	—		
α• 	3	—		
Σ	2	—		
4→	2	—		

Below, these *menus* are listed in full detail, usually like you see them on display. Different *views* within the same *menu* are separated by dashed lines, *submenus* by double lines in this table. Note that individual *labels* may appear in [more than one menu](#) and also on the keyboard.

Menu	(F1)	(F2)	(F3)	(F4)	(F5)	(F6)	Remarks
ADV	PGMSLV	SLVC	f''(x)			PGMINT	advanced operations, → OMS 5
	SOLVE	SLVQ	f'(x)	Π_n	Σ_n	f_{dx}	
	f_{dx}	Draw	ACC	↓Lim	↑Lim	∫	
BIT	SB	BS?	#B	FB	BC?	CB	contains all the <i>Boole's</i> and bit operations (1 st two views) and settings (3 rd) of HP-16C and WP 34S
	NAND	NOR	XNOR	MIRROR			
	AND	OR	XOR	NOT	MASKL	MASKR	
	LJ				RJ	ASR	
	SL	RL	RLC	RRC	RR	SR	
	A	B	C	D	E	F	
	1COMPL	2COMPL	UNSIGN	SIGNMT		WSIZE	

Menu	F1	F2	F3	F4	F5	F6	Remarks
CLK	J→D	D→J		DAY	MONTH	YEAR	date and time functions (first view) and settings (2 nd view)
	DATE	→DATE	DATE→	WDAY	TIME	x→DATE	
	J/G						
	SETTIM	TDISP	SETDAT	D.MY	Y.MD	M.DY	
CLR	CLall	CLBKUP	DelVall		DelMall	CLLCD	commands for clearing and deleting
	CLREGS	CLPall	CLVall	CLFall	CLMall	CLSTK	
	CLΣ	CLP	CLCVAR	CF	CLMy...	CLX	
	RESET		CLKEYS	DelITM	CLMENU		
	CLMy...			CLMyM	CLMyP	CLMyα	
CONST	catalog of constants						→ 116ff
CPX	CX→RE	RE→CX		Im			complex functions
	dot	cross	UNITV	Re	conj	Re↔Im	
DISP	SDL	SDR			RDP	RSD	display rounding and shifts, formats and settings, mostly for reals
	FIX	SCI	ENG	ALL	ROUNDI	ROUND	
	GAP	HIDE	?HIDE	RANGE	?RANGE	DSTACK	
	CHINA	EUROPE	INDIA	JAPAN	UK	USA	
EQN	DELETE						equations (→ OMS 5)
	NEW	EDIT	f''	f'	∫f	Solver	
EQ.EDI	←	()	^	:	=	→	Equation Editor
Solver					Draw	Calc	list the names of all variables in current equation and more
∫f						Draw	
f'							
f''							
EXP	sinh	arsinh	cosh	arcosh	tanh	artanh	exponential, logarithmic, and hyperbolic functions
	$\sqrt[3]{x}$	$\sqrt[3]{y}$		ln 1+x	e^x-1		
	x^3	y^x	log _x y	lb x	2 ^x	x ²	

Menu	(F1)	(F2)	(F3)	(F4)	(F5)	(F6)	Remarks
FIN	%	%MRR	%T	%Σ	%+MG	TVM	financial functions and settings (OMS 6). 2 nd row of TVM is for recalling
TVM	Begin					End	
	RCL n_p	i%/a	per/a	PV	PMT	RCL FV	
	n_{PER}	i%/a	per/a	PV	PMT	FV	
FIT	CLΣ	ASSESS				PLOT	2D sample statistics, regression, and MSA. See p. 147 for HISTOG . The 2 nd view is for selecting curve fit model(s)
	Σ-		s_{xy}	cov		HISTOG	
	Σ+	L.R.	s(a)	r	$\hat{x}$	$\hat{y}$	
	GaussF	CauchF		BestF			
	ParabF	HypF	RootF				
	LinF	ExpF	LogF	PowerF		OrthoF	
ASSESS	NXTFIT	ZOOM					→ OMS 2
PLOT	CENTRL	s_{mi}					
FLAG						CLFall	
	FS?S	FS?C	FS?F	FC?F	FC?S	FC?C	
	SF	FS?	FF	STATUS	FC?	CF	
INFO	WHO	VERS	?DIM	±∞?	αPOS	αLENG	system information and one non-binary test (±∞?)
	?LocR	?DISK	ULP	NEIGHB	?SDIGS	?BATT	
	?SSIZE	?MEM	?RM	?ISM	?WSIZE	?KTYP	
	?RANGE	?HIDE	?J/G	?VOL		?BestF	
INTS	DBL /	DBLR	DBL ×	^MOD	ceil	GCD	digits for <i>short integers</i> with bases > 10, integer settings and operations
	IDIV	RMD	MOD	×MOD	floor	LCM	
	A	B	C	D	E	F	
	1COMPL	2COMPL	UNSIGN	SIGNMT		WSIZE	
	→BIN	→DEC			→OCT	→HEX	
I/O	BEEP	TONE	BUZZ	PLAY	VOL-	VOL+	backup, data exchange and signalling
	LoadSS	LoadV	LoadR	LoadΣ	READP		
	LoadAll	LoadPr	LoadEq	DUMP	WRITEP	SAVEST	

Menu	F1	F2	F3	F4	F5	F6	Remarks
LOOP	DEC					INC	
	DSE	DSZ	DSL	ISE	ISZ	ISG	
MATX	ENORM		STOEL	RCLEL	PUTM	GETM	matrix operations (<i>items</i> are sorted almost as in <i>HP-42S</i>)
	dot	cross	UNITV	DIM	INDEX	EDITN	
	NEW	$[M]^{-1}$	$ M $	$[M]^T$	SIMEQ	EDIT	
	CMAX	CMIN	M.QR	FIND	EIGVAL	EIGVEC	
	RNORM	RSUM	M.LU	?DIM	C↔C	R↔R	
	I+	I-	STOIJ	RCLIJ	J-	J+	
M.EDIT	INSR		DELR		WRAP	GROW	Matrix Editor (as in <i>HP-42S</i>)
	←	↑	OLD	GOTO	↓	→	
M.SIMQ	Mat A	Mat B				Mat X	for systems of linear equations
MODE	DMCP						mode settings; DMCP calls the menu of SM
		GRAD	RM		SETSIG	DENMAX	
	SF	DEG	RAD	MIL	MUL π	CF	
MyMENU							will pop up in RUM beyond AIM ¹⁰⁵
MyPFN							will pop up in PEM beyond AIM ¹⁰⁵
	LOOP	TEST			P.FN	P.FN2	
My α							will pop up in AIM ¹⁰⁵
	←	α MATH	α .	A... Ω	α INTL	→	
PART							
					Re	Im	
	IP	FP	sign	MANT	EXPT	DECOMP	

¹⁰⁵ ... as long as no other *menu* is called (→ *OMS 7*). MyMENU is empty at startup. All three My... *menus* can be filled with up to 18 items each and edited by you.

Menu	F1	F2	F3	F4	F5	F6	Remarks
PRINT ()	 x	 r	 Σ	 ADV		 PROG	like WP 34S
	 STK	 REGS	 USER	 TAB	 #	 CHAR	and similar to HP-42S PRINT
				 WIDTH	 DLAY	 MODE	commands

PROB		NBin:	Geom:	Hyper:	Binom:	Poiss:	combinations, permutations, random number generators, and 14 probability distributions. Selecting one (e.g., NBin) opens a sub-menu featuring four entries for its PDF (or PMF), its CDF, error probability, and quantile function
		LgNrm:	Cauch:		Expon:	Logis:	
		Norml:	t:	C _{y_x}	P _{y_x}	F:	
		RAN#	SEED	RANI#		lnΓ	
	Binom:	Binom _p		Binom _Δ	Binom _Δ		Binom ⁻¹
	Cauch:	Cauch _p		Cauch _Δ	Cauch _Δ		Cauch ⁻¹
	Expon:	Expon _p		Expon _Δ	Expon _Δ		Expon ⁻¹
	F:	F _p (x)		F _Δ (x)	F _Δ (x)		F ⁻¹ (p)
	Geom:	Geom _p		Geom _Δ	Geom _Δ		Geom ⁻¹
	Hyper:	Hyper _p		Hyper _Δ	Hyper _Δ		Hyper ⁻¹
	LgNrm:	LgNrm _p		LgNrm _Δ	LgNrm _Δ		LgNrm ⁻¹
	Logis:	Logis _p		Logis _Δ	Logis _Δ		Logis ⁻¹
	NBin:	NBin _p		NBin _Δ	NBin _Δ		NBin ⁻¹
	Norml:	Norml _p		Norml _Δ	Norml _Δ		Norml ⁻¹
	Poiss:	Poiss _p		Poiss _Δ	Poiss _Δ		Poiss ⁻¹
	t:	t _p (x)		t _Δ (x)	t _Δ (x)		t ⁻¹ (p)
	Weibl:	Weibl _p		Weibl _Δ	Weibl _Δ		Weibl ⁻¹
	χ ² :	χ ² _p (x)		χ ² _Δ (x)	χ ² _Δ (x)		(χ ²) ⁻¹

P.FN	PUTK	PROMPT	AVIEW	CNST	TICKS	REM	additional programming functions (avoided a 2-view menu here to allow for easier program browsing)
	R-CLR	R-COPY	R-SORT	R-SWAP	LocR	PopLR	
	INPUT	MVAR	VarMNU	END	PAUSE	P.FN2	
P.FN2	AGRAPH	PIXEL	POINT	ERR	MSG	XEQ.SKIP	
	BACK	CASE	SKIP	NOP	SDL	SDR	
	MENU	KEYG	KEYX	CLMENU	EXITall	RTN.SKIP	

Menu	(F1)	(F2)	(F3)	(F4)	(F5)	(F6)	Remarks
STAT	CLΣ	$\bar{x}_G$	ε	ε_p	ε_m	FIT	1D and 2D statistics of independent samples (→ also FIT on p. 144)
	Σ-	$\bar{x}_w$	s_w	σ_w	s_{mw}	HISTOG	
	Σ+	$\bar{x}$	s	σ	s_m	SUM	
		$\bar{x}_H$					
		$\bar{x}_{RMS}$	x_{min}	x_{max}	Δx		
		MEDIAN	MAD	Q_3-Q_1	%ile		
HISTOG	HISTOX	HISTOY	↓BIN	nBINS	↑BIN	HPLOT	→ OMS 2
HPLOT	HNORM						
STK							stack operations
	x↔	y↔	z↔	t↔	↔	DROPy	
TEST	ENTRY?	KEY?	LBL?	STRI?	CONVG?	TOP?	binary tests.
	INT?	EVEN?	ODD?	PRIME?	LEAP?	FP?	
	x<?	x≤?	x=?	x≠?	x≥?	x>?	
	SPEC?	NaN?		M.SQR?			
	x=+0?	x=-0?	x≈?	MATR?	CPX?	REAL?	
TRI	sinh	arsinh	cosh	arcosh	tanh	artanh	trigonometric & hyperbol. functions (→ EXP).
	sinc	sincπ				atan2	
	sin	arcsin	cos	arccos	tan	arctan	
U→	°C ↔ °F				V:	A:	unit conversions (→ 152ff).
	E:	P:	Time:	F&p:	m:	x:	
	dB ↔ power ratio		dB ↔ field ratio		Nm ↔	lbf·ft	
A:	ha ↔ acre		m² ↔ ha		ha ↔ acre _{US}		units of area; 'kitchen units' (→ 154ff)
	m² ↔ acre		m² ↔ mū		m² ↔ acre _{US}		
E:	J ↔ cal		J ↔ Btu		J ↔ Wh		units of energy
F&p:			Pa ↔ mmHg				units of force and pressure
	Pa ↔ in.Hg		Pa ↔ torr		Pa ↔ atm		
	N ↔ lbf		Pa ↔ bar		Pa ↔ psi		

Menu	(F1)	(F2)	(F3)	(F4)	(F5)	(F6)	Remarks
m:	kg $\leftrightarrow$ stone		kg $\leftrightarrow$ short cwt		kg $\leftrightarrow$ troy oz.		units of mass
	kg $\leftrightarrow$ lb.		kg $\leftrightarrow$ cwt		kg $\leftrightarrow$ oz.		
	kg $\leftrightarrow$ liǎng				kg $\leftrightarrow$ jīn		
	kg $\leftrightarrow$ ton		kg $\leftrightarrow$ short ton		kg $\leftrightarrow$ carat		
	g $\leftrightarrow$ oz.		g $\leftrightarrow$ troy oz.		g $\leftrightarrow$ carat		kitchen units
P:	W $\leftrightarrow$ hp _E		W $\leftrightarrow$ hp _{UK}		W $\leftrightarrow$ hp _M		units of power
Time:	s $\leftrightarrow$ day				s $\leftrightarrow$ year		units of time
V:	m ³ $\leftrightarrow$ floz _{UK}				m ³ $\leftrightarrow$ floz _{US}		units of volume
	m ³ $\leftrightarrow$ gal _{UK}		m ³ $\leftrightarrow$ barrel		m ³ $\leftrightarrow$ gal _{US}		
	ml $\leftrightarrow$ floz _{UK}		ml $\leftrightarrow$ pint _{UK}		l $\leftrightarrow$ gal _{UK}		Imperial kitchen units
	ml $\leftrightarrow$ tsp _{UK}		ml $\leftrightarrow$ tbs _{UK}		ml $\leftrightarrow$ cup _{UK}		
	ml $\leftrightarrow$ tea spoon _I		ml $\leftrightarrow$ table spoon _I		ml $\leftrightarrow$ cup _I		international kitchen units
			l $\leftrightarrow$ quart		l $\leftrightarrow$ gal _{US}		US customary kitchen units
	ml $\leftrightarrow$ floz _{US}		ml $\leftrightarrow$ pint _{US}				
	ml $\leftrightarrow$ tsp _{US}		ml $\leftrightarrow$ Tbsp _{US}		ml $\leftrightarrow$ cup _{US}		
x:	m $\leftrightarrow$ inch		m $\leftrightarrow$ point		m $\leftrightarrow$ yard		units of length
	m $\leftrightarrow$ mile		m $\leftrightarrow$ nmi.		m $\leftrightarrow$ foot		
	m $\leftrightarrow$ au		m $\leftrightarrow$ l.y.		m $\leftrightarrow$ pc		
	m $\leftrightarrow$ fath _m						
	m $\leftrightarrow$ chǐ		m $\leftrightarrow$ cùn		m $\leftrightarrow$ fēn		
	m $\leftrightarrow$ lǐ		m $\leftrightarrow$ yīn		m $\leftrightarrow$ zhàng		
	km $\leftrightarrow$ mile		km $\leftrightarrow$ nmi.		m $\leftrightarrow$ surv. foot _{US}		kitchen units
	mm $\leftrightarrow$ inch		mm $\leftrightarrow$ point				

X.FN	J _y (x)	lnβ	lnΓ	max	min	NEXTP	advanced mathematical functions like Beta, Bessel, etc.
	erfc	FACTOR	FIB	g _d	g _d ⁻¹	I _{xyz}	
	AGM	B _n	B _n *	erf	Ellipt	Orthog	
	Γ _q	γ _{xy}	Γ _{xy}	ζ(x)	(-1) ^x		
	W _m	W _p	W ⁻¹	Y _y (x)	β(x,y)	Γ _p	

Menu	F1	F2	F3	F4	F5	F6	Remarks
Ellipt	$\psi(u,m)$			$F(\varphi,m)$	$E(\varphi,m)$	$Z(\varphi,m)$	elliptic functions and integrals
	sn	cn	dn	$K(m)$	$E(m)$	$\Pi(n,m)$	
Orthog	H_{np}						orthogonal polynomials
	H_n	L_m	$L_{m\alpha}$	P_n	T_n	U_n	

α INTL (A)	A a	À à	Á á	Â â	Ã ã	Ä ä	[α] <i>catalog</i> of all Latin letters provided, printed here in reverse vertical order following common reading habits. All letters (except ß) will change when case is switched in A/M – note you will see the individual letters displayed in either case only at one time. You can reach each of these 99 letters by ≤3 keystrokes via alphabetical access as explained on pp. 138f. You might be even faster using your computer keyboard in the <i>Simulator</i>
	Å å	Æ æ	Ā ā	Ă ă	Ą ą	Ȑ ȑ	
	C c	Ç ç	Ć ć	Č č	D d	Đ đ	
	Ď ě	Ð ð	E e	È è	É é	Ê ê	
	Ë ë	Ē ē	Ě ě	Ê ê	Ę ę	Ĕ ĕ	
	F f	G g	Ğ ğ	H h	I i	İ ı	
	Í í	Î î	Ĭ ĭ	Ī ī	Ĵ ĵ	Į į	
	Ĳ ĳ	J j	K k	L l	Ł ł	Ĺ ľ	
	Ł ł	M m	N n	Ñ ñ	Ń ń	Ň ň	
	O o	Ò ò	Ó ó	Ô ô	Õ õ	Ö ö	
	Ø ø	Ȯ ȯ	Ȱ ȱ	Œ œ	P p	Q q	
	R r	Ř ř	Ŕ ŕ	S s	Ś ś	Ş ş	
	Š š	ß	T t	Ț ț	Ť ť	U u	
	Ù ù	Ú ú	Û û	Ü ü	Ũ ũ	Ū ū	
	Ŭ ŭ	Ų ų	Ų ų	V v	W w	Ẃ ẃ	
	X x	Y y	Ý ý	Ŷ ŷ	Ỳ ỳ	Z z	
	Ž ž	Ż ż	Ž ž				

Menu	F1	F2	F3	F4	F5	F6	Remarks
αMATH (≈)	$\cdot / \times$ ¹⁰⁶	$\div //$ ¹⁰⁷	$\int$	∞	∞	∞	[α] for inserting comparison symbols, parentheses and brackets, as well as more mathematical and related symbols. You can reach every <i>character</i> herein by ≤ 3 keystrokes.
	{	[	(	)	]	}	
	<	≤	=	≈	≥	>	
	$\bar{x}$	$\bar{y}$	$\hat{x}$	$\hat{y}$	$\bar{x}$	$\bar{y}$	
	$\frac{4}{5}$	$\frac{6}{7}$	$\frac{8}{9}$	$\frac{10}{11}$	$\frac{12}{13}$	$\frac{14}{15}$	
	$\neg$	$\wedge$	$\vee$	$\neq$	$ $	$\&$	
	$\pm$	$\wedge$	$\top$	-1	$\hbar$	$\mathbb{Z}$	
	$\odot$	$\ominus$	$\oplus$	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{Q}$	
	$:=$	$\hat{=}$	$\equiv$	$\ll$	$\gg$	$\mathbb{E}$	

α.FN	FBR						dedicated operations on <i>text strings</i> plus the font browser
					αLENG	αPOS	
	$x \rightarrow \alpha$	αRL	αRR	αSL	αSR	$\alpha \rightarrow x$	

A...Ω ¹⁰⁸	A α	B β	Γ γ	Δ δ	E ε	Z ζ	[α] Greek letters. The keyboard grants direct access to 24 of them. Note the two kinds of lower case Σ. → αINTL for more.
	H η	Θ θ	I ι	K κ	Λ λ	M μ	
	N ν	Ξ ξ	O ο	Π π	P ρ	Σ σ	
	ς	T τ	Υ υ	Φ φ	X χ	Ψ ψ	
	Ω ω	ά	έ	ή	ί	ϊ	
	İ ĩ	ó	ú	ÿ ü	û	ώ	

¹⁰⁶ With *SDC*, the multiplication dot is found here and the cross is called via   in *AIM*. For *MULTX*, this dot is called via   in *AIM* and the cross via .

¹⁰⁷ With *SDC*, the obelus is found here and the solidus is called via   in *AIM*. For *MULTX*, the obelus is called via   in *AIM* and the solidus via . An unshifted division colon is in [α.](#)

¹⁰⁸ Access this catalog via [CATALOG'CHARS'A...Ω](#). Note the classical Greek alphabet goes alpha, beta, gamma, delta, épsilon, zeta, eta, thêta, iôta, kappa, lambda, mÿ, nÿ, xī, ó-mikron, pī, rhō, sigma, tau, ýpsilon, phī, chī, psī, and ô-mega (ancient Greek H, Θ, and Y are taught to be pronounced like Finnish ÄÄ, T, and Y – Finnish Y sounds like French U or German Ü, unlike English U! Think of *Nils Holgersson's* goose *Yksi* (followed by *Kaksi*, *Kolme*, *Neljä*, *Viisi*, and *Kuusi* for obvious reasons)).

Menu	(F1)	(F2)	(F3)	(F4)	(F5)	(F6)	Remarks
 	\$	€	%	&	£	¥	[α] for inserting punctuation marks, currency symbols, arrows, and further special characters.
	!	;	:	'	"	✓	
	⌚	🕒	🕒	🕒	🕒	*	
	«	»		🕒	🕒	*	
	←	↑	↕	↗	↓	→	
	”	”	...	-	•		

Σ	Σx ² y	Σxlny	Σx ² lny	Σlny/x		Σylnx	all the sums necessary and accumulated for the statistics in <u>STAT</u> .
		Σlnx	Σln ² x	Σlnxy	Σln ² y	Σlny	
	n	Σx	Σx ²	Σxy	Σy ²	Σy	
	Σx ³	Σx ⁴					
	Σx ² /y	Σ ¹ /x	Σ ¹ /x ²	Σx/y	Σ ¹ /y ²	Σ ¹ /y	

↔	D→R	R→D	Mπ→R	R→Mπ	D→D.MS	D.MS→D	angular conversions, → 163f.
	DEG→	RAD→	MULπ→	MIL→	GRAD→	D.MS→	
	→DEG	→RAD	→MULπ	→MIL	→GRAD	→D.MS	
	D→MIL	MIL→D	MIL→R	R→MIL			

The *menus* below will pop up for parameter input when calling...

	(F1)	(F2)	(F3)	(F4)	(F5)	(F6)	Remarks
... STO or RCL	...EL	...IJ					→ OMS 1
	Config	Stack			min	max	
	→	VAR	X	Y	Z	T	
... x ₁ , y ₁ , etc.	→	VAR	X	Y	Z	T	→ OMS 1
... x<?, etc.	0.	1.					→ OMS 1
	→	VAR	X	Y	Z	T	

Greek pronunciation changed significantly in 2500 years: nowadays, **H**, **I**, **Y**, and **OI** all sound like **I** (i.e. like **EE** for English speakers).

	F1	F2	F3	F4	F5	F6	Remarks
... SF, CF, etc.	→	SYS.FL	X	Y	Z	T	→ OMS 1
... XEQ, GT0, etc.	→	PROG	X	Y	Z	T	→ OMS 4
... DELITM				PROGS	VARS	MENUS	→ OMS 7

PROG will display all *global labels* present in *RAM* at execution time. PROGS and MENUS are identical to the respective CATALOG submenus (→ 135ff). VAR and VARS point to CATALOG'VARS'V.ALL; VAR and SYS.FL will display only writeable *items* for write operations but all for read.

Unit Conversions

Your *WP43* features 22 angular conversions provided in 4 → (→ 151) and 142 unit conversions in U → (→ 147f). The latter *menu* is subdivided into various branches as shown above. If local units will be dropped once for *SI*, just 18 conversions will do beyond the angular ones.

The seven fundamental *SI* units are *kelvin*, *meter*, *kilogram*, *second*, *ampere*, *mol*, and *candela*.¹⁰⁹ Beyond these and products, powers, and ratios of them, knowing the symbols and names of some *SI derived* (a.k.a. *compound*) *units* is helpful:

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>	$1 \text{ Pa} = 1 \text{ N/m}^2 = 1 \text{ kg/m s}^2$

¹⁰⁹ *Candela* measures *luminous intensity*.

TNG: *Luminous intensity* = *Lichtstärke*. Umseitig bedeuten *luminous flux* = *Lichtfluss* oder *Lichtstrom* und *luminance* = *Leuchtdichte*.

Quantity	Unit	Symbol and formula
Energy	<i>joule</i>	$1 \text{ J} = 1 \text{ N m} = 1 \text{ kg m}^2/\text{s}^2$
Power	<i>watt</i>	$1 \text{ W} = 1 \text{ J/s}$
Electric potential, voltage	<i>volt</i>	$1 \text{ V} = 1 \text{ W/A} = 1 \text{ J/A s}$
Charge	<i>coulomb</i>	$1 \text{ C} = 1 \text{ A s}$
Capacitance	<i>farad</i>	$1 \text{ F} = 1 \text{ C/V} = 1 \text{ A s/V}$
Resistance	<i>ohm</i>	$1 \Omega = 1 \text{ V/A}$
Conductance	<i>siemens</i>	$1 \text{ S} = 1 \text{ A/V}$
Magnetic flux	<i>weber</i>	$1 \text{ Wb} = 1 \text{ V s}$
Magnetic flux density	<i>tesla</i>	$1 \text{ T} = 1 \text{ Wb/m}^2 = 1 \text{ V s/m}^2$
Inductance	<i>henry</i>	$1 \text{ H} = 1 \text{ Wb/A} = 1 \text{ V s/A}$
Frequency	<i>hertz</i>	$1 \text{ Hz} = 1/\text{s}$
Radioactivity	<i>becquerel</i>	$1 \text{ Bq} = 1 \text{ decay/s}$
Absorbed radiation 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, also knowing the symbols and names of SI prefixes is beneficial. 36 orders of magnitude are covered below: ¹¹⁰

¹¹⁰ *TN for US-Americans:* SI unit symbols and prefixes are case sensitive – like A, H, K, N, M, P, and S. Hence, e.g., a unit ‘mah’ doesn’t exist at all.

Prefix letter	Name	Factor	Exemplary units used
h	<i>hecto-</i>	10^2	hl, ha, hPa
k	<i>kilo-</i>	10^3	many
M	<i>mega-</i>	10^6	many
G	<i>giga-</i>	10^9	many
T	<i>tera-</i>	10^{12}	Tb, TW, TWh
P	<i>peta-</i>	10^{15}	PFlops
E	<i>exa-</i>	10^{18}	Eb ¹¹¹

Prefix letter	Name	Factor	Exemplary units used
d	<i>deci-</i>	10^{-1}	dm, dl, dtex
c	<i>centi-</i>	10^{-2}	cm, cl
m	<i>milli-</i>	10^{-3}	many
μ	<i>micro-</i>	10^{-6}	many
n	<i>nano-</i>	10^{-9}	many
p	<i>pico-</i>	10^{-12}	ps, pF
f	<i>femto-</i>	10^{-15}	fs, fm
a	<i>atto-</i>	10^{-18}	as

All the conversions featured in U → are explained in alphabetical order below.¹¹² There, numeric values are either exact (printed on white) or

Fun fact: Long distances are measured in *kilometer*, then in thousands or millions of km – Mm and Gm are simply not used. For astronomic distances, *astronomic units* and *light years* are applied – some 10^{11} l.y. suffice for all we know.

TN: People knowing ancient Greek and Latin enjoy advantages here. Prefixes beyond the ones listed here are defined though hardly used. Using exponents of ten is easier (as long as no engineers are around).

¹¹¹ In computer science, there are also kib (1 kib = 1024 *bytes*) and MiB (1 MiB = 1 048 576 *bytes*) not used here.

¹¹² Within U →, all conversions are displayed as pairs with the *SI* units on the left side. Note that most of these conversions begin or end with *SI base* units, mainly m and kg, or directly derived units like Pa or m³. This makes subsequent conversions most easy.

TN: ATTENTION: Conversions of so-called '*kitchen units*' deviate on request of some *British Imperial* natives (applies also to *acres*, *carat*, *miles*, and *points*): There you must apply the necessary powers of 10 before continuing converting. We separated those conversions from the coherent ones in every branch of U → as far as possible.

The *British Imperial* units in this table cover the most popular in the *UK* and its ex-colonies and territories. The *US* volume units are the customary ones for liquids (they differ for dry volumes for historical reasons). In *SI*, on the other hand, a volume of, e.g., 1 *cubic meter* stays 1 m³, be it evacuated or filled with gas, steam, water, mercury, or solid iron. Just *litre* is used exclusively for liquids, gases, or empty space. (Being at it, nobody talks of *kiloliter* like *Merriam-Webster* does instead of *cubic meter*; be aware there are more unusual 'metric' quantities in their 'metric system' table of 2026 still – seems they are just too far off.)

rounded to six significant digits (your *WP43* uses more precise values wherever applicable). In this chapter, commas are used as radix marks for better visibility, multiplication dots for space reasons.

Conversion	Calculation	Remarks	Branch
$^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$	$\cdot 1,8 + 32$	$\vartheta [^{\circ}\text{C}] = (T [\text{K}] - 273,15)$ and $\tau [^{\circ}\text{F}] = (t [^{\circ}\text{R}] - 491,67)$	
$^{\circ}\text{F} \rightarrow ^{\circ}\text{C}$	$- 32) / 1,8$		
acre $\rightarrow$ ha	$\cdot 4,046\ 86$	This <i>acre</i> is based on the ‘ <i>inter-national foot</i> ’ ($\rightarrow$ below); 640 <i>acres</i> = 1 <i>square mile</i>	A:
acre $\rightarrow$ m ²	$\cdot 40\ 468,6$		
acre _{US} $\rightarrow$ ha	$\cdot 4,046\ 87$		
acre _{US} $\rightarrow$ m ²	$\cdot 40\ 468,7$	This <i>acre</i> is based on the <i>US survey foot</i> ($\rightarrow$ below)	
atm $\rightarrow$ Pa	$\cdot 101\ 325$	Atmosphere	F&p:
au $\rightarrow$ m	$\cdot 1,495\ 98 \cdot 10^{11}$	Astronomic unit; 1 a.u. $\approx$ 150 million km (= 150 Gm)	x:
barrel $\rightarrow$ m ³	$\cdot 0,158\ 987$	(US) barrel of oil, abbr. bbl	V:
bar $\rightarrow$ Pa	$\cdot 100\ 000$	1 mbar = 1 hPa	F&p:
Btu $\rightarrow$ J	$\cdot 1\ 055,06$	British thermal unit	E:
cal $\rightarrow$ J	$\cdot 4,186\ 8$	Calory	
carat $\rightarrow$ g	/5		m:
carat $\rightarrow$ kg	/5 000		

The Chinese units are a selection taken from the market standard *Shizhi* (市制) valid for mainland China. Note that 10 000 *fēn* = 1000 *cùn* = 100 *chǐ* = 10 *zhàng* = 1 *yīn* – a reasonable decimal system. Also note that units carrying the same names but used in Hongkong and Macao may differ significantly, as well as those used in other states in this region (e.g., Malaysia, Singapore, Taiwan, or Vietnam) with a major Chinese population.

Tibet used to have its own independent weights and measures before 1959 – then it was not inhabited by a significant number of Chinese either.

Myanmar has got its own units not covered here.

Conversion	Calculation	Remarks	Branch
chǐ → m	/ 3	'Chin. foot'; 1 市尺 := 10 cùn	x:
cùn → m	/ 30	'Chin. inch'; 1 市寸 := 10 fēn	
cup _I → mℓ	· 250	1 intl. (metric) cup := 250 ml	V:
cup _{UK} → mℓ	· 284,131	1 cup _{UK} := 10 fluid ounces (UK)	
cup _{US} → mℓ	· 236,588	1 cup _{US} := 8 fluid ounces (US)	
cwt → kg	· 50,802 4	1 hundredweight (UK) := 112 lbs	m:
day → s	· 86 400		Time:
dB → field ratio	$10^{R_{dB}/20}$	Decibel	▼
dB → power ratio	$10^{R_{dB}/10}$		
fēn → m	/ 300	1 市分 := 0,1 cùn	x:
field ratio → dB	$20 \lg(a_1/a_2)$	Also known as <i>amplitude</i> , <i>voltage</i> or <i>current ratio</i> . a_2 = reference	▼
floz _{UK} → m ³	· 28,413 1 · 10 ⁻⁶	1 fluid ounce := 1/16 pint	V:
floz _{UK} → mℓ	· 28,413 1		
floz _{US} → m ³	· 29,573 5 · 10 ⁻⁶		
floz _{US} → mℓ	· 29,573 5		
fathm → m	· 1,828 8	1 fathom := 2 yards	x:
foot → m	· 0,304 8	So-called 'international feet' (of 1959); 1 foot := 12 inches	
gal _{UK} → ℓ	· 4,546 09	Gallon; 1 Imperial gallon := 8 Imp. pints 1 US liquid gallon := 231 cubic inches	V:
gal _{UK} → m ³	· 4,546 09 · 10 ⁻³		
gal _{US} → ℓ	· 3,785 42		
gal _{US} → m ³	· 3,785 42 · 10 ⁻³		

Conversion	Calculation	Remarks	Branch
g → carat	· 5	<i>Gram</i> ; 1000 g = 1 kg	m:
g → oz.	/ 28,349 5		
g → troy oz.	/ 31,103 5		
ha → m²	· 10 000	1 a [<i>are</i>] := 100 m ² , 1 ha [<i>hectare</i>] := 100 a, 1 km ² [<i>square kilometer</i>] = 100 ha = 10 ⁶ m ²	 A:
ha → acre	/ 4,04686		
ha → acre_{US}	/ 4,04687		
hp_E → W	· 746	<i>Electric horsepower</i>	P:
hp_M → W	· 735,499	So-called <i>metric horsepower</i> ¹¹³	
hp_{UK} → W	· 745,700	<i>British Imperial horsepower</i>	
inch → m	· 0,025 4	1 <i>inch</i> := 1000 <i>mil</i> ; but watch the angular <i>mil</i> !	x:
inch → mm	· 25,4		
in.Hg → Pa	· 3 386,39	<i>Inch of mercury</i>	F&p:
jīn → kg	· 0,5	‘ <i>Chin. pound</i> ’; 1 市斤 := 10 liǎng	m:
J → Btu	/ 1 055,06	<i>Joule</i>	E:
J → cal	/ 4,186 8		
J → Wh	/ 3 600		
kg → carat	· 5 000	<i>Kilogram</i>	m:
kg → cwt	/ 50,802 4		
kg → jīn	/ 0,5		
kg → lb.	/ 0,453 592		
kg → liǎng	/ 0,05		
kg → oz.	/ 0,028 349 5		
kg → short cwt	/ 45,359 2		
kg → short ton	/ 907,185		
		1 t [(<i>metric</i>) <i>ton</i>] := 1000 kg	

¹¹³ **TN:** This hp equals ch in French and PS in German, for example.

Conversion	Calculation	Remarks	Branch
kg → stone	/ 6,350 29	← this is <u>not</u> a metric <i>ton</i> !	
kg → ton	/ 1 016,05		
kg → troy oz.	/ 0,031 103 5		
km → mile	/ 1,609 344	<i>Kilometer</i>	x:
km → nmi	/ 1,852		
lbf → N	· 4,448 22	<i>Pound force</i>	F&p:
lbf·ft → Nm	· 1,355 82	Torque: <i>Pound force times foot</i>	
lb. → kg	· 0,453 592	<i>Pound; 1 lb := 16 ounces</i>	m:
liǎng → kg	· 0,05	'Chin. ounce' 1 市两 := 0,1 jīn	
lǐ → m	· 500	'Chin. mile' 1 市里 := 15 yīn	x:
l.y. → m	· 9,460 73 · 10 ¹⁵	<i>Light year</i> ¹¹⁴	
ℓ → gal_{UK}	/ 4,546 09	1 litre := 1 dm ³ = 10 ⁻³ m ³ . <i>Litre</i> is only used to measure liquids, gases, or empty space. For everything else, take m ³ and its fractions.	 V:
ℓ → gal_{US}	/ 3,785 42		
ℓ → quart	/ 0,946 353		
m² → acre	/ 40 468,6	<i>Square meter</i>	 A:
m² → acre_{US}	/ 40 468,7		
m² → ha	/ 10 000		
m² → mǔ	· 0,001 5		
m³ → barrel	/ 0,158 987	<i>Cubic meter.</i> Feel free to use <i>litre</i> for smaller volumes of liquids, gases, or empty space (compare above).	 V:
m³ → fluid oz_{UK}	/ 28,413 1 · 10 ⁻⁶		
m³ → fluid oz_{US}	/ 29,573 5 · 10 ⁻⁶		
m³ → gal_{UK}	/ 4,546 09 · 10 ⁻³		
m³ → gal_{US}	/ 3,785 42 · 10 ⁻³		
m³ → quart	/ 1,136 52 · 10 ⁻³		

¹¹⁴ Note this is defined by the *IAU* as the distance light travels within one [Julian](#) year, i.e. 365,25 days.

Conversion	Calculation	Remarks	Branch
mile → km	· 1,609 344	1 (statute) mile := 1760 yards	x:
mile → m	· 1 609,344		
mℓ → cup^I	/ 250	<i>Millilitre</i> ; 1 mℓ = 1 cm ³	V:
mℓ → cup_{UK}	/ 284,131		
mℓ → cup_{US}	/ 236,588		
mℓ → floz_{UK}	/ 28,413 1		
mℓ → floz_{US}	/ 29,573 5		
mℓ → pint_{UK}	/ 568,261		
mℓ → pint_{US}	/ 473,176		
mℓ → table spoon^I	/ 15		
mℓ → tbsp_{UK}	/ 14,207		
mℓ → Tbsp_{US}	/ 14,787		
mℓ → tea spoon^I	/ 5		
mℓ → tsp_{UK}	/ 3,552		
mℓ → tsp_{US}	/ 4,929		
mmHg → Pa	· 133,322	→ Pa.	F&p:
mm → inch	/ 25,4	<i>Millimeter</i>	x:
mm → point	/ 0,352 778		
mǔ → m²	/ 0,001 5	平方亩 (píng fāng mǔ), used for pieces of land in particular.	A:
m → au	/ 1,495 98·10 ¹¹	<i>Meter</i>	x:
m → chǐ	· 3		
m → cùn	· 30		
m → fathm	/ 1,828 8		
m → fēn	· 300		

Conversion	Calculation	Remarks	Branch
m → foot	/ 0,304 8	<i>Meter</i>	x:
m → inch	/ 0,025 4		
m → lř	/ 500		
m → l.y.	/ 9,460 73·10 ¹⁵		
m → mile	/ 1 609,344		
m → nmi	/ 1 852		
m → pc	/ 3,085 68·10 ¹⁶		
m → point	/ 3,527 78·10 ⁻⁴		
m → survey foot_{us}	/ 0,304 801		
m → yard	/ 0,914 4		
m → yřn	· 0,03		
m → zhǎng	· 0,3		
nmi → km	· 1,852	<i>Nautical mile</i>	
nmi → m	· 1 852		
Nm → lbf·ft	/ 1,355 82	<i>Newton meter (for torques)</i>	▼
N → lbf	/ 4,448 22	<i>Newton</i>	F&p:
oz. → g	· 28,349 5	<i>Ounce</i>	m:
oz. → kg	· 0,028 349 5		
Pa → atm	/ 101 325	<i>Pascal</i> ; 1 Pa = 1 N/m ² 1 hPa = 1 mbar For all real-world applications (i.e. within experimental errors), <i>torr</i> are equivalent to <i>millimeter of Hg</i> . Possible differences are merely calculatory. See also the <i>NIST</i> document linked in f. 115 (→ 162f)	F&p:
Pa → bar	/ 100 000		
Pa → in.Hg	/ 3 386,39		
Pa → mmHg	/ 133,322		
Pa → psi	/ 6 894,76		
Pa → torr	/ 133,322		
pc → m	· 3,085 68·10 ¹⁶	<i>Parsec</i>	x:

Conversion	Calculation	Remarks	Branch
pint_{UK} → mℓ	· 568,261	1 <i>Imp. pint</i> := $\frac{1}{8}$ <i>gallon (UK)</i>	V:
pint_{US} → mℓ	· 473,176	1 <i>US liq. pint</i> := $\frac{1}{8}$ <i>liq. gallon</i>	
point → m	· 3,527 78 · 10 ⁻⁴	1 (typographical) <i>point</i> := $\frac{1}{72}$ <i>inch</i>	x:
point → mm	· 0,352 778		
power ratio → dB	$10 \lg(\frac{p_1}{p_2})$	p ₂ = reference power	▼
psi → Pa	· 6 894,76	<i>Pound per square inch</i>	F&p:
quart → ℓ	· 0,946 353	1 <i>US liq. quart</i> := $\frac{1}{4}$ <i>liq. gallon</i>	V:
quart → m³	· 9,463 53 · 10 ⁻⁴		
short cwt → kg	· 45,359 2	1 <i>sh. hundredweight (US)</i> := 100 lbs	m:
short ton → kg	· 907,185	1 <i>short ton (US)</i> := 2000 lbs	
stone → kg	· 6,350 29		
survey foot_{US} → m	· 0,304 801	1 <i>US survey foot</i> := $\frac{1200}{3937}$ m	x:
s → day	/ 86 400		Time:
s → year	/ 31 556 952		
table spoon_I → mℓ	· 15	1 <i>international (metric) tablespoon</i>	V:
tbsp_{UK} → mℓ	· 14,207		
Tbsp_{US} → mℓ	· 14,786 8		
tea spoon_I → mℓ	· 5	1 <i>international (metric) teaspoon</i>	
tsp_{UK} → mℓ	· 3,552	1 <i>Brit. teasp.</i> := $\frac{1}{4}$ <i>tablesp. (UK)</i>	
tsp_{US} → mℓ	· 4,928 92	1 <i>US teaspoon</i> := $\frac{1}{3}$ <i>tablesp. (US)</i>	
ton → kg	· 1 016,05	1 <i>Imperial ton</i> := 200 cwt	m:
torr → Pa	· 133,322	→ Pa.	F&p:

Conversion	Calculation	Remarks	Branch
troy oz. → g	· 31,103 5	<i>Troy ounce</i>	m:
troy oz. → kg	· 0,031 103 5		
Wh → J	· 3 600	<i>Watt-hour</i>	E:
W → hp^E	/ 746	<i>Watt</i>	P:
W → hp^M	/ 735,499		
W → hp_{UK}	/ 745,700		
yard → m	· 0,914 4	1 <i>yard</i> := 3 <i>feet</i>	x:
yǐn → m	/ 0,03	1 市引 := 10 <i>zhàng</i>	
year → s	· 31 556 952	= 365,242 5 · 24 · 60 ²	Time:
zhàng → m	/ 0,3	1 市丈 := 10 <i>chǐ</i>	x:

Some units more, being really simply convertible, follow:

- 1 *ångström* $\equiv 1 \text{ \AA} = 10^{-10} \text{ m} = 0,1 \text{ nm}$ may be convenient for wavelengths and atomic diameters (howsoever the latter are specified)
- 1 *fermi* $\equiv 1 \text{ fm} = 10^{-15} \text{ m}$; although a regular part of a *meter*, this is found as a separate unit on some calculators
- 1 *barn* $\equiv 1 \text{ b} = 10^{-28} \text{ m}^2$ for cross sections in nuclear and particle physics (note the name; *millibarn* and *nanobarn* are ‘common’ units in reactions there)
- 1 *tex* = $10^{-6} \text{ kg/m} = 1 \text{ mg/m}$ for yarn density in textile industry (where *decitex* is common for nylons), and
- 1 *muggeseggele* $\approx 3 \dots 1 \text{ }\mu\text{m}$ for Swabian precision mechanics.¹¹⁵

¹¹⁵ → <https://en.wikipedia.org/wiki/Muggeseggele> (frankly, I didn’t expect an article about this pretty local unit there at all).

Personal side note: I enjoyed the privilege of observing the *muggeseggele* in its natural habitat at the premises of a Swabian manufacturer of precision components for the textile industry – founded in 1852, family-owned until today, and one of several ‘hidden champions’ (i.e. world market leaders) located in south-western Germany (in

The 22 angular conversions provided work as follows (→ [OMS 3](#)):

 ...	Remarks
DEG→	Input: if x is of DT 1 or 2 then this number is taken as angular input in <i>decimal degrees</i> ; alternatively, x may be an angle (DT 4) in <i>decimal degrees</i> . Output: this angle in current ADM.
D.MS→	Input: x of DT 1 or 2 is taken as angular input in <i>sexagesimal degrees</i> (formatted dddd.d.mmsshh); alternatively, x may be an angle in <i>sexagesimal degrees</i> . Output: this angle in current ADM.
D.MS→D	Works like D.MS→ but converts x to an angle in <i>decimal degrees</i> (corresponding to vintage H.MS→H).

a hilly region whose inhabitants rightfully claim [wir können alles außer hochdeutsch](#) (translated: *we can do/make everything except speaking standard German*)) – and its affiliates worldwide for 16 years.

It was definitely fun listening to, e.g., Portuguese or Indian staff talking about *muggeseggele* – you can't translate it. On the shop floor, 1 *muggeseggele* meant significantly less than 1 *hundertstel* (= 0.01 mm) – but could still be reached manually by a well-trained technician tightening the proper bolt using appropriate manual (!) torque. It was a real difference, although any quantity deviating from just one *muggeseggele* was never observed in the wild by me (may be caused by language: German ein *Muggeseggele* translates to a as well as to one *muggeseggele*).

In production, quality microscopes and projectors featuring magnifications times 20 to 50 were typical tools for machine setup control. By the way, the staff never talked of high quality or precision there – machines, instruments, tools, and products were 'usable' or 'just right' at best. Feel free to compare to other companies' wording and the outcomes there, in particular in medium to long range.

Though there (as in almost all companies) was also some magic, like seasoned staff claiming being able to see a *micrometer* under one of the usual microscopes – you could verify such claims dispassionately using the *MSA* explained in [App. I](#).

Specifying the *muggeseggele* any further is certainly a challenge for a dedicated metrology engineer with very long wind.

Find more exotic units in a guide of *NIST* (<https://physics.nist.gov/cuu/pdf/sp811.pdf>). In its appendix B, it covers on 30 full pages not only many outdated units like *footcandles*, *oersted*, and *tons of refrigeration* but also tells comprehensively how to deal with them. Numerous *Imperial* units, however, are missing therein.

¹¹⁶Inputs in [radians](#) can't be converted exactly to other angular units (nor vice versa) since the necessary decimal value of 2π features a finite precision only in a calculator.

 ...	Remarks
D→D.MS , D→MIL , D→R	Work like DEG→ but convert x to an <i>angle</i> in <i>sexagesimal degrees</i> (corresponding to vintage H→H.MS), <i>mils</i> , or <i>radians</i> , ¹¹⁶ respectively.
GRAD→	Input: x of DT 1 or 2 is taken as angular input in <i>gradians</i> ; alternatively, x may be an <i>angle</i> in <i>gradians</i> . Output: this <i>angle</i> in current ADM.
MIL→	Input: x of DT 1 or 2 is taken as angular input in <i>mils</i> ; alternatively, x may be an <i>angle</i> in <i>mils</i> . Output: this <i>angle</i> in current ADM.
MIL→D , MIL→R	Work like MIL→ but convert x to an <i>angle</i> in <i>decimal degrees</i> or <i>radians</i> , ¹¹⁶ respectively.
MULπ→	Input: x of DT 1 or 2 is taken as angular input in <i>multiples of π</i> ; alternatively, x may be an <i>angle</i> in <i>multiples of π</i> . Output: this <i>angle</i> in current ADM.
RAD→	Input: x of DT 1 or 2 is taken as angular input in <i>radians</i> ; alternatively, x may be an <i>angle</i> in <i>radians</i> , ¹¹⁶ Output: this <i>angle</i> in current ADM.
Mπ→R	Works like MULπ→ but converts x to an <i>angle</i> in <i>radians</i> . ¹¹⁶
R→D , R→MIL , R→Mπ	Work like RAD→ but convert x to an <i>angle</i> in <i>decimal degrees</i> , <i>mils</i> , or <i>multiples of π</i> , respectively.
→DEG	a) x of DT 1 or 2 is taken as angular input in the current ADM and converted to an <i>angle</i> in <i>decimal degrees</i> , <i>sexagesimal degrees</i> , <i>gradians</i> , <i>mils</i> , <i>multiples of π</i> , or <i>radians</i> , ¹¹⁶ respectively. b) If, on the other hand, x is a tagged <i>angle</i> (DT 4) then its tag information will be used in conversion. Example: if $x = 1.5\pi$ (with the tag π) then →GRAD will return 300° regardless of the current ADM.
→D.MS	
→GRAD	
→MIL	
→MULπ	
→RAD	

Angular output is tagged always. – Also angular units may carry prefixes as listed on p. 154 in some industries: military instruments may indicate *milliradians*, for example.

Conversions between the other units (incl. *multiples of π*) may be more precise.

SECTION 3: CALLING AND EXECUTING COMMANDS

As mentioned above, the number of *items* featured on your *WP43* is far too large to fit them on the keyboard. Hence, there are several ways to call such an *item*.

You already know how to call an *item* appearing on the keyboard or in a *menu* or *catalog*. In [OMS 7](#), you have learned about storing *items* in user *menus* and /or assigning them to specific locations on your *WP43*. There is one more way you may call an operation, described in next chapter.

In the chapters following then, we will deal with input of parameters and strings, list all functions requiring parameters, allowing for indirect addressing, changing *data types*, and being allowed in equations.

Using XEQ for Executing Operations

Instead of picking an operation from a *menu* or *catalog*, you may also call it by specifying its *name*:

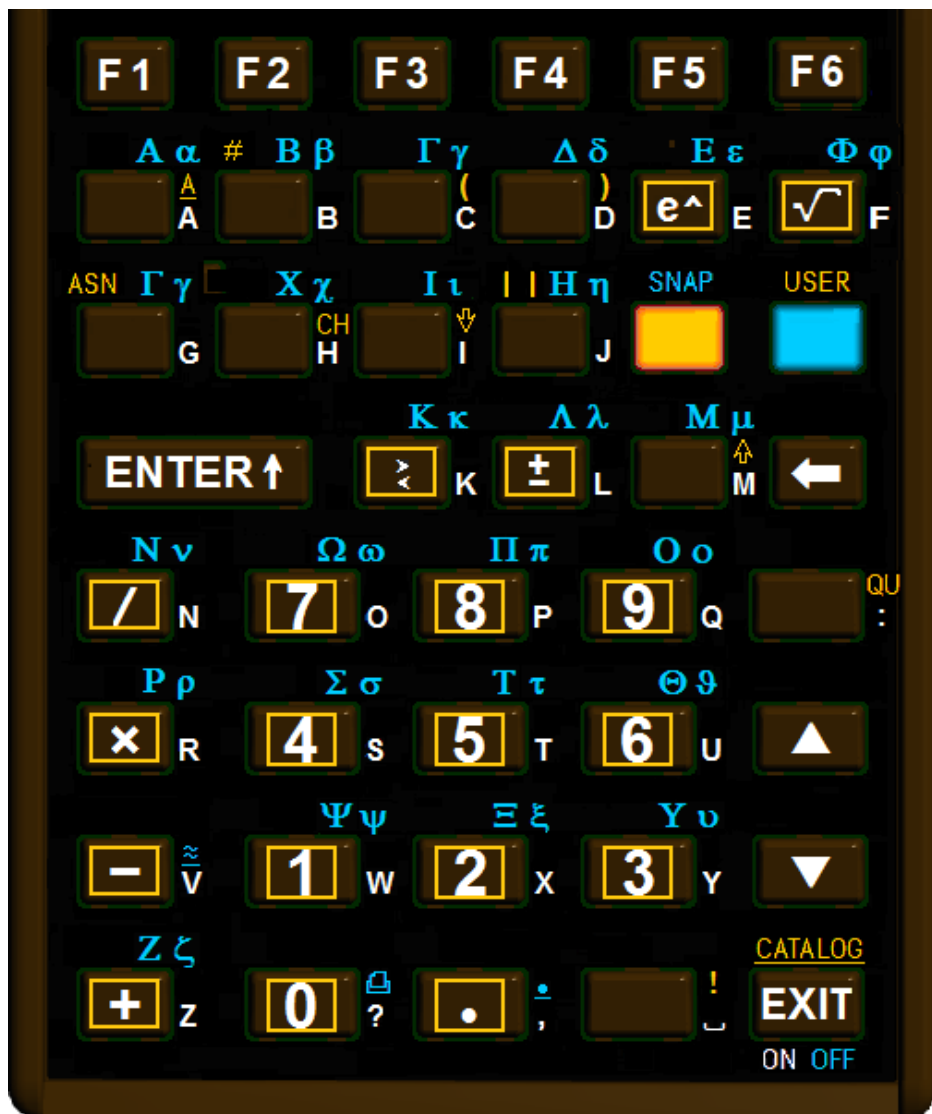
1. Press **XEQ** **(X)**. You enter *AIM* this way (→ 167 for the *virtual keyboard* applicable).
2. Type the *name* of the function wanted. Case is important, subscript or superscript is not. Press **ENTER** if the *name* is shorter than 7 *characters*.
3. Your input will be examined; if the specified operation exists, ...
 - a. you will be prompted for trailing parameters, if applicable (→ next chapter); then the function will be executed.
 - b. else **No such function** will be thrown.

This may be neither the easiest nor the fastest way to access an operation (correct spelling is required) but it is allowed for sake of tradition (cf. [HP-41C](#)). And it is independent of *menus*.

Command Parameter Input and Closing It

What will happen when a particular key is pressed while command parameter input is open? Note that a *character* may be a letter, digit, punctuation mark, symbol, etc. The table lists the respective keys beginning top left on the keyboard (→ 168ff for input in **X** instead):

Keystrokes	Situation	Meaning
A , B , C , D , I , J , K , L , T , X , Y , Z	addressing	Enters the address of a <i>global GP register</i> or <i>user flag</i> .
A ... Z  A ...   0 ...  9	entering a <i>label</i> , <i>system flag</i> , <i>variable</i> , string constant, or comment with <i>AIM</i> open	Appends the corresponding Latin or Greek letter or digit to the <i>item</i> pending. Use  and  to switch case for letters. Use  and  to move the cursor. → 167f for the <i>virtual keyboard</i> and limitations applying to comments and string constants.
ENTER ↑	arbitrary parameter input pending	If there is no input yet, assumes the default, if applicable. Else closes pending input, interprets it as expected for this command, and executes it (→ OMS 1).
	arbitrary parameter input pending	Deletes the last <i>character</i> entered. If nothing is left, cancels the pending command, returning to the state of your <i>WP43</i> as it was before that command was called.
0 ... 9	addressing or specifying	Enters a numeric parameter, an address, or a <i>local label</i> (→ OMS 1 and 4 for valid number ranges).
.	addressing	Header for <i>local registers</i> or <i>local user flags</i> .
EXIT	arbitrary parameter input pending	If there is an open <i>menu</i> , closes it. Else cancels pending command input, returning to the status of your <i>WP43</i> as it was before the current command was called.



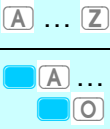
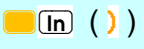
Virtual keyboard in *alpha input mode (AIM)*. AIM is also active while ...

- editing equations (→ 177f) as well as
- while a *catalog* is open, so you can use all accessible letters for alphabetic searching (→ 138f).

Note there is an 'alpha helper' on the backplate of your WP43 (→ 427).

Alphanumeric Input in X and Closing It

The table below shows what will happen when a particular key is pressed with (alpha-) numeric input being open (turn to p. 166 for command parameter input instead). Keystrokes printed on light blue work in equations, too. There and for *text strings*, characters will be either appended or inserted, depending on the actual cursor position. The table lists the respective keys on the keyboard top left to bottom right:

Keystrokes	in modes	Meaning
	A, α	Appends the corresponding Latin or Greek letter. Use  and  to switch case before if needed. Compare previous page and OMS 3. Input strings in <i>RUM</i> may take 2 full rows, string constants in <i>PEM</i> as well as REM-comments 4 rows or 196 <i>characters</i> (whatever is reached first), and variable names or labels up to 7 <i>characters</i> each.
	A, α	Appends # .
	$\neg$ (A, α)	Closes input of a short integer of base <i>n</i>
		sexagesimal angle
		date
		sexagesimal <i>time</i> in X. ¹¹⁷
	A, α	Appends (.
	A, α	Appends) .
	A, α	Appends $e^{(\cdot)}$.

¹¹⁷ Sexagesimal [angles](#) shall be entered like d.mmssff (> 1 digit is allowed for *degrees*).

Input of [times](#) shall be like h.mmssff (*hours* and fractions of *seconds* may both comprise > 1 digit). → [OMS 3](#) for examples and [App. B](#) for numeric limitations.

At closure, input will be checked – *seconds* (or *minutes*) > 60 will be carried to *minutes* (or *hours*). Illegal digits (e.g., 8 in octal input or C in decimal), bases, numbers, or *characters* found, or out-of-range conditions detected will cause an error thrown then (→  on next page and the error messages compiled in [App. C](#)).

Keystrokes	in modes	Meaning
	A, α	Appends $\sqrt{\text{ }}(\text{ })$.
	A, α	Appends CH or ch.
	A, α	Makes the next <i>character</i> a subscript, if applicable.
	$\neg$ (A, α)	Closes input of the first (i.e. <i>real</i> or <i>magnitude</i>) part of a <i>complex number</i> in X and waits for input of its second part (i.e. imaginary or <i>phase</i> , $\rightarrow$ OMS 3 and OMA 1). Any additional in input will be ignored.
	A, α	Appends $ \text{ } $.
	arbitrary input pending	If input was expected but not entered, cancels entry. If editing an equation, just closes it; else it <u>closes</u> input in X and checks the following conditions top-down: 1. If x contains at least one non-numeric <i>character</i> (except) , takes it as a text string. 2. Else (i.e. x is purely numeric) if x contains a , takes it as a complex number. 3. Else if x contains two , takes it as a fraction. 4. Else if x contains a or an , takes it as a real. 5. Else (i.e. x contains neither nor nor) looks for : a. If x contains a and a valid <i>base</i> trailing it, takes x as a short integer (<i>s.i.</i>) of said <i>base</i>; b. else recalls if the <u>previous entry</u> was a <i>s.i.</i>: if true, takes x as another <i>s.i.</i> of the same <i>base</i>; else takes x as a long integer. Then checks x (according to the condition met) as outlined in the 3rd paragraph of f. 117 on p. 168 and interprets it. Finally, unless an error had to be thrown, copies x into Y.
	A, α	Appends $\text{ }^{\text{ }}_{\text{ }}$.
	$\neg$ (A, α)	Changes the sign of the number entered. Affects the exponent if pressed after .

Keystrokes	in modes	Meaning
$\pm/\square$	A, α	Appends $\pm$.
E	$\neg (A, \alpha)$	Closes input of the mantissa, expecting input of the exponent ($\rightarrow$ OMS 1). Any additional E in input will be ignored.
E ($\uparrow$)	A, α	Makes the next <i>character</i> a superscript, if applicable.
$\leftarrow$	arbitrary input pending	Deletes the <i>character</i> left of the cursor. If nothing is left, cancels pending input, returning to the status of your WP43 as it was before that input was started.
$/$	A, α	Appends a $/$.
XEQ ($:$)	A, α	Appends a $:$.
XEQ (QU)	A, α	Appends QU or qu to the <i>string</i> x .
x	A, α	If $MULTx$ is set then appends x , else $\cdot$ to the <i>string</i> x . Editing an equation, inserts a $\cdot$ always.
$-$, $+$	A, α	Appends $+$ or $-$ to the <i>string</i> x or the current equation.
$INTS$ A ... F or BIT $\blacktriangle$ A ... F	$\neg (A, \alpha)$	Numeric input for <i>integer</i> bases >10 , appending the corresponding digit to x ($\rightarrow$ OMS 3). Digits will be checked when x is closed (compare the description of $ENTER\uparrow$ on previous page).
0 ... 9	$\neg (A, \alpha)$	Standard numeric input, appending the corresponding digit to x . Note you can enter ... • up to 34 digits plus a sign in the mantissa¹¹⁸ and up to four 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.
0 ($?$)	A, α	Appends a $?$.

¹¹⁸ You can enter even more digits but they will make no difference.

Keystrokes	in modes	Meaning
0 ... 9	A, α	Appends the respective digit.
0 ()	A, α	Appends .
	α	Turns to upper case for the letters following.
	A	Turns to lower case for the letters following.
	$\neg$ (A, α)	For D.MS input, separates <i>degrees</i> from <i>minutes</i> , <i>seconds</i> , and <i>hundredths of seconds</i> , so input format is dddd.mmsshh (d.ms) ($\rightarrow$ 168 and OMS 3). For <i>time</i> input, separates <i>hours</i> from <i>minutes</i> , <i>seconds</i> , and <i>fractions of seconds</i> , so input format is hhhh.mmssffff (h.ms) ($\rightarrow$ 168 and OMS 3). Else inserts a radix mark as selected.
2 nd	$\neg$ (A, α)	A 2 nd in input signals a fraction ($\rightarrow$ OMS 2 for examples). It separates nominator and denominator. Any further will be ignored. Note you cannot enter (E) after you entered twice but you may delete the 2 nd point while editing input.
	A, α	Appends a comma.
	A, α	Appends a point.
, program waiting for input (R/S)		Closes pending input and starts checking and interpreting it like (ENTER) on p. 169. Resumes program execution.
	A, α	Appends a blank space.
(R/S) ()	A, α	Appends a .
(EXIT)	arbitrary input pending	If there is an open <i>menu</i> , closes it. Else works like (ENTER) ($\rightarrow$ 169) but doesn't copy <i>x</i> into <i>Y</i> .

There are many more *characters* you can enter via the three alpha *menus* or via (CATALOG) (CHARS) ($\rightarrow$ 136 and 149ff). Call FBR for browsing the entire *character* sets provided.

Any key not specified in this table will close pending input and release it for interpretation like **ENTER**, **R/S**, and **EXIT** do.

Operations Requiring Trailing Parameters

Traditionally, *RPN* operations fetched their parameters from the *stack*, specifying what they shall do precisely (→ [OMS 1](#)). Many *WP43* functions, however, require at least one trailing numeric or text parameter now. Three tables below list these operations.

Note that for any command **CMX** requiring one trailing parameter, you can enter

CMX → **X**

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

Ops requiring 1 trailing parameter	Numeric parameter	Text parameter
AGRAPH BUZZ CONVG? DEC DSE DSL DSZ INC INPUT ISE ISG ISZ KEY? PLAY PUTK RCL RCL... STO STO... t_z VIEW x→α x_z x=? x≠? x≈? x<? x≤? x≥? x>? y_z z_z αLENG αPOS αRL αRR αSL α→x ?K TYP	Register # ¹¹⁹	Variable name
ALL DSP ENG FIX GAP RDP RSD SCI SDL SDR	# of decimals	
ASR MASKL MASKR RL RLC RR RRC SL SR WSIZE	Number of bits	
BACK CASE SKIP	# 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 or shortcut

¹¹⁹ # stands for *number* in this table, wherever applicable.

Ops requiring 1 trailing parameter	Numeric parameter	Text parameter
CNST	Constant number	
DSTACK	# of <i>stack registers</i>	
ERR MSG	Error number	
$f'(x)$ $f''(x)$ GTO LBL LBL? PGMINT PGMSLV VARMNU Π_n Σ_n and also SOLVE and $\int$ in RUM		<i>Label</i>
GTO		<i>Label</i>
GTO.	# of the target step	
INDEX MVAR M.DIM M.EDIN also SOLVE and $\int$ in PEM		<i>Variable name</i>
LocR	# of local <i>registers</i>	
PAUSE	Number of ticks	
RM	Mode number	
SIM_EQ	# of unknowns	
TDISP	Time display precis.	
TONE	Tone number	
VOL	Volume	
XEQ		<i>Label or name of operation</i>
→INT	Base	

Operations requiring 2 trailing parameters	1 st parameter	2 nd parameter
ASSIGN	<i>Item</i>	ENTER or sequence of key-strokes to access the target
KEYG KEYX	Key number (1 ... 18)	Program <i>label</i>

Operation requiring 4 trailing parameters	1 st to 4 th parameter
$\mathbb{Z}$	Name of stack register

Operations Allowing For Indirect Addressing

AGRAPH ALL ASR AVIEW BACK BC? BestF BS? CASE CB CF CNST
 CONVG? DEC DSE DSL DSP DSTACK DSZ ENG ERR FB FC? FC?C
 FC?F FC?S FF FIX FS? FS?C FS?F FS?S $f'(x)$ $f''(x)$ GAP GT0
 INC INDEX INPUT ISE ISG ISZ KEYG KEYX KEY? LBL? LocR MASKL
 MASKR MSG M.DIM M.EDIN PAUSE PGMINT PGMSLV PLAY PROMPT PUTK
 RCL... RDP RL RLC RM RR RRC RSD SB SCI SDL SDR SF SIM_EQ SKIP
 SL SOLVE SR ST0... TDISP TONE $t\mathbb{Z}$ VARMNU VIEW VOL WSIZE
 XEQ $x \rightarrow \alpha$ $x\mathbb{Z}$ $x=?$ $x \neq ?$ $x \approx ?$ $x < ?$ $x \leq ?$ $x \geq ?$ $x > ?$ $y\mathbb{Z}$ $z\mathbb{Z}$ α LENG
 α POS α RL α RR α SL α SR $\alpha \rightarrow x$ Π_n Σ_n ?KTYP and $\int$.

Note that correct indirect addressing can't be verified in *PEM*. This is because the content of the pointer *RoV* at execution time is unknown at the time the respective program step is entered.

Example:

Assume you are entering a program step

STO $\rightarrow$ POINT

then your WP43 can't know now what POINT will contain at (later) execution time of this step.

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* (functions printed on yellow below will return the input *DT* wherever possible). Find all these operations below:

Input DT	Operations	Output DT	Output registers
1	1/x AGM ALL cos ENG erf erfc e^x FIX gd gd^{-1} $J_y(x)$ $\ln\beta$ $\ln\Gamma$ MANT SCI sin tan W_m W_p W^{-1} $\beta(x,y)$ $\Gamma(x)$ Γ_{xy} γ_{xy} $\Delta\%$ $\rightarrow$ REAL % %MRR %T % Σ %MG and as well as all distributions, unit conver- sions, and orthogonal polynomials	2	X
	$\sqrt[3]{x}$ lb lg ln LOG _x y $\sqrt[y]{x}$ $\sqrt{x}$	2 or 3	
	$\rightarrow$ POL $\rightarrow$ REC	2	X, Y
	SLVC SLVQ	2 or 3	X, Y, Z
	$f'(x)$ $f''(x)$ SOLVE	2	X, Y, Z, T
	$\Sigma+$ $\Sigma-$	2	X, Y, STATS
	arccos arcsin arctan and all angular conversions	4	X
	$\rightarrow$ H.MS	5	
	J $\rightarrow$ D $\rightarrow$ DATE	6	
	$x\rightarrow\alpha$	7	
	FACTOR	8	
	M.GET M.NEW	8 or 9	
	$\rightarrow$ INT	10	
2	AND ceil DATE $\rightarrow$ DAY D $\rightarrow$ J EXPT floor IDIV IDIVR IP MONTH NAND NEXTP NOR NOT OR RANI# ROUNDI sign WDAY XNOR XOR YEAR	1	X
	DECOMP	1	X, Y
	SLVC SLVQ	2 or 3	X, Y, Z
	RE $\rightarrow$ CX	3	X
	arccos arcsin arctan and all angular conversions	4	
	$\rightarrow$ H.MS	5	
	$x\rightarrow$ DATE $\rightarrow$ DATE	6	

Input DT	Operations	Output DT	Output registers
2	$x \rightarrow \alpha$	7	
	M.GET M.NEW	8 or 9	
	$\rightarrow \text{INT}$	10	
3	ABS cross Im Re $ x $ $\nless$	2	X
	CX $\rightarrow$ RE	2	X, Y
	SLVC SLVQ	2 or 3	X, Y, Z
4	cos sin tan sinc sinc π $\rightarrow$ REAL	2	X
5	$\rightarrow$ HR	2	X
6	DAY D $\rightarrow$ J MONTH WDAY YEAR	1	X
	DATE $\rightarrow$	1	X, Y, Z
	$\rightarrow$ REAL	2	X
7	$\alpha \rightarrow x$	1	X
8	?M.DIM	1	X, Y
	DET dot ENORM $ M $	2	X
	$\Sigma+$	2	X, Y, STATS
	$V \nless$	4	X
9	?M.DIM	1	X, Y
	ENORM	2	X
	DET dot $ M $	3	
	ABS Im Re ROUNDI $ x $	8	
10	sign	1	X
	e^x LN LOG $_xy$ $\rightarrow$ REAL	2	
	$x \rightarrow \alpha$	7	
	FACTOR	8	

Operations in Equations

$\sqrt[3]{}$ () ABS() ACOS() arccos() arcosh() arcsin() arctan()
 arsinh() artanh() ASIN() ATAN() atan2(:) $B_n()$ $B_n^*()$ ceil()
 COMB(:) cos() cosh() erf() erfc() $e^{}()$ FIB() floor() $g_d()$
 $g_d^{-1}()$ $H_n()$ $H_{np}()$ IDIV(:) lb() lg() $L_m()$ $L_{m\alpha}()$ ln() $\ln\Gamma()$
 log(:) max(:) min(:) MOD(:) PERM(:) $P_n()$ RMD(:) sin()
 sinc() sinc $\pi()$ sinh() SQRT() tan() tanh() $T_n()$ $U_n()$ $W_m()$
 $W_p()$ $W^{-1}()$ $\sqrt[4]{}(:)$ $\Gamma()$ $\zeta()$ + - * / ^ ! | ∞ | $\sqrt{}$ ()

Equations shall be written in algebraic notation and will be evaluated following *PFEMDAS*. Functions and operations shall be spelled as they are in the *IOI* but followed by (), except the following ($\rightarrow$ 167):

Items	In equations	Examples in equations ¹²⁰
$1/x$	1 /	1 / b 1 / (17 + y)
10^x 2^x y^x	^	10^7.36 2^10 23.4^(-5) 2^(y · 1.2 + 3)
x^2 x^3	^	x^2 y^3 (3.4 · z)^2
e^x $(-1)^x$	e^, (-1)^	e^x e^(-6.7) e^(y - 2) (-1)^4.7
x!	!	z! 17! 23.4567! (x+3)!
x ABS		y - 7 z
$\sqrt{x}$ SQRT	$\sqrt{}$ ()	$\sqrt{(123.4)}$
$\sqrt[3]{x}$	$\sqrt[3]{}$ ()	$\sqrt[3]{(614.125)}$
$\log_x y$	log(:)	log(625:5)
$\sqrt[4]{y}$	$\sqrt[4]{}$ (:)	$\sqrt[4]{(625:4)}$ $\sqrt[4]{((x + 0.5):6)}$

Functions with two parameters like COMB shall read like COMB(y:x) in equations, with a colon separating both parameters and the 1st value corresponding to y, the 2nd to x.

Square roots, absolute values, and exponentials can be entered quickly

¹²⁰ Note that b, x, y, and z are variables in these examples.

using their function keys $\sqrt{x}$, $|x|$, and e^x inserting $\sqrt{(\quad)}$, $|\quad|$, and $e^{\quad}$, respectively ($\rightarrow$ OMS 5). All other functions can be either

- typed and spelled as listed above – watch upper and lower case; super- and subscripts can be replaced by standard *characters* in entry: **HnP** will be decoded as **H_nP**, for instance; or
- picked from CATALOG'FCNS.

There are no implicit multiplications. Multiplication dots are used for optimum visual differentiation from the letter **x**.

Constants may be entered

- numerically (like 1.2 or 345.69E78, the latter with a capital E) or
- spelling their *names* as they read in CONST ($\rightarrow$ 116ff) or
- picked from CATALOG'CONST.

ATTENTION 1: The plain lower case **e** recalls the *elementary charge* – this is not *Euler's e*! Also **a**, **c**, **F**, **G**, **h**, **k**, **R**, **α** , **γ** , **Φ** , and **ω** are constants and hence must not be taken for *variables* in equations (while you may use these in routines).

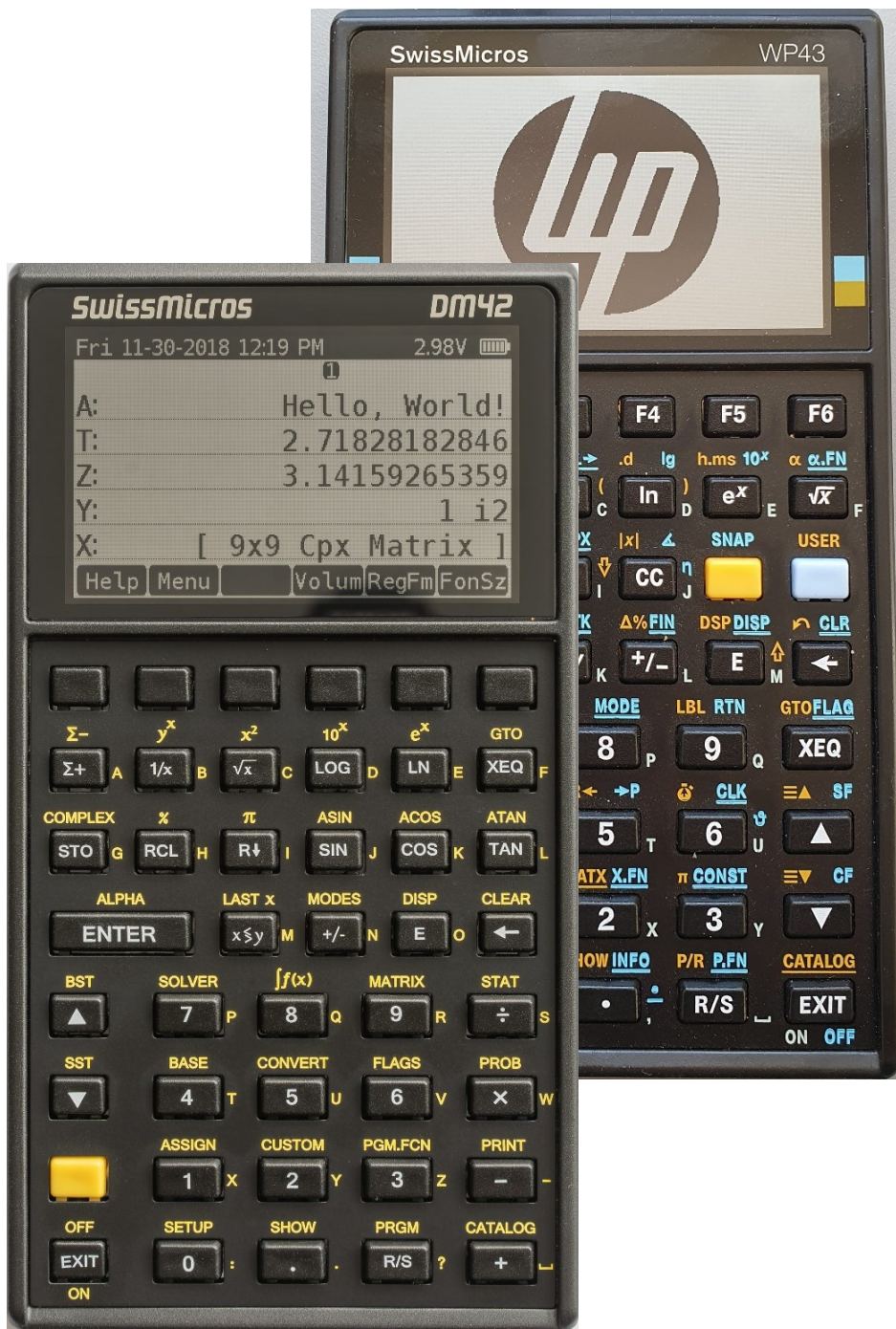
ATTENTION 2: Watch proper spelling of functions, constants, and *variables*. Your WP43 will definitely not decode **SIN(α)** as you may have meant it, but as you wrote it. How shall it know any better?

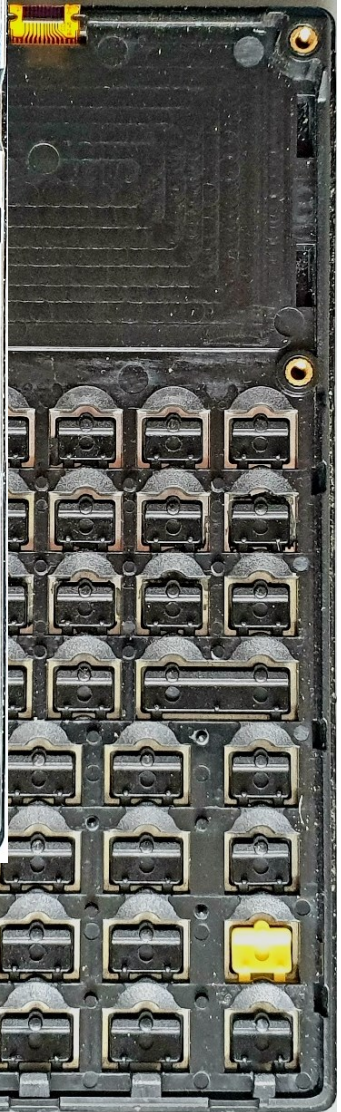
APPENDIX A: HARDWARE

- Dimensions: wedge-shaped 77 mm × 144 mm × 13 mm (at top above display) or 8 mm (at bottom).
- Mass: 180 g incl. battery.
- LCD: monochrome high contrast (14:1) transfective memory display with an active area of 58.8 mm × 35.3 mm, 400 × 240 square 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 at 80 MHz with power supplied through the *USB* port.¹²¹
- Memory: 1 MB *FM* and 128 kB *RAM* (incl. 96 kB user *RAM*) on *CPU*. 8 MB external *QSPI FM* (~ 6 MB of these are formatted as *FAT* filesystem for loading and saving programs as well as backup of *WP43 states*, accessible via *USB* as *FD*; → [App. F](#)).
- Power supply: 3 V by one *CR2032* lithium coin cell (battery life up to 3 years, non-rechargeable); optionally powered through the *USB* port (a good battery shall be installed); typical average currents drawn: power on and 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 as external mass storage device.
- Keyboard overlays: three short slots on either side of the keyboard are provided in the calculator frame for easily fixing your overlay sheets (≤ 0.35 mm thick) with your personal layouts printed on. → 186 for a drawing. Look up [OMS 7](#) for inspiration.

Pictures of the hardware are shown on the next six pages. The keyboard views on pp. 181f are printed approximately to scale. See there why the positions of the keys (in particular the prefixes) are fixed.

¹²¹ → https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-ultra-low-power-mcus/stm32l4-series/stm32l4x6.html.

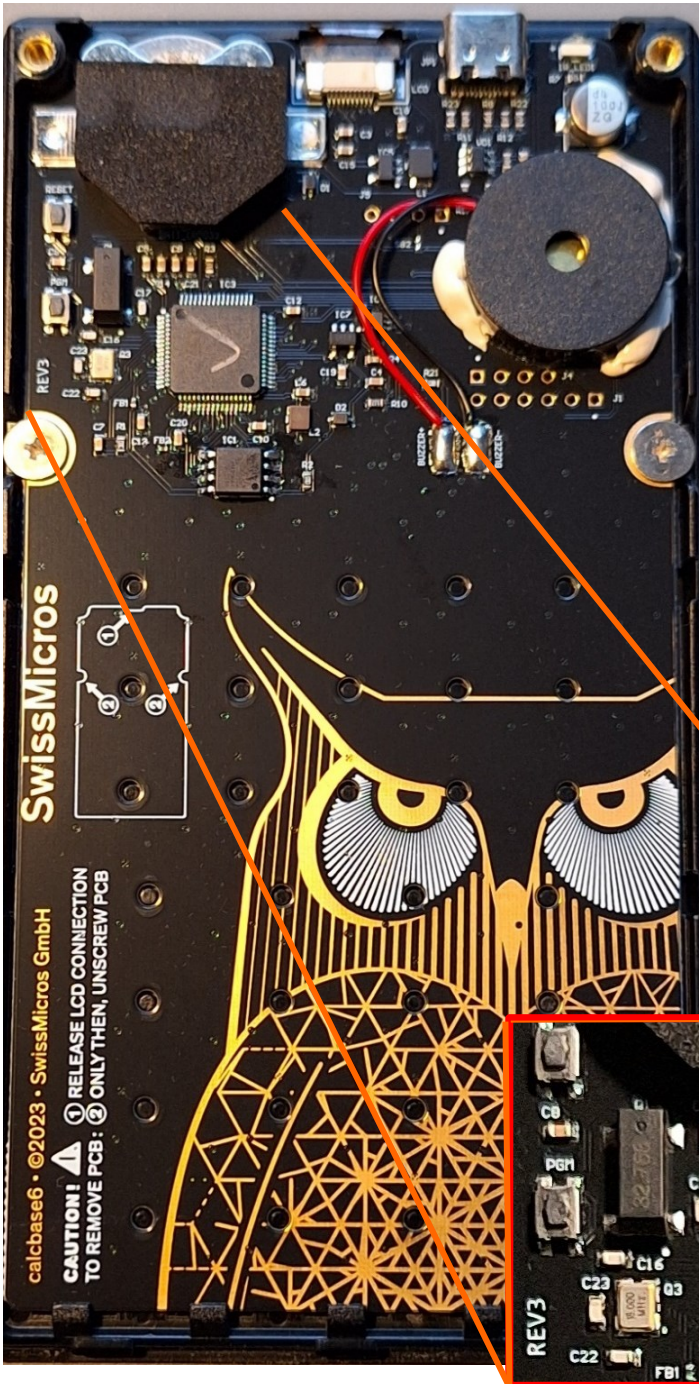




here: <https://www.hpmuseum.org/forum/thread-10143.html>

See below the *PCB* of an early *DM42* (β) of spring 2017. RESET is the top left button. The wide trace around the *PCB* is on minus (-).





This is the inside of a *DM42n* launched in 2024-09.

The two buttons changed shape, and a (wider and higher) USB-C socket is mounted for connecting to your PC or an auxiliary power supply. Hence, also the corresponding hole in the bottom case became larger than for *DM42* / *WP43*.

Note *WP43* does not run on *DM42n* for the time being.

Off-Images

When turned off, your *WP43* will show an image until it is turned on again. You can create such *off-images* for your *WP43* yourself. They shall be stored on the *FD* in directory *OFFIMG* – your *WP43* will display them one after the other.

See below and on the frontpage of this manual what could be done:¹²²

Periodic Table of Elements																		He ²
1 H 1,008																	2 He 4,003	
IIA																		
3 Li 6,968	4 Be 9,012																	10 Ne 20,18
		Number		Symbol		Atomic mass												
		e = 1.602 176 634 E-19 As		m _e = 9.109 383 701 5 E-31 kg		m _n = 1.674 927 498 04 E-27 kg		m _p = 1.672 621 923 69 E-27 kg		N _A = 6.022 140 76 E23								
11 Na 22,990	12 Mg 24,305	3 IIIB	4 IVB	5 VB	6 VIB	7 VIIB	8	9 VIII	10	11 IB	12 IIB	13 Al 26,981	14 Si 28,085	15 P 30,974	16 S 32,067	17 Cl 35,451	18 Ar 39,948	
19 K 39,098	20 Ca 40,078	21 Sc 44,956	22 Ti 47,867	23 V 50,941	24 Cr 51,996	25 Mn 54,938	26 Fe 55,845	27 Co 58,933	28 Ni 58,693	29 Cu 63,546	30 Zn 65,380	31 Ga 69,723	32 Ge 72,631	33 As 74,922	34 Se 78,972	35 Br 79,904	36 Kr 84,798	
37 Rb 85,468	38 Sr 87,620	39 Y 88,906	40 Zr 91,224	41 Nb 92,906	42 Mo 95,950	43 Tc 97,070	44 Ru 101,070	45 Rh 102,905	46 Pd 106,420	47 Ag 107,868	48 Cd 112,414	49 In 114,818	50 Sn 118,711	51 Sb 121,760	52 Te 127,600	53 I 126,904	54 Xe 131,294	
55 Cs 132,906	56 Ba 137,328	57 La 138,905	72 Hf 178,490	73 Ta 180,948	74 W 183,840	75 Re 186,207	76 Os 190,230	77 Ir 192,217	78 Pt 195,085	79 Au 196,967	80 Hg 200,592	81 Tl 204,384	82 Pb 207,200	83 Bi 208,980	84 Po 209	85 At 210	86 Rn 222	
87 Fr 223	88 Ra 226	89 Ac 227	104 Rf 261	105 Db 269	106 Sg 270	107 Bh 272	108 Hs 273	109 Mt 277	110 Ds 281	111 Rg 281	112 Cn 285	113 Nh 286	114 Fl 289	115 Mc 288	116 Lv 293	117 Ts 294	118 Og 294	

Overlays

A drawing for a blank overlay is printed overleaf, copied directly from *SM*.¹²³ All dimensions are given in *millimeter* without tolerances there. This drawing is not printed to scale. Note the overlay is 72.5 mm wide overall and its upper horizontal inter-key spacing is

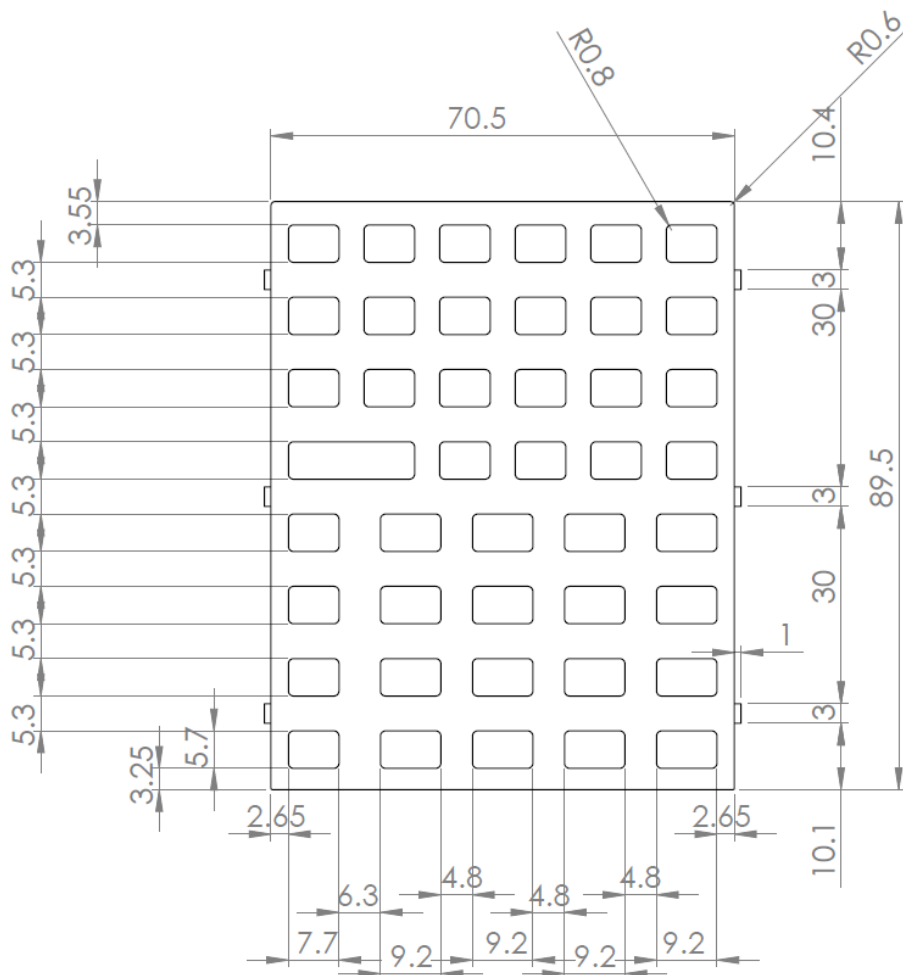
$$(70.5 - 2 \times 2.65 - 6 \times 7.7)/5 = 3.8$$

For cutting the key voids, do not expect matching punches being readily available. A standard 6 mm square punch might help considerably.¹²⁴

¹²² Find many more at <https://forum.swissmicros.com/viewforum.php?f=14>.

¹²³ → http://technical.swissmicros.com/graphics/key_overlay_DM42.PDF.

¹²⁴ Found here in 2024 for < 20 €:



In China, there was a manufacturer cutting overlays out of 0.5 mm plastic sheets in 2023, as pictured overleaf. This matter was discussed on the forum¹²⁵ (the picture overleaf is copied from it). Recently, tabbed (0.45 mm thick) or

https://www.amazon.de/Vierkant-Leder-Handwerk-Punch-Gr%C3%B6%C3%9Fen/dp/B00NVHB134/ref=sr_1_37?_mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crd=3NQMUMGVB2I04&keywords=stanzwerkzeug+papier+vierkant+6mm&qid=1654787046&prefix=stanzwerkzeug+papier+vierkant%2Caps%2C73&sr=8-37 though certainly available from less relentless sources as well.

¹²⁵ → <https://forum.swissmicros.com/viewtopic.php?f=24&t=3442>

adhesive overlays are available from [Jeff Roetman](#) (USA).¹²⁶ Do not choose any thicker material – I recommend 0.35 mm like the overlays of [HP-48](#) or [HP-41C](#) were.

Find the original print of keys and keyplate (to scale) for your *WP43* on p. 428.



¹²⁶ → <https://calcoverlays.com>. Note all SM portrait pocket calculators (and thus also your *WP43*) share the same keyboard and slot geometry

APPENDIX B: MEMORY MANAGEMENT

Data Types

You know ten *DT*'s from [OMS 3](#). Some more had to be defined for internal use, e.g.:

- 7-character strings for all kinds of *names*, e.g., of commands, *variables*, or *labels*,
- system integers in the range of $\pm 2\,147\,483\,648$ (i.e. 32 *bits*),
- *flag* words for storing 144 (i.e. 112 global plus 32 local) *user flags* and 112 *system flags* in 256 *bits* in total,¹²⁷
- two *DT*s for two kinds of *menus*,
- one *DT* of variable length for storing *configurations* (→ **ST0CFG** and **RCLCFG**),
- another *DT* for *expressions* in **EQN** (→ [OMS 5](#)), and
- two more for program steps and routines (→ [OMS 4](#)).

Generally, data (or objects, outside of routines) can be contained in (*global* or *local*) *RoV*'s.

A 1-*block* (= 4 *byte*) *register header* is specified for each of the 112 global *registers*, containing a pointer to the data contained in this *register*¹²⁸, the *internal DT* (→ overleaf), and some data information.¹²⁹

For each (sub-) routine level, 3 *blocks* are reserved in *RAM* ensuring proper return. For every (sub-) routine requiring *n* local *registers*, 1 + *n* *blocks* are allocated in addition, with the first *block* for the 32 local *flags*.

An 2-*block* *named variable header* is specified for each *variable*, containing a pointer to the data contained in this *variable*, the *internal DT*, some data information,¹²⁹ a pointer to the *name* of the *variable*, and its length.

¹²⁷ We need far less than 112 *system flags* so far.

¹²⁸ It points to the first memory block number of the *register* content. This allows access to $65\,535 \text{ blocks} \times 4 \text{ bytes / block} = 262\,140 \text{ bytes} = 256 \text{ kB}$ minus 4 *bytes*. A special value is reserved for the NULL pointer.

¹²⁹ Like display base for a *short integer* (2 ... 16), sign for a *long integer* (0: zero, 1: negative, 2: positive), angular unit for an *angle* or *real number* (0: *radian*, 1: *grad* or *gon*, 2: *degree*, 3: *mil*, 4: *sexagesimal degree*, 5: *multiple of π* , 6: no *angle* = plain *real*).

The data themselves are then stored in the following way:

DT number and meaning			Size [in <i>blocks</i> or <i>bytes</i>]	
1	0	$\mathbb{Z}$ Long integer ¹³⁰	$\geq 1 + 1$	8 - 1109
2	1	$\mathbb{R}$ Real number ¹³¹ (<i>real34</i>)	$1 + 4$	20
3	2	$\mathbb{C}$ Complex number, always stored in rectangular notation (<i>complex34</i>)	$1 + 2 \times 4$	36
4	1	Angle ¹³²	$1 + 4$	20
5	3	Time ¹³³	$1 + 4$	20
6	4	Date ¹³⁴	$1 + 4$	20
7	5	Text string ¹³⁵	$1 + \frac{j+1}{4} + \frac{k}{2}$	8 - 100
8	6	Real matrix (of n rows and m columns) ¹³⁶	$1 + n \times m \times 4$	≥ 20
9	7	Complex matrix (of n rows and m columns) ¹³⁶	$1 + n \times m \times 8$	≥ 36

¹³⁰ DT 1 is for number theory problems. The header *block* indicates the size (in *blocks*) of the *integer* following. 1 *block* following allows for signed *integers* up to $2^{31} \approx 2 \times 10^9$. Size increment is 1 when required. Maximum *long integer* size is 1108 *bytes* equivalent for 1001 decimal digits. Look up the display limits further below.

¹³¹ Standard *reals* feature 128 *bits* (= 16 *bytes*) and 34-digits precision here. For comparison, *WP 34S* standard was 64 *bits* and 16 digits. → [Range and Representation of Real Numbers](#) below.

¹³² A tagged *angle* is stored as a *real number* with a special header as described on previous page.

¹³³ Internally, a *time* or time *interval* is stored as a *real number* of *seconds*, just with a specific header. Decimal times are displayed in *hours*. This format allows for expressing intervals of some 5×10^{11} *years* with *femtosecond* precision, exceeding requirements in this universe.

¹³⁴ Internally, 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).

¹³⁵ ... containing j 1-byte characters and k 2-byte characters (with bit 15 always set). The header *block* indicates the total string length L in *blocks*, and 1 stop *byte* after all the *characters* contains 00. The minimum increment of L is 1 *block*.

In programs, *text strings* are shorter: there is just 1 *byte* heading and no trailing 00. → [Program Step Sizes](#) below.

¹³⁶ 2 *bytes* for n , 2 *bytes* for m , then follow the individual matrix elements row by row, taking either 4 or 8 *blocks* each.

DT number and meaning			Size [in <i>blocks</i> or <i>bytes</i>]
10	8	Short integer ¹³⁷	1 + 4 20
11	9	Configuration (as stored by STOCFG)	1 + M ¹³⁸
12	5	Label (or name – up to 7 characters) ¹³⁵	$1 + \frac{j}{4} + \frac{k}{2}$ 2 - 5
13		Constant ¹³⁹	1 4
14		System and user flags (112 + 144 flags)	8 32
15		Extended precision <i>real number</i> (39 digits, for internal use only, not exposed to the user) ¹⁴⁰	1 + 8 36
16		System integer (for internal use only)	1 4
17		User-created <i>menu</i> (limited to 1 <i>view</i>)	1 + 18 × 7/2 256
18		Predefined <i>menu</i> (featuring <i>n</i> <i>views</i>)	1 + <i>n</i> × 18 × 7/2
19		Program step	→ 204f
20		Program, containing <i>n</i> program steps	1 + <i>M</i>
21		<i>Expression</i> (for members of <u>EQN</u>), may be stored as <i>text string</i> (DT 7) as well	1 + <i>M</i>
22		<i>Calculator state</i>	1 + <i>M</i>

The DTs 7 - 9, 11, and 19ff are of 'infinite' size limited by available memory (*M*) only. The size of each individual object is fixed though.

Mathematically, every *integer* is included in the set of *reals* which is included in the set of *complex numbers* (generally: $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$). Thus, **3 CPX?** should return **true**. But we doubt you expected that. Hence, **x CPX?** will return only **true** if *x* is no *real* object, and **x REAL?** will be only **true** if *x* has a non-zero fractional part. **MATR?** will return **true** also for 1x1 matrices.

¹³⁷ DT 10 is for computer science problems.

¹³⁸ The size will vary with the number of user assignments being part of the *configuration* (→ **OMS 7**).

¹³⁹ A pointer to the constant is sufficient here, regardless of its own precision.

¹⁴⁰ There are even longer (i.e. more precise) *reals* used in internal computations, → further below.

As mentioned above, any object of any *DT* will take one storage space only, i.e. one *RoV*. In consequence, *RoV* lengths may vary considerably. You do not have to bother – the operating system of your *WP43* 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 – monitoring the free space available by **?MEM**, you may observe differences following, e.g., changing contents of your *stack*.

What Will Happen when Changing Word Size?

WSIZE affects *short integer* contents of all allocated *stack registers* and *L* exclusively: Reducing word size will truncate all *short integer* values therein down to the new size. Increasing word size will fill empty bits in the significant side of each *short integer* concerned, up to the new size specified.

ATTENTION: Increasing word size will just add empty bits. Thus, a negative *short integer* will immediately become positive then. There is no automatic sign extension! If you want it, take care of it yourself.

All other memory content of your *WP43* will stay as it is. This differs from the implementation as known from the *HP-16C*, where all *registers* were remapped on word size changes.

When *short integers* are recalled from memory which are longer than current word size, these *integers* will be truncated during recall. When *short integers* are recalled from memory which are shorter than current word size, these *integers* will be filled with empty bits during recall.

SAVE and LOAD...

SAVE stores the *calculator state* in the standard backup file on the *FD* in directory *SAVFILES* (a 2nd **SAVE** will overwrite the 1st). Alternatively, **I/O SAVEST** stores the current *state* in directory *STATE* in a named file specified by you, using *DMCP* menus. Residing in *FM*, all *state files* will survive **RESET**s, even hard ones by the **RESET** button, and are battery fail safe.

In detail, the data stored by **SAVE** and **SAVEST** are ...

1. all equations and programs present in *RAM*,
2. all *local registers* and *flags*,
3. all *global registers*, *global user* and *system flags*,
4. all *variables* (incl. *STATS* if allocated),
5. all statistical sums (if *STATS* is allocated),
6. all user *menus* (incl. *MyMENU*, *MyPFN*, and *MyQ*),
7. all keyboard assignments,
8. all current values of *ADM*, *D.MAX*, *ISM*, *REALDF*, and *#DEC*,
9. all present settings of *DSTACK*, *GAP*, *J/G*, *BestF*, *LocR*, *HIDE*, *RANGE*, *RM*, *RSD*, *TDISP*, *WSIZE*, and of the random number generator.

Various flavors of **LOAD** allow for recalling *states* or parts of them according to your requests, and resuming calculator operation also after RESETs. Turn to the *IOI* for more information about these commands.

A *calculator configuration*, on the other hand, encompasses all data printed on light blue above plus the *system flags* marked light blue on pp. 129 - 132. A *configuration* can be saved in *RAM* by **STOCFG** and recalled by **RCLCFG** only.

Range and Representation of Real Numbers

Your *WP43* can calculate with *real numbers* of more than 12 000 orders of magnitude (precisely: for $10^{-6143} \leq |x| < 10^{+6145}$). Its software is based on the *decNumber library* supporting arbitrary precision *BCD* numbers (→ 334).

Overall accuracy should be equal or better than $\pm 3 \times 10^{-33}$ per operation (→ 265ff) except at particularly 'nasty' locations.¹⁴¹ Check your algorithms carefully to avoid such locations. Note that rounding errors accumulate. In most cases, however, precision of your calculatory results will exceed anything known in the real world nevertheless (→ f. 80 on p. 124).

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

All these effects are caused by the internal representation of *reals*: *Standard*

¹⁴¹ E.g., $0.5^\circ [\cos] [\arccos] 0.5^\circ \square 0.5^\circ \angle$ returns 3.6×10^{-31} due to the cosine of such a small angle being very close to 1.

floating point numbers are stored in your WP43 in sixteen bytes using an internal format coarsely following *decimal128* packed coding, though with some exceptions:¹⁴²

- *Real* zero is stored as *integer* zero, i.e. all bits cleared.
- The *significand*¹⁴³ of a *real number* is encoded as pure integer in eleven groups of three digits. Each such group is packed into ten bits straight forward, meaning, e.g., $555_{10} = 10\ 0010\ 1100_2 = 22B_H$ or $999_{10} = 11\ 1110\ 0111_2 = 3E7_H$. So the 33 rightmost decimal digits of the *significand* take its least significant 110 bits. Trailing zeros are omitted, so the *significand* will be right adjusted.
- The most significant (128th) bit carries the sign of the *significand*.
- The remaining 17 bits are used for the exponent and the leftmost digit of the *significand*. 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 1000_2 and the result (8_{10} or 9_{10}) will represent the leftmost digit of the *significand*.

If uu reads 00, 01, or 10_2 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 (→ 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 $12287 - 6176 +$

¹⁴² Find out about *decimal128* in https://en.wikipedia.org/wiki/Decimal128_floating-point_format. Our coding comes close to what Prof. *William Kahan* called *quadruple (precision)* in his paper titled '*IEEE Standard 754 for Binary Floating-Point Arithmetic*' (<https://people.eecs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF>) of 1997.

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

¹⁴³ *Significand*, not *mantissa* – the *significand* contains no radix mark.

34 = 6145 as well as $-6176 + 34 = -6142$, *DT 2* can support 34-digit numbers within $10^{-6143} \leq |x| < 10^{+6145}$.

Rewarding your patience so far, we will show you some illustrative **examples** of encoding in your *WP43* instead of telling you more theory. *SE* stands for the *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 (111 111×10 ⁻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 ×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
9.99×10 ⁹⁹ (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
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 in the *significand*, stored as 1.

number displayed on your WP43 will be true to 34 digits ($\rightarrow 265\text{ff}$). And errors accumulate as explained in f. 71 on p. 117.

Settings of **RM** may affect results of high precision calculations.

Limitations

Maximum numeric input for *data type (DT)* ...

- 1: $\pm 10^{1001}$ or $\pm 2^{3326}$
- 2: $\pm 9.999\ 999\ 999\ 999\ 999\ 999\ 999\ 999\ 999 \times 10^{\text{RANGE}-1}$
(absolute maximum for **RANGE** is 6145)
- 3: $\pm 9.999... \times 10^{\text{RANGE}-1}$ for either part ($\rightarrow \text{DT } 2$)
- 4: $\pm 9.999... \times 10^{\text{RANGE}-1}$ in arbitrary *ADM*, so the actual absolute maximum is $9.999... \times 10^{\text{RANGE}-1} \times \pi$.

ATTENTION: Trigonometric functions actually operate on *angles* between $-\pi$ and $+\pi$ *radians* exclusively. The precise modulo 360° (or 2, 400^g, 6400[˚], or 2π) reduction of greater *angles* ensures that these functions return calculatory correct (though not necessarily meaningful) 34-digit results for an input range of $\pm 10^{999}$ *degrees* (or *multiples of π , gradians, mils or radians*). Note input $\gtrsim 10^{34}$ makes no sense for these functions since no partial rotation can be specified there anymore.

- 5: $\pm 9\ 999\ 999\ 999\ 999\ 999\ h\ 59' 59.999\ ..." \approx \pm 1.14 \times 10^{12}$ *years* exceeding the age of the universe.
- 6: 99 999 999 999.123 123 595 9 .d with YMD. Else the input maxima are much smaller: 31.129 999 235 959 .d with DMY and only 12.319 999 235 959 .d with MDY. Minimum is -4 712.010 112 .d with YMD (or -1.014 712 12 .d with DMY/MDY, $\rightarrow 190$) for the first day of year 4712 B.C. 12:00:00 (year 0 for *Julian days*).
- 8: $\pm 9.999... \times 10^{\text{RANGE}-1}$ for each matrix element ($\rightarrow \text{DT } 2$)
- 9: $\pm 9.999... \times 10^{\text{RANGE}-1}$ for either part of each matrix element ($\rightarrow \text{DT } 3$).
- 10: For arbitrary word size **n** and unsigned mode, $x < 2^n$ (i.e. about 1.8×10^{19} for **n** = 64); for signed modes, $-2^{n-1} < x < 2^{n-1}$.

Internal limits:

- Any operation on long integers must not exceed 3328 bits (corresponding to 6.727×10^{1001}).
 - PRIME?** works like the other binary tests described in [OMS 4](#). Above 3 317 044 064 679 887 385 961 981 ($\approx 3 \times 10^{24}$), however, **NEXTP** can find no more primes but 'probable primes' only ($\rightarrow 71$). If you want to apply **NEXTP** or **PRIME?** there, consult appropriate reference literature.¹⁴⁴
 - Prime factors will be accurate up to 3 317 044 064 679 887 385 961 980 ($\approx 3 \times 10^{24}$, $\rightarrow 40$ for **FACTOR** and $\rightarrow 272$ for some caveats).
- | | | | | | | |
|-----------------------------------|-----|-----|-------------------|-------------------|----------------------|--|
| 3 317 044 064 679 887 385 961 980 | | | | | | |
| 2.0 | 3.0 | 5.0 | 1.3×10^2 | 1.9×10^7 | 8.6×10^{11} | |
| 2.0 | 4.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
- Any intermediate *real* result within $(-10^{-6176}, 10^{-6176})$ will be assessed as **0**. (compare previous chapter). Any intermediate *real* result exceeding $(-10^{6145}, 10^{6145})$ will be assessed as $-\infty$ or ∞ , respectively, and an overflow error will be thrown for **SPGRES**. This holds for matrix elements and parts of *complex numbers* as well.
 - The equation solver (in **EQN**) features an independent numeric *stack* (rather than the global **XYZTABCD** *stack*). Both operator and numeric *stacks* are 10-level each, so up to 9 pairs of parentheses may be nested.

Maximum numeric output for **DT** ...

- 1:** $\pm 10^{1001}$ with full 1001-digit precision (though you can see just up to 298 digits of a very long integer in one view, $\rightarrow 83$). **DUMP** allows to write any number entirely onto the **FD** ($\rightarrow 37$).

¹⁴⁴ $\rightarrow$ https://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test for a start. Whatever is a true prime will be confirmed by **PRIME?** – a composite may be falsely assessed as prime with a probability of 9×10^{-16} .

With some care and a lot of time (we recommend **USB** power supply), **NEXTP** on the **WP43** calculator could find 'probable primes' for *long integers* up to some 7.721×10^{317} with few exceptions near the upper limit (hence input for **NEXTP** is limited to 10^{308} here). You can get a 298-digit prime within some 40 *seconds* on your battery powered **WP43** – and read it entirely on its screen using **SHOW**.

On the **WP43 Simulator**, **NEXTP** finds 'probable primes' for *long integer* input up to some 4×10^{1001} ; and **PRIME?** will check them and their neighboring numbers (but you have to use **DUMP** to get all their digits out). – Presently, it can even do a lot more: if you start with a *real* $> 10^{1001}$, **NEXTP** will return 'probable primes' up to 10^{4000} within some minutes! Although integers $> 10^{1001}$ are not supported officially, these numbers can be dumped and are probably correct.

- **2, 3, 8, and 9:** The maxima equal those for input above. Any *real* result exceeding $(-10^{RANGE}, +10^{RANGE})$ will be displayed as $-\infty$ or ∞ , respectively, and any result within $(-10^{RANGE}, +10^{RANGE})$ or $(-10^{HIDE}, +10^{HIDE})$ will be displayed as '0.' as long as **RANGE** is within the limits stated for intermediate results. Else any *real* result assessed as 0. will be displayed as such, and anyone assessed as $-\infty$ or ∞ will be displayed as $-\infty$ or ∞ if SPCRES is set; else an overflow error will be thrown. This holds for matrix elements and parts of *complex numbers* as well.
- **4:** For angular conversions, the limits are as specified for input above. The functions **arcsin**, **arccos**, and **arctan** return values between $-\pi$ and $+\pi$ (or their equivalents) only.
- **5:** The maximum is as specified for input above. Time intervals smaller than the display limit set by **TDISP** (< 1 ms for **TDISP** 0) are shown in a format like **SCI** 4 (or **ENG** 4 with **ALLENG** set) but with a trailing 's', e.g., -1.2345×10^{-9} s (more digits can be shown by **RBR** or **SHOW**).
- **6:** The *date* limits are as specified for input with **YMD** above. Large *dates* may be displayed as well with **DMY** or **MDY**, however; resulting negative *dates* will be displayed like 11.09.-0753 or 09/11/-0753.
- **10:** The maxima are as specified for input above.

Further limitations for *data type* ...

- **7:** *Text strings* are limited to 196 *characters*.
- **11:** *Configurations* cannot be displayed. Thus, if any *RoV* containing such data is on screen (e.g., in **RBR** or after **RCL**), **Configuration data** is shown instead of its content.

Special Results

Throughout this chapter, SPCRES is presumed being set. Thus, infinities and non-numeric results are legal – no error messages will be thrown if such results happen to occur (→ 196).¹⁴⁵

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

¹⁴⁵ Results were crosschecked against **WP 34S** and *WolframAlpha* wherever possible. Where deviations are observed we are confident your **WP43** returns the correct results.

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

And the following monadic functions operate also on infinities:

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

The functions printed in light yellow fields in the two tables above will also return NaN (or NaN+i×NaN) with complex results allowed. Others will change their output

when $\mathbb{C}$ is lit. – For dyadic functions, we combined the respective tables:

Input y	x	Operation(s)	Output for $\mathbb{R}$ lit
∞	$x \neq -\infty$	$+$	∞ ¹⁴⁶
$-\infty$	$x \neq \infty$		$-\infty$ ¹⁴⁶
$-\infty$	∞	$+$	NaN ¹⁴⁶
∞	$x \neq \infty$	$-$	∞ ¹⁴⁷
$-\infty$	$x \neq -\infty$		$-\infty$ ¹⁴⁷
$-\infty$	$-\infty$	$-$	NaN
∞	∞		
∞	$x > 0$	$\times$	∞ ¹⁴⁶
$-\infty$	$x < 0$		
∞	$x < 0$	$\times$	$-\infty$ ¹⁴⁶
$-\infty$	$x > 0$		
0 or $0.$	$-\infty$ or ∞	$\times$	NaN ¹⁴⁶
$0 < y \leq \infty$	$0.$	$/$	∞
$-\infty \leq y < 0$			$-\infty$
$-\infty$ or ∞	$-\infty$ or ∞	$/$	NaN
0 or $0.$	$0.$	$/, y^x$	
$-\infty$ or ∞	0 or $0.$	y^x	
$-\infty \leq y < 0$	$-\infty$ or ∞	$\sqrt[y]{y}$	NaN
$-\infty < y < 0$	non-integer x	$y^x, \sqrt[y]{y}$	NaN
$-\infty$	odd $x > 0$		$-\infty$
$-\infty$	even $x > 0$	y^x	∞
		$\sqrt[y]{y}$	NaN

¹⁴⁶ Swapping x and y will return the same result here.

¹⁴⁷ Swapping x and y here will return this result times -1 .

Input y	x	Operation(s)	Output for $\mathbb{R}$ lit
$y \neq 0$	$-\infty$	y^x	$0.$
	∞		∞
$0.$	$-\infty \leq x < 0$	y^x	∞
	$0 < x \leq \infty$		$0.$
	$-\infty < x < 0$	$\sqrt[x]{y}$	∞
	$0 \leq x < \infty$		$0.$
$0 \leq y \leq \infty$	$-\infty$ or ∞	$\sqrt[x]{y}$	$1.$
∞	$-\infty \leq x < 0$	y^x	$0.$
	$0 < x \leq \infty$		∞
	$-\infty < x < 0$	$\sqrt[x]{y}$	$0.$
	$0 \leq x < \infty$		∞
$0.$	$0 < x < \infty$	$\log_x y$	$-\infty$

Some particular returns of elementary transcendental functions ...

- operating at the edge of the *complex* plane at $\pm\infty$ (or close to it) or ...
- returning values at the edge ...

... are listed below. The left columns contain the input again: ¹⁴⁸

Cartesian		Polar		Op.	Output for $\mathbb{C}$ lit
Re(x)	Im(x)	r(x)	$\varphi(x)$		
$-\infty$	0	∞	180°	x^2	$(\infty \nless 360^\circ =) \infty + i \times 0.$
$0.$	∞	∞	90°		$(\infty \nless 180^\circ =) -\infty + i \times 0.$
$-\infty$	—	—		$\sqrt[x]{x}$	$(\infty \nless 90^\circ =) 0. + i \times \infty$
$-\infty$	0	∞	180°		
$-\infty$	—	—		x^3	$-\infty$
$-\infty$	0	∞	180°		$-\infty + i \times 0.$

¹⁴⁸ Actually, *complex* infinities should be treated in polar notation ($\rightarrow$ p. 27, left column, in <http://hparchive.com/Journals/HPJ-1984-07.pdf> for reasons).

Cartesian		Polar		Op.	Output for $\mathbb{C}$ lit
Re(x)	Im(x)	r(x)	$\varphi(x)$		
$-\infty$	—	—		$\sqrt[3]{x}$	$-\infty$
$-\infty$	0	∞	180°		$-\infty + i \times 0.$ ¹⁴⁹
-10^{999}		10^{999}			$(1. \times 10^{333} \angle 60^\circ =)$ $0.5 \times 10^{333} + i \times 0.866\ 025\ 404 \times 10^{333}$ $= 5 \times 10^{332} (1 + i \times \sqrt{3})$
$-\infty$	0	∞	180°	e^x	$0. + i \times 0.$
-10^{999}	10^{999}	10^{999}	135°		$0. + i \times 0.$
$-\infty$	∞	∞			NaN+i×NaN
0.	∞	∞	90°		NaN+i×NaN
∞	∞	∞	45°		$\infty + i \times 0.$
∞	0	∞	0°		NaN+i×NaN
∞	$-\infty$	∞	-45°		NaN+i×NaN
0.	$-\infty$	∞	-90°		NaN+i×NaN
$-\infty$	$-\infty$	∞	-135°		$0. + i \times 0.$
-10^{999}	-10^{999}	10^{999}			
$-\infty$	—	—			$\ln$
$-\infty$	0	∞	180°	$\infty + i \times 2.356\ 194\ 4 \dots = \infty + i\frac{3\pi}{4}$	
$-\infty$	∞	∞	135°	$\infty + i \times 1.570\ 796\ 3 \dots = \infty + i\frac{\pi}{2}$	
0.	∞	∞	90°	$\infty + i \times 0.785\ 398\ 1 \dots = \infty + i\frac{\pi}{4}$	
∞	∞	∞	45°		

¹⁴⁹ 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 generally stored in *Cartesian* coordinates in your WP43 (→ [Background](#) ...). Compare previous footnote though.

Cartesian		Polar		Op.	Output for $\mathbb{C}$ lit
Re(x)	Im(x)	r(x)	$\varphi(x)$		
∞	—	—		[In]	∞
∞	0	∞	0°		$\infty+i\times 0.$
∞	$-\infty$	∞	-45°		$\infty-i\times 0.785\ 398\ 1\dots = \infty - i\pi/4$
0.	$-\infty$	∞	-90°		$\infty-i\times 1.570\ 796\ 3\dots = \infty - i\pi/2$
$-\infty$	$-\infty$	∞	-135°		$\infty-i\times 2.356\ 194\ 4\dots = \infty - i\ 3\pi/4$
0.	0.	0.	0°		$-\infty+i\times 0.$
0.	—	—			$-\infty$

Computation of [lg] and lb x is derived from [ln]. The same applies in analogy for e^x , 10^x , and 2^x . – At the bottom line, we recommend keeping off $\pm\infty$ in *complex* plane.

Program Step Sizes

Program steps are 1 to 3 *bytes* long typically but may be significantly longer. Popular commands (like $1/x$, 2^x , **arccos**, **CF**, **DEC**, **ENTER↑**, **FILL**, **GT0**, **IP**, **ln**, **MOD**, **NEXTP**, **RTN**, **ST0**, **tan**, x^2 , y^x , **+**, $\sqrt{x}$, and almost all binary tests) take 1 *byte*, less frequently used ones take 2 *bytes* each. Commands with numeric parameters (like **RCL+ 02**), lettered *registers* (like $x < ? Y$), or *local labels* (like **LBL 66**) take one additional *byte* for the parameter; using indirect addressing costs another *byte* more.

Within programs, numeric constants (*reals*) take 4 to 18 *bytes* ($\rightarrow$ examples below). Maximum length of constants is 4 program lines. Constants recalled from **CONST** take 2 *bytes* only. *Short integers* take 11 *bytes*, *complex* constants 7 to 34 *bytes*. *Long integers* (like 42) take 4 to 416 *bytes* due to their length.

Global LBL steps specifying an *n*-letter *label* take $n + 3$ *bytes*. Same applies to commands operating on *variables*. Also pure *text strings* in programs consisting of *n* *characters* take $n + 3$ *bytes*. These statements hold for *characters* with their *Unicode* below $U+0080_{16}$ only; any *character* above requires 2 *bytes* ($\rightarrow$ DT 7 as described on p. 190). – Check the following **examples**:

GTO 17 is encoded in 2 *bytes* as follows:

- 1 *byte* for the GTO statement itself;
- 1 *byte* for the *local label* 17.

RCL+ →X is encoded in 3 *bytes*:

- 1 *byte* for the RCL+ statement itself;
- 1 *byte* signaling indirect access;
- 1 *byte* for the destination X (= R100).

LBL 'Prime' is encoded in 8 *bytes*:

- 1 *byte* for the LBL statement itself;
- 1 *byte* signaling a *global label*;
- 1 *byte* indicating the length of the *label* following (here: 5 usual letters);
- 5 *bytes* for these letters.

STO 'Count_1' is encoded in 10 *bytes*:

- 1 *byte* for the STO statement itself;
- 1 *byte* signaling a *variable*;
- 1 *byte* indicating the length of the *variable* following (7 usual letters);
- 7 *bytes* for these letters.

1e3 is encoded in 6 *bytes*:

- 1 *byte* signaling a numeric constant following;
- 1 *byte* telling its *DT* (*short real number*);
- 1 *byte* indicating its length (3);
- 3 *bytes* for the *characters* constituting the number.

-1.5 is encoded in 7 *bytes*:

- 1 *byte* signaling a numeric constant following;
- 1 *byte* telling its *DT* (*short real number*);
- 1 *byte* indicating its length (4);
- 4 *bytes* for the *characters*.

4.160 643 081 028 110 401 606 014 040 140 401 $\times 10^{-6083}$ (and any similarly big and /or precise *real number*) is encoded in 18 *bytes*:

- 1 *byte* signaling a numeric constant following;
- 1 *byte* telling its *DT* (*long real number*);
- 16 *bytes* = 128 *bits* for the number (→ 193ff for the representation).

8 07 06 05 04 03 02 01₁₆ is encoded in 11 *bytes*:

- 1 *byte* signaling a numeric constant following;
- 1 *byte* telling its *DT* (*short integer*);
 - 1 *byte* telling its display base (here 16);
 - 8 *bytes* for the digits (any *short integer* ≤ 64 *bits* can be stored therein).

-12 345 is encoded in 9 bytes:

- 1 byte signaling a numeric constant following;
- 1 byte telling its DT (shorter long integer);
- 1 byte indicating its length (6);
- 6 bytes for the sign and digits.

Here are all 122 commands taking 1 byte only:

- the four basic arithmetic operators +, -, *, and /,
- sin, sinh, cos, cosh, tan, tanh, and their inverses,
- the five basic flag commands CF, FC?, FF, FS?, and SF,
- the four basic Boolean operations AND, OR, XOR, and NOT,
- RCL, RCL+, RCL-, RCL*, RCL/, STO, STO+, STO-, STO*, and STO/,
- most stack commands: DROP↑, ENTER↑, FILL, R↑, R↓, x↑, and x↓y,
- all 8 commands of LOOP: DEC, DSE, DSL, DSZ, INC, ISE, ISG, and ISZ,
- 7 angular conversions: D→R, R→D, →DEG, →D.MS, →GRAD, →MULπ, and →RAD,
- 13 basic transcendental functions: 10^x, 2^x, $\sqrt[x]{x}$, e^x, lb, lg, ln, LOG_xy, x², x³, $\sqrt[y]{x}$, y^x, and $\sqrt{x}$,
- 9 basic programming commands: GTO, INPUT, LBL, PAUSE, ROUND, RTN, STOP, VIEW, and XEQ,
- 1/x, ceil & floor, CLX, COMB & PERM, FP & IP, GCD & LCM, IDIV, max & min, MOD & RMD, NEIGHB, NEXTP, x!, π, +/-, and |x|,
- 25 commands stored in TEST: CONVG?, CPX?, ENTRY?, EVEN?, FP?, INT?, KEY?, MATR?, M.SQR?, NaN?, ODD?, PRIME?, REAL?, SPEC?, STRI?, TOP?, and all 9 comparisons with x, as well as
- ±∞?

Program Files

Files written by WRITEP are ASCII files but not program text, similar to the .raw files used on DM42 (requiring an external converter to and from program text). They employ a binary format similar to the one used in the backup file, rather than a text format due to the constraints of translating all the non-ASCII

characters used on the WP43 (but → 346ff). This minimizes the effort in calculator software development and still allows an external converter as for the DM42, or an assembler/disassembler as for the WP 34S to be developed in a second step. This binary program file format includes the same version numbering as for the backup file.

The .p43 file format is described at begin of saveRestorePrograms.c :

```
// Structure of the program file.  
// Format: ASCII  
// Each line after line 6 contains the decimal value of a  
//                                     program byte  
// +-----+-----+  
// | Line| Content |  
// +-----+-----+  
// | 1  | "PROGRAM_FILE_FORMAT" |  
// | 2  | <file format> |  
// | 3  | "WP43_program_file_version" |  
// | 4  | <program file version> |  
// | 5  | "PROGRAM" |  
// | 6  | <program size> |  
// | 7  | <first program byte> |  
// | ... | ... |  
// | n  | <last program byte> |  
// +-----+-----+
```

APPENDIX C: MESSAGES AND ERROR CODES

There are some commands generating *temporary information* (*TI*, as specified in [OMS 2](#)), like 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, the operating system may throw one of the following error messages under the respective conditions (listed alphabetically – *EC* means *error code* here):

Error message	EC	Explanations, countermeasures, and examples
An argument exceeds the function domain	1	$\{1, 2, 3, 4, 10\}$ May be caused by roots of negative numbers or logs for $x \leq 0$ (for GPXRES), as well as by 0^0, $x/0$, $0/0$, $\Gamma(0)$, $\tan(\pm 90^\circ)$ and equivalents, $\operatorname{artanh}(x)$ for $\operatorname{Re}(x) \geq 1$, $\operatorname{arcosh}(x)$ for $\operatorname{Re}(x) < 1$, etc.¹⁵⁰
Bad time or date input	2	$\{2, 5, 6\}$ Invalid date format or incorrect <i>date</i> or <i>time</i> in input, like month > 12, day > 31, or <i>date-time</i> exceeding the range limits ($\rightarrow$ <i>Limitations</i> on pp. 197ff). Will be thrown as soon as input is closed.
Cannot delete a pre-defined item	27	Self-explanatory
Cannot read file	35	Futile attempt to Load... or READP something from FM.
Cannot write file	53	Futile attempt to SAVE, SAVEST, or WRITEP to FM.

¹⁵⁰ Note that, e.g., $\tan(90^\circ)$ and logs of 0 are legal operations on $\{1, 2, 3\}$ if SPCRES is set. $\rightarrow$ end of this appendix.

Error message	EC	Explanations, countermeasures, and examples
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	Regain space by deleting a file from <i>FM</i> .
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.
Input data types do not match	31	Attempt to operate on different <i>DTs</i> (e.g., for a <i>Boolean</i> operation: 10. AND 11).
Invalid input data type for this operation	24	Convert what is necessary, if possible (→ also Operation is undefined in this mode). But this error may also appear in attempts to calculate with <i>configuration</i> data.
Invalid name	48	Attempt to create a <i>variable name</i> which does not comply to the rules.
Invalid or corrupted data	18	Thrown when there is a checksum error in <i>FM</i> or a <i>FM</i> segment is not usable otherwise.
Item to be coded	29	May be thrown during firmware development.
I/O error	17	→ OMS 4 .
Large Δ and opposite signs, may be a pole	41	The <i>Solver</i> can't find a root but diverging points of opposite sign. This may point to a pole.
Matrix mismatch	21	{8, 9} • A matrix is not square although it should be. • Matrix sizes are not miscible.
May be calculatory correct but meaningless	71	{1, 2, 3, 4} Returned for inputs to sin , cos , tan exceeding 10^{33} for reasons explained under <i>Background ... Angles</i> . Actually this is just a warning.

Error message	EC	Explanations, countermeasures, and examples
No errors calculable for this fit model	40	Thrown by s(a) if a model except LinF , ExpF , Powerf , LogF , or OrthoF was chosen.
No global label in this program	51	Self-explanatory.
No matrix indexed	38	Returned by RCLEL , RCLIJ , STOEL , or STOIJ if there is no <i>indexed matrix</i> .
No root found	20	The <i>Solver</i> did not converge. Check, e.g., the HP-34C OH and Programming Guide for advice.
No program specified	54	... for solving or integrating.
No statistic data present	28	Attempt to address statistics storage not allocated yet.
No such function	7	Thrown when calling a nonexistent function, e.g., via XEQ α ... ENTER (check for typos!). May also be found in equations or after coding on <i>PC</i> , where you are free to enter anything – and your <i>WP43</i> can find errors no earlier than when executing your code.
No such label found	6	Attempt to address an undefined <i>label</i> .
No such local register allocated	58	Self-explanatory.
Non-programmable command, please remove	50	Self-explanatory. Such commands can't be entered in <i>PEM</i> on your <i>WP43</i> but may have slipped into a routine otherwise, e.g., when coding on <i>PC</i> .
Not enough memory for such a matrix	39	Will be thrown by M.NEW or M.DIM if allocating a too large matrix is attempted – or if space is insufficient for saving or recalling STATS .
Not enough memory to load the equations	57	Will be thrown by LoadAll or LoadEq or LoadPr if <i>RAM</i> doesn't suffice.
... load the programs	55	

Error message	EC	Explanations, countermeasures, and examples
Operation aborted	56	... by pressing EXIT while FACTOR , NEXTP , the <i>Integrator</i> , or the <i>Solver</i> were running.
Operation is undefined in this mode	13	Caused, e.g., by calling a <i>real-number</i> operation in <i>A/M</i> . Compare also Illegal input data type for this operation .
Out of range	8	{1, 2, 3, 10} • A number exceeds the valid range. This can be caused by indirectly specifying <i>word size</i> > 64 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 (like R-COPY) attempts accessing invalid <i>register</i> addresses.
Output would exceed 196 characters	33	{7} Maximum length of <i>text strings</i> .
Overflow at $+\infty$	4	{1, 2, 3, 8, 9} unless SPCRES is set • Division of a positive number by 0. • Divergent sum, product, integral, derivative. • Positive overflow ($\rightarrow 193\text{ff}$).
Overflow at $-\infty$	5	{1, 2, 3, 8, 9} unless SPCRES is set • Division of a negative number by 0. • Divergent sum, product, integral, derivative. • Negative overflow ($\rightarrow 193\text{ff}$).
Please enter a NEW name	26	Attempt to define a new <i>variable</i> or user <i>menu</i> taking a <i>name</i> already in use (<i>name conflict</i>).

Error message	EC	Explanations, countermeasures, and examples
RAM does not suffice	11	... to provide the space required. May be caused by attempts to load or write too big routines or matrices, allocate too many <i>variables</i> , etc. (→ 189ff for space needed for various <i>DTs</i>). May also happen in program execution due to dynamic allocations (→ OMS 4).
Singular matrix	22	{8, 9} - Attempt to use an <i>LU</i> decomposed matrix for solving a system of equations. - Attempt to invert a matrix which isn't of full rank (a.k.a. irregular or degenerate matrix).
Solver reached a local extremum, no root	42	The <i>Solver</i> can't find a root since it got trapped at a local minimum or maximum.
Stack clash	12	STOSTK or RCLSTK attempts accessing <i>registers</i> which would overlap the <i>stack</i> (→ OMS 1). Will happen, e.g., with SSIZE8 and STOSTK 93 .
Syntax error in this equation	45	An equation entered in <u>EQN</u> contains, e.g., implicit multiplication or parentheses not matching or unallowed <i>characters</i> .
The function values look constant	44	The <i>Solver</i> can't find a root since the function seems to be constant (and never crosses zero).
This does not work with an empty string	34	{7} Self-explanatory.
This equation formula is too complex	46	... since it exceeds 9 levels (parentheses, pending operations).
This item cannot be assigned here	47	You may assign an <i>item</i> to a location in <u>MyMENU</u> , <u>MyPFN</u> , <u>Myα</u> , or in a user-created <i>menu</i> only. But you cannot assign <u>new submenus</u> to any of these.
This system flag is write protected	32	Self-explanatory.

Error message	EC	Explanations, countermeasures, and examples
This variable is write protected	37	Self-explanatory.
Too few data points for this statistic	15	A statistical calculation was attempted with too few data, e.g., mean or <i>SD</i> for < 2 points, regression for < 2 or 3, or <i>s_{mi}</i> for < 30 points.
Too many variables	49	Attempt to create a 1001 st <i>variable</i> .
Undefined op-code	3	An instruction with an undefined operation code occurred. (Should never happen.)
Undefined source variable	36	Attempt to recall a <i>variable</i> not defined yet. (That should not occur anymore since such a <i>variable</i> should be created automatically.)
Word size is too small to take this	14	{10} User input or <i>register</i> content is too great to be handled by the <i>word</i> size currently set.
≥ 1 initial guess lies out of the domain	43	<i>Solver</i> error: One or both initial guesses lie out of the function domain.
	25	Left unused for <i>WP 34S</i> compatibility
	10, 19, 30	Left unused since employed earlier in this project

With *SPCRES* set, errors 4 and 5 will vanish, and error 1 will happen less frequently, since $\pm\infty$ and **NaN** are legal results then (→ corresponding entries in *CONST* on pp. 116ff and the table on p. 196). E.g., **0** **(In)** will return $-\infty$ then.

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

There are at least two files in *WP43* which may send *TI* to the screen looking like an error message: *display.c* and *screen.c*. Such *TI* will appear in the *X* or *Y* numeric row, however, and should not occur in regular operation. If you observe a message not listed above, please report.

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 *WP43*. The tables below revert this view. The first table shows the functions of the *HP-42S* and the corresponding ones of your *WP43* 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 the *HP-16C* functions starting on p. 221, one for the *HP-21S* on p. 223, and another one for the *WP 34S* on p. 225. Functions newly introduced with *WP* calculators are compiled on pp. 230ff.

Functional differences of homonymous commands are covered in the *IOI* (→ 14ff).

Corresponding Operations on HP-42S

Remarks printed on light grey indicate commands being either default settings or keyboard accessible on your *WP43* while you must use a menu on the *HP-42S*.

<i>HP-42S</i>	<i>WP43</i>	Remarks
ACOSH	arcosh	In <u>EXP</u> and <u>TRI</u>
ADV	 ADV	In PRINT ()
AIP	Dispensable	Use  to merge text and numeric data easily (→ <i>OMS 3</i>).
ALENG	αLENG	In <u>α.FN</u>
ALLΣ	Dispensable	Your <i>WP43</i> runs in ALLΣ mode always.
ALPHA		→ description of <i>AIM</i> in <i>OMS 3</i> .
AOFF	CF ALPHA	
AON	SF ALPHA	
APOS	αPOS	In <u>α.FN</u>
ARCL	Dispensable	Every <i>RoV</i> can hold a <i>text string</i> . Use RCL .

HP-42S	WP43	Remarks
AROT	α RL or α RR	In <u>α.FN</u>
ASHF	α SL or α SR	
ASINH	arsinh	In <u>EXP</u> and <u>TRI</u>
ASTO	Dispensable	Every RoV can hold a <i>text string</i> . Use [STO] .
ATANH	artanh	In <u>EXP</u> and <u>TRI</u>
ATOX	$\alpha \rightarrow x$	In <u>α.FN</u>
[BASE]	[INTS] or [BIT]	
BASE+	Dispensable	Your WP43 executes these arithmetic commands automatically for <i>short integer</i> inputs.
BASE-		
BASE×		
BASE÷		
BASE+/-		
BINM	Dispensable	Press [#] [2] for converting any closed <i>integer number</i> or integer part in <i>x</i> to binary.
BIT?	BS?	In <u>BIT</u>
[BST]	[≡Δ] ([▲])	The shortcut works if no <i>multi-view menu</i> is open.
CLA	0 [STO] [K]	
CLD	Dispensable	Any keystroke will clear <i>temporary information</i> .
[CLEAR]	[CLR]	
CLRG	CLREGS	In <u>CLR</u>
CLST	CLSTK	Press [0] [FILL] in <i>RUM</i> .
CLV	→ remark	<i>Variables</i> are cleared or deleted as specified in <i>OMS 7</i> .
[COMPLEX]	[CC]	You can also enter <i>complex numbers</i> directly using [CC] (→ <i>OMS 3</i>).
[CONVERT]	[↔] & [PART]	

HP-42S	WP43	Remarks
CUSTOM	n/a	Create as many <i>user menus</i> as memory will hold – not just one CUSTOM menu (→ OMS 7).
DECM	Dispensable	Every input featuring one $\square$ or E is interpreted as a <i>real</i> (decimal) number.
DEL	Not featured.	Too dangerous, in our opinion.
DELAY	DELAY	In PRINT (PRINT)
DELR	M.DELR	Label DELR in the <i>Matrix Editor</i>
DET	$ M $	_____ in MATX
DIM	M.DIM	Label DIM _____ in MATX
DIM?	?M.DIM	Label ?DIM in INFO and MATX
EDIT	M.EDI	Label EDIT _____ in MATX
EDITN	M.EDIN	Label EDITN
FCSTX	$\hat{x}$	In FIT
FCSTY	$\hat{y}$	
FNRM	ENORM	In MATX . <i>Euclid</i> is older than <i>Frobenius</i> .
GAMMA	$\Gamma(x)$	In PROB ; can be called via GAMMA as well.
GETKEY	KEY?	In P.FN
GETM	M.GET	Label GETM in MATX
GROW	M.GROW	Label GROW in the <i>Matrix Editor</i>
HEXM	Dispensable	Press #H for converting any closed <i>integer number</i> or integer part in x to hexadecimal.
H.MS+	Dispensable	Press $\text{+}/\text{-}$ with sexagesimal <i>times</i> in x and y .
H.MS–		
INSR	M.INSR	Label INSR in the <i>Matrix Editor</i>
INTEG	$\int$	In ADV
INVRT	$[M]^{-1}$	In MATX

HP-42S	WP43	Remarks
KEYASN	Dispensable	Obsolete since no CUSTOM menu is featured (→ CUSTOM).
LASTx	RCL L	
LBL	LBL	
LCLBL	Dispensable	Obsolete since no CUSTOM menu is featured (→ CUSTOM). Nevertheless, your WP43 provides <i>local labels</i> (→ OMS 4).
LINΣ	Dispensable	Your WP43 runs in ALLΣ mode always.
LIST	n/a	Use  PROG instead.
LOG	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	MOD	
MODES	MODE	
N!	x!	
NEWMAT	M.NEW	Label NEW in <u>MATX</u>
NORM	Not featured.	
OCTM	Dispensable	Press   for converting any closed <i>integer number</i> or integer part in <i>x</i> to octal.
ON	n/a	Programmable ON is not featured.
PGM.FCN	P.FN	    are on the keyboard.
POSA	αPOS	In <u>α.FN</u>
PRA	 K	In <u>PRINT</u> 
PRGM	P/R	
PRLCD	SNAP	 dumps the entire <i>LCD</i> onto the <i>FD</i> . The <i>LCD</i> of your WP43 is too large to be printed on either <i>HP</i> printer.

HP-42S	WP43	Remarks
PROFF	CF 	
PRON	SF 	
PRP	 PROG	In <u>PRINT</u> ()
PRSTK	 STK	
PRUSR	 USER	
PRV	 r	
PRX	 x	
PRΣ	 Σ	
PUTM	M.PUT	Label PUTM in <u>MATX</u>
PWRF	PowerF	In <u>FIT</u>
RAN	RAN#	In <u>PROB</u>
RND	ROUND	In <u>PART</u>
RNRM	RNORM	In <u>MATX</u>
ROTXY	RL, RLC, RR, and RRC	In <u>BIT</u>
RTN		
SDEV	s	In <u>STAT</u>
SIZE	Dispensable	There are 100 numbered global <i>GP registers</i> available always – plus up to 99 local ones.
SLOPE	L.R.	In <u>FIT</u>
	 ()	Shortcut works if no <i>multi-view menu</i> is open.
STR?	STRI?	In <u>TEST</u>
	Dispensable	... since no top row functions can be overwritten.
	SF 	
TRANS	[M] ^T	In <u>MATX</u>
UVEC	UNITV	In <u>MATX</u> and <u>CPX</u>

HP-42S	WP43	Remarks
VARMENU	VARMNU	Truncated to 6 <i>characters</i> to fit the <i>menu</i> space.
VIEW		
WMEAN	$\bar{x}_w$	In <u>STAT</u>
WRAP	M.WRAP	Label WRAP in the <i>Matrix Editor</i>
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>FIT</u>
Σ REG	Dispensable	Statistical summation <i>registers</i> on your WP43 do not overlap with <i>GP registers</i> .
Σ REG?		
$\rightarrow$ DEC	$\rightarrow$ INT 10 or $\rightarrow$ DEC	... for closed <i>x</i> .
$\rightarrow$ HR		
$\rightarrow$ H.MS		
$\rightarrow$ OCT	$\rightarrow$ INT 8 or $\rightarrow$ OCT	
$\rightarrow$ POL		
$\rightarrow$ REC		
%CH		
		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. Note your WP43 can do everything a HP-16C can do, at least.

HP-16C	WP43	Remarks
RL , RLn	RL	In <u>BIT</u>
RR , RRn	RR	
RLC , RLCn	RLC	
RRC , RRCn	RRC	
÷	/	(→ ISO 80000-2: “The symbol ÷ should not be used.”)
DBL÷	DBL/ /	In <u>INTS</u>
x>=i	Dispensable	Every <i>register</i> may be used for indirection.
x>=l		
SHOW HEX	n/a	Just convert instead.
SHOW DEC		
SHOW OCT		
SHOW BIN		
B?	BS?	In <u>BIT</u>
GSB	XEQ	
HEX	#H or →HEX	
DEC	#D or →DEC	
OCT	#8 or →OCT	
BIN	#2 or →BIN	
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.

Unshifted SF and CF live in FLAG and MODE.

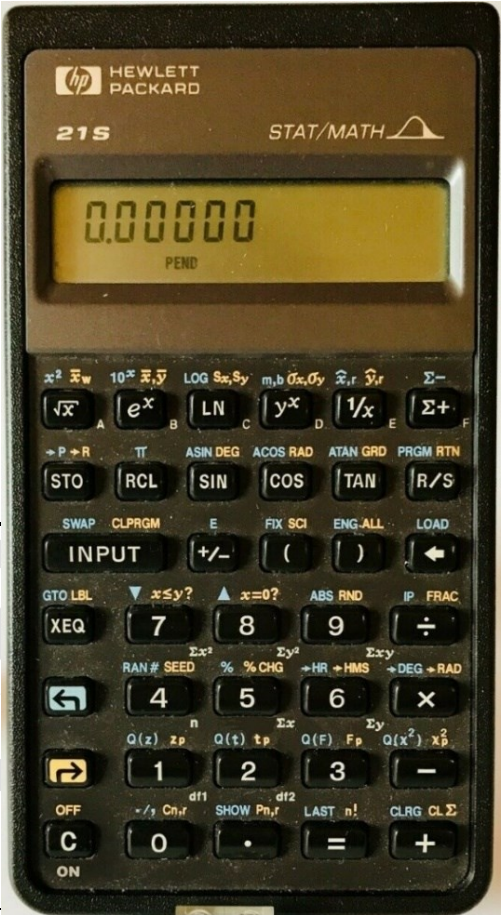
HP-16C	WP43	Remarks
F?	FS?	In <u>FLAG</u>
(i)	Dispensable	Every RoV may be used for indirection.
I		
CLEAR PRGM	CLP	In <u>CLR</u> . Note here is also CLPALL.
CLEAR REG	CLREGS	In <u>CLR</u>
CLEAR PREFIX	Dispensable	  or   will do.
WINDOW	Dispensable	All 64 <i>bits</i> can be displayed in one row.
SET COMPL 1S	1COMPL	In <u>MODE</u> and <u>BIT</u> . Note here is also SIGNMT.
SET COMPL 2S	2COMPL	
SET COMPL UNSGN	UNSIGN	
SST	 ()	 works if no <i>multi-view menu</i> is open.
BSP		
BST	 ()	 works if no <i>multi-view menu</i> is open.
x≤y	x≤?	In <u>TEST</u> . Note far more tests are featured on your WP43.
x<0	x<?	
x>y , x>0	x>?	
FLOAT	.d	
MEM	STATUS	In <u>FLAG</u>
CHS	+/-	
< , >	Dispensable	All 64 <i>bits</i> can be displayed in one row.
LSTX	RCL L	
x≠y , x≠0	x≠?	In <u>TEST</u> . Note far more tests are featured on your WP43.
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](#). That is, however, an algebraic calculator; hence its keys **INPUT**, **(**, **)**, and **=** have no direct equivalents on your [WP43](#).

Consult the [HP-21S OM](#) for additional information about the four most important continuous statistical distributions and their applications.

HP-21S	WP43	Remarks
(x̄w)	x̄w	In <u>STAT</u> or <u>FIT</u>
(x̄.y)	x̄	
(sx.sy)	s	
(m.b)	L.R.	
(σx.σy)	σ	
(x̂.r)	r, x̂	
(ŷ.r)	r, ŷ	
PRGM	P/R	
SWAP	x↔y	
CLPRGM	CLP	In <u>CLR</u>
INPUT	Dispensable	This functionality is contained in ENTER↑. Also your WP43 features a command called INPUT but this works in programs.
(,)	Dispensable	You can forget those keys in <i>RPN</i> .
LOAD	n/a	Loads predefined programs in the HP-21S .



HP-21S	WP43	Remarks
(ABS)	(x)	
(RND)	ROUND	In <u>DISP</u>
(FRAC)	FP	In <u>PART</u>
(÷)	(/)	→ ISO 80000-2: "The symbol ÷ should not be used."
(SEED)	SEED	In <u>PROB</u>
(%CHG)	(Δ%)	
(Q(z))	Norml_Δ	In <i>submenus</i> of <u>PROB</u>. Note your WP43 features the <i>normal distribution</i> for arbitrary μ and σ instead of just the <i>standardized</i> one ($\mu=0$, $\sigma=1$) provided in the HP-21S. And the implementation of the <i>inverse</i> distribution functions (a.k.a. <i>quantile functions</i>) deviates on both calculators: your WP43 calculates with the probability P while the HP-21S calculates with the error probability $Q = 1 - P$ as input (→ OMS 2 and footnotes in the <i>IOI</i>). Labeling these functions using the letter p on the HP-21S may add some confusion.
(z_p)	Norml⁻¹	
(Q(t))	t_Δ(x)	
(t_p)	t⁻¹(p)	
(Q(F))	F_Δ(x)	
(F_p)	F⁻¹(p)	
(Q(χ²))	χ²_Δ(x)	
(χ²_p)	(χ²)⁻¹	
(Cn.r)	COMB	In <u>PROB</u>
(Pn.r)	PERM	
(LAST)	(RCL) (L)	
(=)	Dispensable	You can forget that key in <i>RPN</i> .
(n!)	(x!)	
(CLRg)	CLREGS	In <u>CLR</u>

Corresponding Operations on WP 34S

WP 34S and **WP43** share over 90% of their function sets (→ also next chapter). Most of the discrepancies between both calculators are caused by their different displays: your **WP43** allows for 18 *softkeys* – **WP 34S** can only carry four *hotkeys* instead. As mentioned during **WP 34S** development more than once, that project challenged by the low display performance and little memory of the **HP-20b/-30b** – what was achieved was just the optimum under those boundary conditions, making the **WP 34S** a landmark in 2011.

It was our objective that your **WP43** shall meet or exceed the performance of the **WP 34S** in every aspect. Dealing with matrices, e.g., is distinctly eased by the large high resolution *LCD* of your **WP43** now; thus some elementary matrix commands of **WP 34S** are not required anymore on your **WP43**.

Remarks printed on light grey indicate commands being either default settings or obsolete on your **WP43** while you have to use them on the **WP 34S**.

WP 34S	WP43	Remarks
ANGLE	∠	
BATT?	?BATT	
Binom	Binom _▲	
Binom _u	Binom _▲	
Cauch	Cauch _▲	
Cauch _u	Cauch _▲	
CLα	0 [STO] [K]	Check the <i>OM</i> for the conditions when this <i>register</i> is used.
CONV	[U→]	
DBLOFF	Dispensable	Your WP43 features <i>34-digit reals</i> per default – no need for a <i>double precision</i> mode anymore.
DBLON		
Expon	Expon _▲	
Expon _u	Expon _▲	
F(x)	F _▲ (x)	

WP 34S	WP43	Remarks
$F_u(x)$	$F_{\Delta}(x)$	
dRCL	Dispensable	→ DBLOFF.
gCLR, gDIM, gDIM?, gFLP, gPIX?, gPLOT, gSET	n/a	The LCD of your WP43 features 240×400 px compared to 6×43 px of HP-30b – the graphic paradigm of WP 34S makes no sense on your WP43 anymore. On the other hand, it was not our objective designing a graphing calculator. Thus, we include just the basic graphic support of HP-42S plus ASSESS, CENTRL, HIST, HNORM, HPLOT, POINT, and PLOT.
Geom	Geom Δ	
Geom _u	Geom Δ	
GTO α	Dispensable	Use GTO with an appropriate parameter instead.
H.MS+, H.MS–	Dispensable	Your WP43 features a dedicated DT for times, so $\oplus$ and $\ominus$ suffice for adding and subtracting sexagesimal times.
INTM?	Dispensable	Your WP43 features two dedicated DTs for integers – no integer mode needed anymore.
iRCL	Dispensable	Your WP43 features various DTs.
I_x	I_{xyz}	This is a triadic function after all.
Lgnrm	LgNrm Δ	
Lgnrm _u	LgNrm Δ	
L_n	L_m	Renamed to avoid a search conflict with $\ln$.
$L_{n\alpha}$	$L_{m\alpha}$	Renamed in consequence to L_m .
Logis	Logis Δ	
Logis _u	Logis Δ	
MEM?	?MEM	
MROW+x, MROWx	Dispensable	Obsolete matrix commands.

WP 34S	WP43	Remarks
MROW↵	M.R↵R	Label R↵R in <u>MATX</u>
M+×	Dispensable	Obsolete matrix command.
M^{-1}	[M]⁻¹	
M-ALL, M-COL, M-DIAG, M-ROW	Dispensable	Obsolete matrix commands.
M×	Dispensable	Your <i>WP43</i> features two dedicated <i>DTs</i> for matrices. Thus, you can multiply and copy matrices just like any other objects.
M.COPY		
M.IJ, M.REG	Dispensable	Obsolete matrix commands.
nBITS	#B	
nCOL, nROW	Dispensable	Obsolete matrix commands.
Norml	Norml ▲	
Norml _u	Norml ▲	
Poiss	Poiss ▲	
Poiss _u	Poiss ▲	
REALM?	Dispensable	Your <i>WP43</i> features a dedicated <i>DT</i> for <i>real numbers</i> – no need for a <i>real</i> mode anymore.
RECV	Replaced by READP	Exchanging programs via the <i>FD</i> is easier.
REGS, REGS?	Dispensable	There are 100 numbered global <i>GP registers</i> on your <i>WP43</i> always.
RM?	?RM	
SENDA, SENDP, SENDR, SENDS	Replaced by WRITEP	Exchanging programs via the <i>FD</i> is easier.

WP 34S	WP43	Remarks
SEPOFF, SEPON	GAP	
SHOW	RBR	
sRCL	Dispensable	→ DBLOFF.
SSIZE?	?SSIZE	
TRANSP	[M]^T	
TSOFF	GAP 0	
TSON	GAP 3	
t(x)	t_Δ(x)	
t _u (x)	t_Δ(x)	
VIEW _α , VW _{α+}	Dispensable	Use VIEW or AVIEW instead; <i>text strings</i> are just another <i>DT</i> . Combine text and numeric data easily as shown in OMS 3 .
Weibl	Weibl_Δ	
Weibl _u	Weibl_Δ	
XEQ _α	Dispensable	Use XEQ with an appropriate parameter instead.
XTAL?	Dispensable	A quartz crystal is installed by default.
YDOFF, YDON	Dispensable	Your <i>WP43</i> displays <i>y</i> whenever requested and possible (→ DSTACK).
αDATE, αDAY	Dispensable	You can combine text and numeric data easily as shown in OMS 3 .
αGTO	Dispensable	Use GTO with an appropriate parameter instead.
αIP, αMONTH	Dispensable	→ αDATE.
αRCL, αRC#	Dispensable	Your <i>WP43</i> features various <i>DTs</i> and 'knows' which type is in the <i>RoV</i> specified.

WP 34S	WP43	Remarks
αSTO	Dispensable	Simply press [STO] instead (every <i>register</i> can take a <i>text string</i>).
αTIME	Dispensable	→ αDATE .
αXEQ	Dispensable	Use [XEQ] with an appropriate parameter instead.
β	$\beta(x,y)$	
Γ	$\Gamma(x)$	
ΔDAYS	Dispensable	Simply subtract two <i>dates</i> .
ζ	$\zeta(x)$	
$\Phi(x) \dots$	Dispensable	Use NORML... with $\mu=0$ and $\sigma=1$ instead.
$\chi^2(x)$	$\chi^2_{\Delta}(x)$	
$\chi^2_u(x)$	$\chi^2_{\Delta}(x)$	
→H	→HR	
▢PLOT	n/a	→ gCLR .
▢C_{r_{XY}}	Dispensable	Use ▢r instead.
▢α, ▢$\alpha+$, ▢$+\alpha$	Dispensable	Combine text and numeric data as shown in OMS 3. Then use ▢r .
▢?	Dispensable	A quartz crystal and the proper firmware for printing are installed by default.

New Commands on your WP43

The following table lists the commands and pseudo-commands created for your WP43 (and for preceding WP calculators, if applicable), offering new or extended functionality compared to earlier HP RPN and algebraic pocket calculators. These are some 390 operations (not counting the unit conversions and constants provided); 145 of them are even new and 19 extended compared to earlier WP calculators. The commands are printed below as spelled on your WP43. Commands for probability and statistics are printed on blue background, bit and integer commands on light green.

Command	WP43	WP 31S	WP 34S
2 ^x AGM	●	—	new
ALL	●	●	extended
AND ASR	●	—	extended
ASSESS atan2	new	—	—
BACK	●	—	new
BC?	●	—	new
BestF	extended	●	●
Binom _p Binom _Δ (of binomial distribution)	●	●	new
B _n B _n *	●	—	new
BUZZ CALC	new	—	—
CASE	●	—	new
CauchF Cauch _p Cauch _Δ Cauch _Δ Cauch ⁻¹	new	—	—
ceil	●	—	new
CENTRL CLBKUP CLCVAR	new	—	—
CLFall	●	—	new
CLKEYS CLMall CLMyM CLMyP CLMyα	new	—	—
CLPall	●	—	new
CLVall cn(u,m)	new	—	—
conj CONVG? COV	●	—	new

Command	WP43	WP 31S	WP 34S
CX→RE	new	—	—
DATE	●	—	(●)
DATE→ DAY DEC	●	—	new
DECOMP	●	●	new
DEG→	●	—	new
DelITM DelMall DelVall dn(u,m) Draw	new	—	—
DROP↓ (returned after 50 years, → App. H)	●	—	back
DROPy DSL DSTACK DUMP	new	—	—
D.MS→ D→J	●	—	new
D→MIL EDIT EIGVAL EIGVEC	new	—	—
ENTRY?	●	—	new
EQ.DEL EQ.EDI EQ.NEW	new	—	—
erf erfc ERR EVEN	●	—	new
Expon _p Expon _Δ Expon _Δ Expon ⁻¹	●	●	new
EXPT	●	—	new
E(m) E(φ,m) FACTOR	new	—	—
FB	●	—	new
FBR	new	—	—
FC?F FC?S FF FIB	●	—	new
FILL	●	●	new
floor FP?	●	—	new
F _p (x) F _Δ (x) (of F distribution)	●	●	new
FS?F FS?S	●	—	new
F(φ,m)	new	—	—
f'(x) f''(x)	extended	—	new
GAP	extended	●	new
GaussF	new	—	—

Command	WP43	WP 31S	WP 34S
GCD	●	●	new
g_d g_d^{-1} Geom_p Geom_Δ Geom_Δ Geom⁻¹ GRAD→ H_n H_{np}	●	—	new
HIDE HISTOX HISTOY HNORM HPLOT Hyper_p Hyper_Δ Hyper_Δ Hyper⁻¹ HypF	new	—	—
IDIV	●	—	new
IDIVR Im	new	—	—
INC INT? ISE I_{xyz}	●	—	new
$J_y(x)$	new	—	—
J/G	extended	●	new
J→D KEY? ?KTYF	●	—	new
K(m)	new	—	—
lb LBL?	●	—	new
LCM	●	●	new
LEAP? LgNrm_p LgNrm_Δ LgNrm_Δ LgNrm⁻¹ L_m $L_{m\alpha}$ $\ln\beta$ $\ln\Gamma$	●	—	new
LoadAll	●	●	new
LoadEq LoadPr LoadR LoadSS	●	—	new
LoadV	new	—	—
LoadΣ LocR	●	—	new
Logis_p Logis_Δ Logis_Δ Logis⁻¹	●	●	new
$\log_{xy}$	●	—	new
MAD	new	—	—
MANT max	●	—	new
MEDIAN MIL MIL→ MIL→D MIL→R	new	—	—
min MIRROR	●	—	new
MOD	●	●	new

Command	WP43	WP 31S	WP 34S
MONTH MSG	●	—	new
MUL π MUL $\pi \rightarrow$ M $\pi \rightarrow$ R M.C $\rightarrow$ C	new	—	—
M.LU M.SQR? NAND NaN?	●	—	new
NBin _p NBin _Δ NBin _Δ NBin ⁻¹	new	—	—
NEIGHB	●	—	new
NEXTP	extended	●	new
NOR	●	—	new
Norml _p Norml _Δ Norml _Δ Norml ⁻¹	●	●	new
NOT	●	—	extended
NXTFIT	new	—	—
n Σ (called by name)	●	●	new
ODD?	●	—	new
PLAY PLOT POINT	new	—	—
Poiss _p Poiss _Δ Poiss _Δ Poiss ⁻¹	●	●	new
PopLR	●	—	new
PRIME?	extended	●	new
PUTK	●	—	new
Q ₃ -Q ₁	new	—	—
RAD $\rightarrow$	●	—	new
RANGE RANI#	new	—	—
RBR	●	●	new
RCLCFG	extended	—	new
RCLSTK RCLMAX RCLMIN RDP	●	—	new
Re READP REDRAW RE $\rightarrow$ CX Re $\rightarrow$ Im	new	—	—
RJ	●	—	new
RL RLC	●	—	extended
RM	●	—	new

Command	WP43	WP 31S	WP 34S
RMD	●	●	extended
RootF	new	—	—
ROUNDI	●	—	new
RR RRC	●	—	extended
RSD R-CLR R-COPY R-SORT R-SWAP RTN.SKIP	●	—	new
R→MIL R→Mπ	new	—	—
SAVE	●	●	new
SAVEST	new	—	—
SDL SDR SETCHN	●	—	new
SETDAT	●	—	(●)
SETEUR SETIND SETJPN	●	—	new
SETSIG	new	—	—
SETTIM	●	—	(●)
SETUK SETUSA SIGNMT sinc	●	—	new
sincπ	new	—	—
SKIP	●	—	new
SL	●	—	extended
SLVC	new	—	—
SLVQ	●	—	new
s _m s _{mω}	●	●	new
s _{mi} SNAP sn(u,m)	new	—	—
SPEC?	●	—	new
SR	●	—	extended
STATUS	extended	—	extended
STOCFG	extended	—	new
STOSTK STOMAX STOMIN	●	—	new

Command	WP43	WP 31S	WP 34S
S_w	●	●	new
S_{xy}	●	—	new
s(a) TDISP	new	—	—
TICKS	●	—	new
TIME TIMER	●	—	(●)
T_n TOP?	●	—	new
$t_p(x)$ $t_{\Delta}(x)$ (of t distribution)	●	●	new
$t_{\frac{1}{2}}$ ULP U_n	●	—	new
UNDO	●	new	—
VERS	●	●	new
VOL VOL+ VOL- $V_{\frac{1}{2}}$	new	—	—
WDAY Weib _p Weib _{Δ} Weib _{Δ} Weib ⁻¹ WHO	●	●	new
W_m W_p	●	—	new
WRITEP	new	—	—
W^{-1}	●	—	new
XEQ.SKIP	new	—	—
$\bar{x}_G$	●	—	new
$\bar{x}_H$ $x_{\max}$ $x_{\min}$	new	—	—
XNOR	●	—	new
XOR	●	—	extended
$\bar{x}_{RMS}$ $x \rightarrow \text{DATE}$	new	—	—
$x < ?$ $x \leq ?$ $x = ?$ $x \neq ?$ $x \geq ?$ $x > ?$	extended	—	extended
$x = +0?$ $x = -0?$ $x \approx ?$ YEAR	●	—	new
y^x	extended	●	●
$Y_y(x)$	new	—	—
Y.MD	●	●	new

Command	WP43	WP 31S	WP 34S
y^z	●	—	new
ZOOM	new	—	—
z^z	●	—	new
α LENG	extended	—	●
α POS	extended	—	—
α RL α RR α SL α SR	●	—	extended
$\beta(x,y)$ Γ_p Γ_q Γ_{xy} γ_{xy}	●	—	new
Δx	new	—	—
ε ε_m ε_p $\zeta(x)$	●	—	new
$Z(\varphi,m)$	new	—	—
Π_n	●	—	new
$\Pi(u,m)$ Σ^1/x Σ^1/x^2 Σ^1/y Σ^1/y^2	new	—	—
$\Sigma \ln^2 x$ $\Sigma \ln^2 y$ $\Sigma \ln x$ $\Sigma \ln xy$ $\Sigma \ln y$ (called by names)	●	●	new
$\Sigma \ln y/x$	new	—	—
Σ_n σ_w	●	—	new
Σx Σx^2 $\Sigma x^2 y$ (called by names)	●	●	new
$\Sigma x^2 \ln y$ $\Sigma x^2/y$ Σx^3 Σx^4	new	—	—
$\Sigma x \ln y$ Σxy (called by names)	●	●	new
$\Sigma x/y$	new	—	—
Σy Σy^2 $\Sigma y \ln x$ (called by names)	●	●	new
$\chi^2_p(x)$ $\chi^2_{\Delta}(x)$ (of chi-square distribution)	—	—	—
$\psi(u,m)$	new	—	—
$(-1)^x$	●	—	new
$\pm \infty?$	new	—	—
$\times$ MOD $\wedge$ MOD	●	—	new
?BATT	●	●	new

Command	WP43	WP 31S	WP 34S
?BestF ?DISK ?HIDE ?ISM ?J/G	new	—	—
?LocR	●	—	new
?RANGE	new	—	—
?RM	●	—	new
?SDIGS	new	—	—
?SSIZE	●	●	new
?VOL	new	—	—
?WSIZE →DATE	●	—	new
→DEG	●	●	new
→D.MS	new	—	—
→GRAD	●	—	new
→INT →MIL →MUL π	new	—	—
→RAD	●	●	new
→REAL	new	—	—
↗	●	—	new
	●	●	new
%ile	new	—	—

The statements in parentheses in the rightmost column refer to the *WP 34S* with the optional quartz and capacitors installed for an accurate real-time clock. Both two *WP 34S* and a *WP 31S* are pictured overleaf. For creating one of them for yourself, ...

1. look for an *HP-20b* or *HP-30b* (the latter is recommended since featuring a far better ('rotate and click'¹⁵¹) keyboard – note both *HP* calculators are out of production for years),
2. get a proper programming cable (these cables are very rare meanwhile),
3. a file for flashing,

¹⁵¹ → [App. H.](#)

4. software for transmitting said file to the [HP-20b/-30b](#), and
5. stickers for bezel and keys (e.g., from [Jeff](#), → 188).

See the respective manuals for details (note these are under copyright – please respect it). Alternatively, you may get a [WP 34S](#) or [WP 31S](#) still custom-made (search the internet) or find them (used) in one of the usual bays.



Reference Literature

As mentioned above, some advanced functionality of your WP43 is taken over from previous HP calculators. The following vintage HP manuals are 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:

Topic	Recommended literature
Statistical distributions and their application	<i>HP-21S OM</i> , especially pp. 63 – 105
Manipulating bits and <i>short integers</i>	<i>HP-16C OH</i>
Programming	<i>HP-42S OM</i> <i>HP-42S PET</i>
Root finding and numeric integration	<i>HP-34C OH and Programming Guide</i> <i>HP-15C OH</i> ¹⁵² <i>HP-15C Advanced Functions Handbook</i> ¹⁵³ <i>HP-42S PET</i>
Accuracy of numeric calculations	<i>HP-15C Advanced Functions Handbook</i> , pp. 172 – 211 ¹⁵³
Algebraic equation solver	<i>HP-27S OM</i>
Financial calculations	<i>HP-17BII+ User's Guide</i> ¹⁵⁴

All these manuals can be found in the internet – three of them are even provided by *HP* still. A great collection (in every aspect) of literature about *HP* calculators (and related devices) is stored by *Eric Rechlin* at <https://literature.hpcalc.org/> – here you find best quality scans of all the manuals mentioned above (and more), most of them available in various languages. We recommend this site although its handling of copyright shows potential for improvement.

Depending on your education and professional qualification, textbooks about various mathematical, scientific, engineering, or business topics may be helpful in addition. Ensure you know enough about what you compute and how to do it properly (and please watch f. 185 on p. 271, as well as p. 19 of the *OM*).

¹⁵² Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c03030589.pdf>

¹⁵³ Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c03308725.pdf>

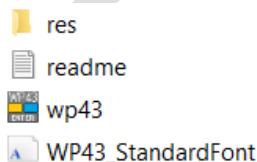
¹⁵⁴ Download a reprint from <http://h10032.www1.hp.com/ctg/Manual/c00363348.pdf>

APPENDIX E: EMULATING A WP43 ON YOUR PC

There is a *WP43 Windows Simulator* for your *PC*, featuring the same function set as the calculator.¹⁵⁵ Find it under <https://gitlab.com/rpncalculators/wp43/-/releases> for free download.



You will be asked to open a zipped folder. Extract it to any suitable location on your *PC*. Open its root directory and you will find this:



Install this font if you didn't do before yet. Start *WP43* by double-clicking on *WP43*. The *Simulator* window will open, looking like the picture here though larger.¹⁵⁶

Operate the *Simulator* with your mouse. Alternatively, the keys **0** ... **9**, **.**, **ENTER**, **+**, **-**, **x**, and **/** may be entered directly using the numeric keypad of your *PC*, **▲** and **▼** via the cursor up and down keys, and **↵** via [**←** Backspace]. Please turn overleaf for further *PC* keyboard shortcuts to *Simulator* keys.

¹⁵⁵ There are also *Simulators* for *MacOS* and *Linux* available. Please ask on the forum.

¹⁵⁶ A vertical screen size of 980 px is required, else the landscape window will open (as shown overleaf), needing only 568 px.

With development ongoing, there may be errors in startup. If you want to ensure the previous *state* will be loaded after a *SW* update to continue your work, do as follows:

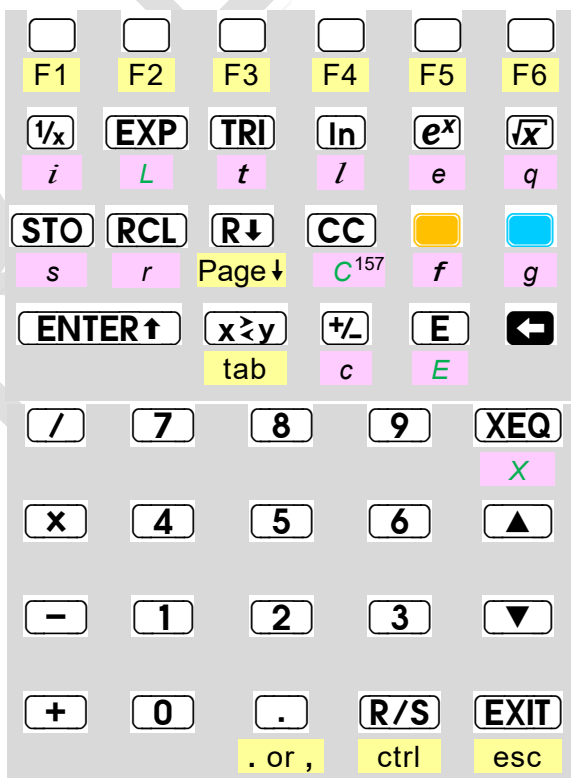
1. Press **SAVE** in the old version to back up to `wp43.sav`
2. Update the *WP43 Simulator*.
3. If `backup.bin` will not load in the new version automatically, delete it (`rm backup.bin`). Start `wp43`, then use **I/O LoadAll** to load `wp43.sav`.



Right clicking with the mouse will call -shifted labels directly –  **EXIT** will close the Simulator.

Note the shortcuts printed on **violet** will vanish in *AIM*, so you can use your *PC* keyboard for typing as usual. This may be beneficial in *PEM*, especially.

Taking advantage of this opportunity, your *PC* keyboard settings will apply, e.g., for upper and lower case letters. And “page ” and “page ” will become shortcuts to  and , respectively (e.g., for Greek letters).



¹⁵⁷ Capitals are printed in **green** here. And “ctrl” means the right “ctrl” key.

Furthermore, pressing ...

... **F9** copies the precise content of **X** to the clipboard. This is the way to output also very long integers $> 10^{298}$ in their full glory from the *Simulator* (compare **DUMP** on p. 37).

... **F10 (Print)** copies the entire 240×400 pixel *Simulator* screen to the clipboard, e.g., for recording or documentation (compare **SNAP** on p. 86).

Beyond *A/M*, pressing ...

... **Z** copies the full contents of all twelve lettered *registers* (**K, J, I, L, D, C, B, A, T, Z, Y, and X**) to the clipboard.

... **Z** copies the full contents of all 112 global *registers* thereto (**K, J, ..., Y, X, R99, ..., R00**).

... **2** (i.e. [alt gr] + [2]) inserts x^2 .

... **'** will open *A/M*.

ATTENTION:

While you work with the *Simulator* under *MS Windows*, **SAVE** will write the backup file to the operating system disk (usually C:) into the directory `msys64\home\user\wp43\SAVFILES` (**user** stands for your user name).

Calling **I/O SAVEST**, the respective *state file* will go into `...wp43\STATE`.

And **WRITEP** will write program files into `...wp43\PROGRAMS`.

At the bottom line, whatever your calculator writes to the *FD*, the *Simulator* will write to `...wp43\` into the corresponding subdirectory.

How to Exchange Programs Between Your *WP43* and Your *PC*

I/O WRITEP writes a program from program memory (in *RAM*) onto the *FD* in a named file. **I/O READP** reads a named file from the *FD* into program memory. Note these files are not designed to be user-readable but can be transferred easily.

To transfer a program from your *WP43* to your *PC*,

1. use **I/O WRITEP** on your *WP43* to write the program onto the *FD* in the

directory `PROGRAMS`; **WRITEP** will ask you for the *source program name* (i.e. its first global label) and a *destination file name*.

ATTENTION 1: You may choose the *file name* entirely independently of the *program name*. Keep suitable records telling you which file contains which program.

ATTENTION 2: **WRITEP** will overwrite a file carrying the same name in this directory!

With writing completed, **WRITEP** will display **Saved**.

2. Connect your *WP43* to your *PC* using a proper *USB* data cable.
3. Press **USB** to 'activate' the connection, so the *FD* will be seen by your *PC*.
(If it is not seen by your *PC*, however, then either
 - your cable is no data cable or
 - it does not connect properly or
 - it is defective. Exchange it and return to step 2.)
4. Start the *WP43 Simulator* and
5. call **I/O READP** on the *Simulator* to read the program into its memory. **READP** will display a list of `.p43` files in the default source directory (→ previous page) to choose from – you may have to redirect it to `DM42:\PROGRAMS`. With reading completed, **READP** will display **Program file loaded**.

The program read in will become the last program in memory; note it may carry the same name as another program being present there already.

Now you can do with it whatever you like, e.g., debug it by single-stepping through it as explained in [OMS 4](#). You can take advantage of your *PC* keyboard whenever you want to change something and *AIM* is active.

If you expect your *Simulator* session taking more time, you can 'eject' the *FD* from your *PC*, then press **EXIT** **ENTER↑** on your *WP43*, optionally disconnect the cable, and use your *WP43* independently in the meantime.

Eventually, when the program does what it was meant to do and you

have verified everything, you can ...

return your program from your PC to your WP43 like this:

- a. If you had disconnected, connect your *WP43* to your *PC* again (→ step 2).
- b. If your *WP43* is off, turn it on and call **USB** (as in step 3).
- c. Verify that the connection between your *PC* and your *WP43* is active (i.e. the *FD* is seen by your *PC*; compare step 3).
- d. Call **I/O** **WRITEP** on the *Simulator* to write your refined program (you will be asked for its name) into DM42 : \ PROGRAMS. **WRITEP** will ask you for a *destination file name* (as in step 1).

With writing completed, **WRITEP** will display **Saved**.

- e. Eject the *FD* (logically) from your *PC*.
- f. Press **EXIT** **ENTER↑** on your *WP43* calculator to return from active *USB disk* mode (you may disconnect the cable now) and
- g. call **I/O** **READP** which will display a list of .p43 files in the directory *PROGRAMS* to choose from. Select your file to read the refined program into the *WP43* program memory. With reading completed, **READP** will display **Program file loaded**. The program read in will become the last program in memory; note it may carry the same name as another program being present there already.

Enjoy!

APPENDIX F: FLASHING & UPDATING YOUR WP43

There are two ways to get your hands on a *WP43* calculator:

1. You can buy a *WP43* off the shelf (→ 3) or
2. you can flash a *DM42*.

Way 2 allows you to repurpose a *DM42* you own already, so you may save costs. This is explained in next chapter. Note a *DM42n* will not do it here.

The chapter thereafter (on pp. 247f) shows how to update your existing *WP43*, be it bought or flashed, when a new firmware becomes available.

How to Create Your *WP43* by Flashing a *DM42*

First, save your data stored on your *DM42*, if applicable. You can do this in the following (reversible) procedure which you will need anyway:

1. Start your *PC*. Take your *DM42* and turn it on; then press

SETUP

then either

or

1 File >

5 System >

2 Enter System Menu

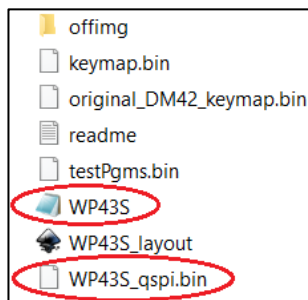
4 Reset to *DMCP* menu

2. Now connect your *PC* to the calculator *Micro-USB (B)* socket using a suitable data cable. Ensure it connects properly also on the calculator side – cutting back the plastic isolation a bit may be necessary. Then press...

- 6 Activate *USB disk*.** The *FD* of your *DM42* should show up as an external mass storage volume on your *PC*.

3. You can copy the entire volume with your data into any suitable directory on your *PC* now.

4. Start the internet browser on your *PC* and fetch the firmware you find under <https://gitlab.com/rpncalculators/wp43/-/releases>. You will be asked to open a zipped folder. Extract it to any suitable location on your *PC*. Open its root directory and you will find this →



5. Copy `WP43.pgm` and `WP43_qspi.bin` onto the *FD*.¹⁵⁸

Then logically disconnect the *FD* via **EXIT** **ENTER↑**. Leave the cable connected since flashing will draw some current, so the little coin cell in the calculator will appreciate your support.

6. Press **4** **Load QSPI from FAT**. When done, press any key (causing a reset), then **EXIT** and **3** **DMCP menu**.

7. Press **SETUP**
5 **2** **4**
3 **Load Program**. Scroll down to `WP43.pgm` and press **ENTER↑** **ENTER↑**. Wait ~ 15 s for flashing completed. Then press **EXIT** **EXIT** **1** and the bottom left key. Thereafter, your *WP43* is up and waiting for your commands. You may disconnect the cable now.



Finally, turn to p. 420, cut out the print there, stretch it over the keyboard, and operate the keys through it (this way is field proven with *WP 34S* since 2011). Alternatively, you can get the required overlay plus 17 key stickers properly printed, cut, and punched from [Jeff](#) (→ 188).

Should you ever want to leave *WP43*, enter **MODE** **DMCP** for returning to the *DMCP* system. If you had chosen option

1 in f. 158, the key assignments will stay as they were in *WP43*; else the EXIT/ON functionality will fall back to the bottom left key.¹⁵⁹

¹⁵⁸ We recommend deleting any other files `*_qspi.bin` and `*.pgm` from the *FD* since the calculator might attempt loading these instead of the two correct files.

Option 1: Copy `keymap.bin` onto the *FD* unless you have done before. This will reassign the keys to match the *WP43* layout also when leaving *WP43*.

Option 2: Copy `testPgms.bin` onto the *FD*; it will give you some test programs pre-entered.

How to Update Your WP43

If you have *Free42* still running on your *DM42* then proceed as explained in previous chapter. But if *WP43* is installed on your calculator already then you can update as follows:

1. Start your *PC*. Turn your *WP43* on.

SAVE your programs and data! Then leave *WP43* by pressing

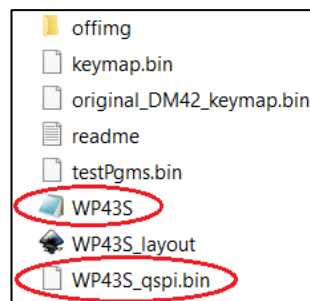
MODE **DMCP** to return to the *DMCP* menu. If you had copied `keymap.bin` of *WP43* before last flashing, key assignments will stay as they were on your *WP43* when navigating in the *DMCP* system – else the *DM42* assignments will become valid (→ 180).

2. Connect your *PC* to the calculator *Micro-USB (B)* socket using a suitable data cable. Ensure it connects properly also on the calculator side – cutting back the plastic isolation a bit may be necessary. Then press...

6 **Activate USB disk**. The *WP43 FD* should show up as an external mass storage volume on your *PC* now (if it does not then either your cable is no data cable or it does not connect or is otherwise defective; try with another cable).

3. Start the internet browser on your *PC*, go to <https://gitlab.com/rpncalculators/wp43/-/releases> and fetch the latest 'Firmware for DMCP' you find there. A zipped folder will be downloaded. Extract it to a suitable location on your *PC*.

You will find this in its root directory →



4. Copy `WP43_qspi.bin` and `WP43.pgm` (optionally `testPgms.bin`, too) onto the *FD*.

Then logically disconnect the *FD* from your *PC* via

¹⁵⁹ To retrieve the original *DM42* keyboard layout (as it is pictured on p. 181), → https://technical.swissmicros.com/dmcp/devel/dmcp_devel_manual/#_recovery_to_requ https://technical.swissmicros.com/dm42/doc/dm42_user_manual.html#quick_update_g [uide](https://technical.swissmicros.com/dm42/doc/dm42_user_manual.html#quick_update_g) – just activate the *FD* by pressing **USB** first and continue with step 2 there then.

EXIT **ENTER↑**.

Leave the cable connected, however, since flashing will draw some current, so the little coin cell will appreciate your support.

5. Press

4 **Load QSPI from FAT.** When done, press any key (causing a calculator reset), then **EXIT**.

6. Press any key to return to the *DMCP* menu. Then press

3 **Load Program.** Scroll down to **WP43.pgm** and press

ENTER↑ **ENTER↑**. Wait some 15 s for flashing completed. When done, press any key (causing another calculator reset), then **EXIT**, and your updated *WP43* will be booted reset. Feel free to disconnect the cable now.

7. Now you can recall all your saved programs and data:¹⁶⁰

I/O **LoadAll**

or recall just a subset (check the various load commands in the *IOI*).

Then resume your work on your updated *WP43*.

Sometimes (seldom), an update of the *DMCP* on your *WP43* may be required.¹⁶¹

¹⁶⁰ If you opted for a new set of `testPgms.bin`, these programs will be loaded when your *WP43* is booted (reset). Note, however, that **LoadAll** will recall all your previously saved programs (incl. the old set of test programs), overwriting the new ones).

Thus, if you want to get *your own* programs back after a firmware update and need a new set of test programs for any reason whatsoever, you shall first write *your own* programs onto the *FD* individually using **WRITEP** – in addition to **SAVE** in step 1. Then update your *WP43*.

Eventually, when you come to step 7 above, do not call **LoadAll** but **LoadV**, **LoadR**, **LoadΣ**, **LoadEq**, and **LoadSS** instead; then recall *your own* saved programs by multiple calls of **READP**. This way you will get back everything you created and have the generic test programs updated.

¹⁶¹ → https://technical.swissmicros.com/dm42/doc/dm42_user_manual/#DMCP_update_guide for how to do this.

APPENDIX G: TROUBLESHOOTING GUIDE

First a Few Words About Coin Cells

Your *WP43* is powered by a single *CR2032* coin cell like other *SM* calculators, allowing for a low hardware profile but carrying some disadvantages elsewhere. In particular, there are lots of cheap coin cells on the market well capable to deliver an open-circuit voltage $> 3\text{ V}$, but quickly render useless under load. Powered by such a cell, your calculator may show various less than funny properties (it may even look completely dead!) which can be retraced to voltage drops under load. Especially when running programs or advanced functions, your *WP43* draws some *milliamps* from its battery – a *WP43* looking innocent before may, e.g., reset suddenly when its battery is close to depletion.

Also low temperatures may let the voltage of such cells break down.

Thus, if you observe anything weird on your *WP43* and it is cold, turn it off, put it in a room of about 20°C or more and let it warm up to thermal equilibrium slowly. Then press  **SAVE** to save your programs and data if this is possible still.

Take a big breath. Then call  **INFO**  **?BATT** : if it returns $< 2.8\text{ V}$, you should open your *WP43* anyway – if it returns more but you suspect the battery being the root cause of your problems nevertheless (it may well be), run the following little program giving you maximum information about the battery state:¹⁶²

```
LBL 'BVTest'
REM 'battery voltage test under load'
CLΣ          reset STATS
LocR 03
15           for 15 sequential measurements
STO .00
STO .01
GTO B       take the very 1st battery measurement unloaded
LBL A       a loop to be executed 14 times
PAUSE 09    let the battery voltage recover from last iteration for 0.9s
10          10 repetitions = ~ 0.1s + PAUSE = ~ 1s per data point
STO .02
1.23456789  arbitrary value to stress your WP43
LBL D       draw voltage down by ...
sin         ... performing pointless work ...
arcsin
```

¹⁶² Kudos to [Warren Young](#) for the model of this routine.

```

DSZ .02
GTO D
LBL B
REM 'record 1 voltage measurement'
?BATT =  $U_n$ 
RCL .01
RCL- .00 =  $n^{\text{th}}$  measurement
Σ+ save data point ( $n$ ,  $U_n$ )
DSZ .00
GTO A
ASSESS
RTN
END

```

... 10 times ...

... then continue

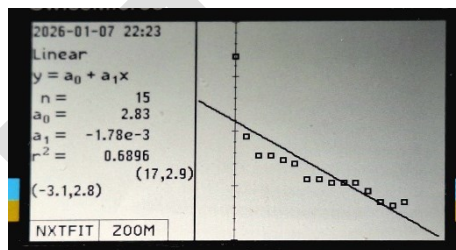
= n^{th} measurement

save data point (n , U_n)

run again for more measurements (15 in total)

thereafter, report results graphically

If all 15 measurements return comparable voltages decreasing linearly by small amounts then your battery is still fine. If the measured voltage drops sharply from 1st to 2nd and 3rd data point and stays low then your coin cell is nearly depleted and should be exchanged soon. This screenshot shows a good but no more fresh cell (U_n between 2.85 and 2.81 V).



To inspect the cell, turn your WP43 off, open it, and remove the battery.¹⁶³ Look at both sides of it: if the producer put a small sticker on one pole (usually on +) in firm belief that would keep toddlers from swallowing it (it does not!), remove it.

Unless you measured its voltage using above program already, you should do this now.¹⁶⁴ Any reading significantly below 2.8 V shall lead to battery exchange.

Insert a fresh quality cell.¹⁶⁵ Then verify if your observations which caused this investigation persist. **Be sure you did this before reporting any troubles.**

¹⁶³ → https://technical.swissmicros.com/dm32/doc/dm32_user_manual.html#_battery.

¹⁶⁴ **ATTENTION:** Most digital voltmeters (with an input resistance of some MΩ) and also a single call of ?BATT will tell you the open-circuit voltage of your battery only: even an almost depleted cell may still show ~ 3 V there. You are far better off as long as you find a moving-coil voltmeter somewhere: due to its lower input resistance of some kΩ /V it will give you better information about the state of your cell (but less than the program).

¹⁶⁵ Renowned brands control their cell production in a way you can trust them – but there are many fakes of such cells, too. Personally, I gathered good experiences with *Amann*, *Duracell*, *Energizer*, and *Varta*. *Panasonic* and *Toshiba* are reported being good, too.

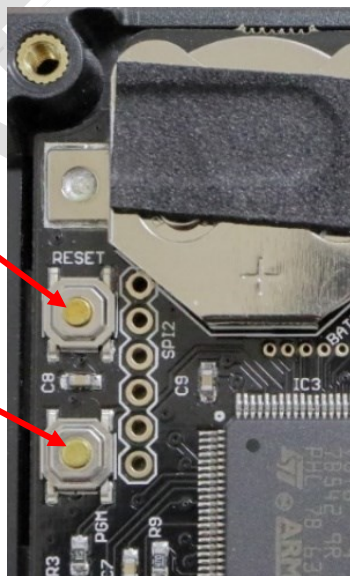
Furthermore, note that such a coin cell has significantly less capacity than, e.g., an AAA battery for obvious reasons. Looking at the differences in power consumption stated on p. 179, you will realize that using your *WP43* mainly for advanced math will deplete the cell 40 times faster than with occasional number crunching – it will last for many months with the latter.

Calculator Freeze

There are several ways to put your calculator in a freeze state wherein the *LCD* may show anything (or nothing at all) and your calculator will not react on any key you press. Usually, actuating the RESET button through the hole on its rear side should bring it back to life.

If this did not work, however, power your *DM42* via *USB*: this may suffice to reset it; if it does and the calculator was at room temperature, then the battery is the culprit.¹⁶⁶ After exchanging it you are usually done (check with the open calculator) – else do:

1. You will find the top left of the *PCB* looking like this (compare p. 183 for an earlier *PCB*). In any case, you will notice two small buttons labeled PGM (or BOOT0) and RESET.
2. Now:
 - a. Press and hold PGM.
 - b. Press RESET for some 2 s.
 - c. Release PGM. This shall reset your *DM42* and put it in bootloader mode.If it does then you can flash your calculator now (→ 245f), if necessary.



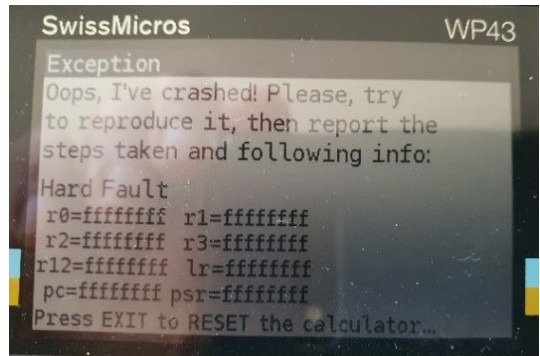
Generally speaking, the internal resistance of *CR2032* lithium cells is significantly higher (~ 10 Ω even for good cells) than in other common battery chemistry systems (~ 0.3 Ω for alkaline AAA batteries). If *SM* had chosen their top calculators to run on two AAA batteries instead, they had become a bit thicker at their top but had saved us a lot of troubles.

¹⁶⁶ Compare previous chapter. Almost depleted cells may be unable to process a power on, a reset, or even a single keystroke, leaving you wondering what has happened to your calculator (which tells you nothing).

- d. If it does not, however, and especially if the *LCD* stays blank also under *USB* supply, this may point to a real hardware problem. Read also the other cases in this *App*. If nothing applies, ask on the *SM* forum. If this does not help either, contact *SM*.

Exception

When you see a screen like this, you have experienced a crash, presumably due to a firmware bug. Please act as requested. If the crash is reproducible, record its boundary conditions and report this incident to us, please. We will investigate and return to you in this matter as soon as possible, either with advice or with a new firmware release.



Fresh Quality Battery Constantly Low

This was observed with a factory-fresh calculator in 2020: The low-battery indicator  was lit and the command ?BATT persistently returned 1.21 V, although the battery was actually good and measured over 3 V (open-circuit). If resetting the calculator, removing the battery, updating the firmware, etc. does not change the output of ?BATT, then inspect the *PCB*. There may be an improper soldering next to FB1 (a.k.a. a solder bridge – compare the picture on previous page).

If you have proper tools and experience, feel free to fix this yourself – else contact *SM*.

Individual Keys not Reacting Properly

This may be caused by switching domes faulty or misplaced. Lesser faults may cause particular keys requiring more force to actuate or just “feeling different”.

To access the keyboard side of the *PCB* carrying the switching domes, open your calculator by unfastening the two bolts at the top of its backside like you do for changing the battery. Thereafter carefully release the *LCD* connection [Ⓐ]¹⁶⁷ – note **this operation is at your own risk**. Then unfasten the two wide-headed Phillips bolts [Ⓑ].

With a light source located far away from this surface of your *PCB* and the domes placed properly, you should see equal reflections from all of them. The picture here shows an early *PCB* of 2017. Any anomaly you detect on this side of your *PCB* could cause a malfunction of the corresponding key.



If you are handy then you can try, at least, a cure available for little money.¹⁶⁸ Replacing the domes is easy – you just need a minimal quantity of isopropyl alcohol and a slow hand.

¹⁶⁷ https://technical.swissmicros.com/dm32/doc/dm32_user_manual.html#_disassembly contains some drawings in this matter.

¹⁶⁸ → <https://www.swissmicros.com/product/dm42-replacement-foil-with-domes>

Keymap Trouble

If you had loaded a `keymap.bin` not matching the layout you want or you do not find the keys properly assigned (and you are not in user mode!) then proceed this way (presuming both calculator and *PC* are running and connected properly, → 245):

1. Open the *menu* **MODE** on your calculator.
2. Therein, call  (**DMCP**) to return to *SM*'s **DMCP** menu.
3. Press  **Activate USB disk**.
4. Start the internet browser on your *PC* and look up https://technical.swissmicros.com/dmcp/devel/dmcp_devel_manual/#_recovery_to_required_keymap . Act accordingly.
5. Press **(EXIT)**, then the bottom left key – else you will run into the same trouble again.
6. Go to p. 245 for proper flashing.

Other Trouble

There is a plenty of opportunities more to kill or brick your calculator: drop it on rocky ground, sink it in a puddle, leave it in the laundry for too long, spill coffee over it, break its display, leave a leaking battery in it, etc. – just your imagination is the limit (and you may even go beyond). All these mishaps,¹⁶⁹ however, don't affect the firmware in any way but the basic hardware. Thus, please check https://technical.swissmicros.com/dm42/doc/dm42_user_manual.html#DMCP_update_guide and the chapters following there for advice. If nothing helped, turn to *SM*.

¹⁶⁹ *TN*: I came across this cute little term in Indian newspapers. It used to appear there as *kitchen mishap*, frequently linked to poor senior married women killed by exploding gas cylinders, opening the chance for their husbands to wed once more and get dowry again. In my opinion, *mishap* sounded far nicer than *accident* or worse.

APPENDIX H: HISTORIC REMARKS AND CAVEATS

FOR YOUNGER USERS

(WHAT GRANDPA STILL KNEW)

In company-internal communication, *HP*'s pocket calculator lines carried a family name each:

1. **Classics** for the *HP-35* of 1972 and its immediate successors *-80*, *-45*, *-65*, *-70*, and *-55*
2. **Woodstocks** for the smaller devices following, i.e. the *HP-21*, *-25*, *-22*, *-27*, *-25C*, (*-67*), and *-29C* (its printing sibling *HP-19C* belongs to the **Stings**)
3. **Spices** for the *HP-31E*, *-33E/C*, *-38E/C*, *-32E*, *-34C*, and *-37E* of 1978ff
4. **Nuts** for the *HP-41C* (*HP*'s first with an *LCD*) and its siblings *CV* and *CX*
5. **Voyagers** for the compact *LCD* landscape models *HP-11C*, *-12C*, *-15C*, *-16C*, and *-10C* of 1981ff (showing legendary minimum power consumption)
6. **Pioneers** for the line of portrait calculators following (including also algebraic models for the first time), i.e. the *HP-17B*, *-27S*, *-22S*, *-32S*, *-42S*, *-14B*, *-20S*, *-21S*, to the *HP-32SII* whose production ceased in 2001.

This was the last real product family of *HP* pocket calculators; single calculator designs followed. The *Classics*, *Woodstocks* and *Voyagers* had a particular fame for their mechanical ruggedness – various stories are told. You can still find many flawlessly working until today, for more than 50 years. That's quality!

The *Classics*, *Woodstocks*, and *Spices* featured power-hungry *LEDs*, so they enclosed rechargeable battery packs (usually *NiCd* – lithium-ion batteries were developed many years later). Though the *Woodstocks* and *Spices* also contained a serious electronic design flaw in their charging circuitry, actually killing thousands of them, each in a fraction of a *second*:

WARNING: Never (!!!) connect any *Woodstock* or *Spice* to its mains adapter unless you are sure it contains a good battery making good contact! And it shall be **turned OFF when you connect it. Since you can actually never be 100% sure about an old rechargeable battery, just never connect any of these calculators to the mains – **charge their batteries externally**.**

Also do not operate the card readers of *HP-65* or *-67* without a good battery properly installed.

Up to and including the original *Voyagers*, HP's calculators were built to be repaired. You can disassemble them easily, more or less. Getting suitable electronic spare parts after decades may become a challenge, however. Seek advice on the forum of the *MoHPC*.

On the other hand, disassembling a recent *Voyager* (e.g., a [HP-12C Platinum](#) or [HP-15C CE](#)) is a challenge – you may open its case (although they use tiny bolts with quite soft heads) but the *PCB* is fixed on the keyboard assembly by heat stakes which can be opened only once.

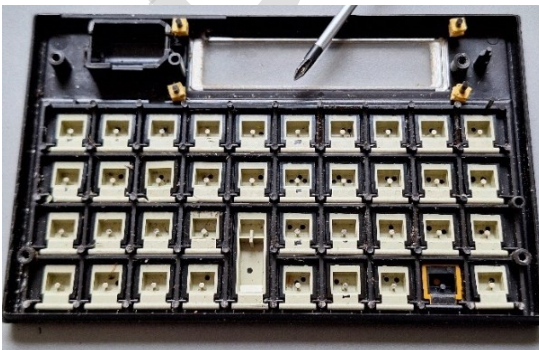


Keys and Keyboards

HP had a reputation for producing outstanding, solid pocket calculators also hardware-wise. This was earned with the *Classics* (competing with *Texas Instruments' SR5x*) and kept until today, although most factors of

distinction were sacrificed to cost reduction as time went on. Remember the [HP-35](#) was built by engineers for engineers and a major investment when it was launched – 395 US\$ of 1972 correspond to 3062 US\$ of today (the picture shows it in its original full plastic sales case – it was presented like a scientific instrument). It was even more expensive beyond the *USA* (selling for more than 1000 DM in Germany in 1974 equivalent to 1826 € of today) – HPs were simply unobtainable for most university students until prices dropped in the late 70's.

What made these calculators *special*? They featured ...



1. double-shot keys – their legends withstood all wear and tear,
2. individual keys – each key sat in its own little niche and moved entirely independent of its neighbors (→ this picture of an open [HP-10C](#)),

3. a precise click indicating reliably that the key had registered, and
4. they were built to last (well, most of them – compare previous chapter).

Double-shot keys were abandoned while the *Pioneers* were made in the 80's.

The last calculators featuring individual keys were the *Voyagers* – and since the *HP-12C* is produced continuously from 1981 until today, you can still buy such a calculator new (though accessing its keys became a challenge, cf. above). Actually, however, today's way of making the keys all together in one shot (→ *App. A*) is simply good enough – individual keys don't pay anymore.

And feature 3 was actually achieved by each key acting on a metallic snap contact, gold-plated and designed by HP itself (as for the *HP-35* here →). Later, circular switching domes became commercially available. More recently, HP's fabled 'rotate-and-click' keyboards (actually showing very little rotation) resulted from a careful combination of hardware decisions and software parameter settings, a combination HP tuned very well. Though also this expertise faded as HP's once renowned calculator division shrunk – it is actually extinct now.



Reverse Polish Notation

RPN is traced to *prefix* and *postfix* mathematical systems, developed by *Jan Łukasiewicz*¹⁷⁰ and proposed in his book of 1951. *Prefix notation* means placing operators before the numbers they operate on instead of between them.

17 + 4, for example, reads **+ 17 4** with *prefix notation* (called *Polish Notation* in honor of *Jan*). In *postfix notation* (hence *Reverse Polish Notation*), it reads **17 4 +** instead. But be aware of the necessity to separate input numbers.

¹⁷⁰ Pronounced ~Wookus-jéwidch → <https://www.youtube.com/watch?v=qRrAj-GCTQM>

¹⁷¹ The blue text following is quoted from the *MoHPC* as found at 2023-11-11, but with some additions in brackets.

Note, however, that many mechanical calculators (a.k.a. *adding machines*) worked alike before *Lukasiewicz* already.¹⁷¹ Entering and storing [two] numbers in some kind of *registers* and then performing an operation on them is a natural implementation. – In the years that followed [*Jan's book*], computer scientists realized that *RPN* or *postfix notation* was very efficient for computer math. As a postfix expression is scanned from left to right, operands are simply placed into a last-in, first-out (*LIFO*) *stack* and operators may be immediately applied to the operands at the bottom of the *stack*. By contrast, expressions with parentheses and precedence (infix notation) require that operators be delayed until some later point. Thus, the compilers on most modern computers converted statements to *RPN* for execution. (In fact, some computer manufacturers designed their computers around *postfix notation*.)

At the time that the *HP-35* was introduced, other pocket calculators typically used a partial algebraic model. That meant they could evaluate trivial expressions like $4 + 5$ but couldn't handle anything that involved parentheses or algebraic precedence. The technology of the time didn't allow for full algebraic compilers in pocket calculators.

RPN allowed *HP* to produce a pocket calculator that could evaluate [almost] arbitrary expressions [(→ *OMS 1*)] using the available technology. For many, learning a new style of entry was a small price to pay to be able to evaluate arbitrary expressions on a calculator. Once the technology to produce algebraic compilers could fit into a pocket calculator, most *RPN* users had decided that *RPN* was more efficient and consistent for the user as well as for the calculator. Also, because subexpressions are evaluated as they are entered, entry errors are more obvious with *RPN*. On an algebraic calculator, omitting an opening parenthesis, may not lead to a calculation error until much later when an entire subexpression is evaluated.

Another advantage to *RPN* is consistency between machines. Early algebraic models had differing limits of the complexity of the expressions they could evaluate. For example, *TI* [= *Texas Instruments*] catalogs from the late 70's listed how many levels of parentheses and pending operations each model could handle. Even today if you begin to use an algebraic calculator, you need to determine just "how algebraic" it really is. For example, some switch to *RPN* for unary [a.k.a. *monadic*] operators (i.e. 5 SIN rather than $\text{SIN}(5)$ or even $\text{SIN } 5 =$), some are still missing parentheses and/or precedence and many can't directly enter an expression like:

$$\frac{4 + 5}{6 + 7}$$

even though they claim to allow expressions to be entered as they are written.

Why Does ENTER↑ Work the Way It Does? And Why Are there Automatic Stack Lifts and Drops?

HP’s very first electronic calculator, the desktop model [9100A](#) of 1968, featured only a 3LS – but all three levels were always displayed on the screen of a small cathode ray tube: the *keyboard (entry) register X*, the *accumulator Y*, and the *temporary register Z*.

Earlier mechanical calculators worked with a *key-board register* and an *accumulator* only. The 3rd *register* became necessary to enable the [HP-9100A](#) to deal with slightly more advanced mathematical expressions.



See Its center keyboard section enlarged here. To solve a little calculating problem like

$$\frac{123}{45 + 67} ,$$

the following steps had to be executed leading to consecutive displays as shown in separate columns below (beginning with a clear *stack*):



<i>z</i>	0	0	0	123	123	123	123	123
<i>y</i>	0	123	123	45	45	112	123	1.0982...
<i>x</i>	123	123	45	45	67	67	112	112
Input	123	$\uparrow$ 172	45	$\uparrow$	67	+	$\downarrow$ 173	$\div$

But a pocket calculator like the [HP-35](#) (pictured overleaf to scale) could display only one *register* with its *LEDs* for space and power reasons – and that *register* had to be **X** to give feedback during keyboard entry. In consequence, to show the calculation result (supplied in the *accumulator* **Y**), *y* must be moved into **X**. Thus, e.g., both **+** and **↓** have to be executed immediately after each other, preferably in a single new operation **+** (with an *automatic stack drop*, → [OMS 1](#)). The same applies to the other *dyadic* operations as well.

Hence, on an [HP-35](#), above little problem is solved like this:

z	0	0	0	123	123	0	0
y	0	123	123	45	45	123	0
x	123.	123.	45.	45.	67.	112.	1.0982...
Input	123	ENTER↑	45	ENTER↑	67	+	÷

Calculating inside out (as recommended in [OMS 1](#)) requires less *stack* levels for this task. But if there is an intermediate result in **X**, it has to be shifted out of the way into the *accumulator* **Y** before entering the next number into **X**. Thus, **↑** is executed automatically then (= *automatic stack lift* ([ASL](#)), → [OMS 1](#)).¹⁷⁴

y	0	45	45	0	112.	123.	0
x	45.	45.	67.	112.	123.	112.	1.0982...
Input	45	ENTER↑	67	+	123	x↔y	÷

¹⁷² This key worked like **ENTER↑** did (and does) on subsequent calculators.

¹⁷³ **↓** did exactly what **DROP↓** does on your *WP43*; this desktop calculator key disappeared on *HP*'s pocket calculators for space reasons (→ an [HP-35](#) of 1972 overleaf). **ROLL↓** survived as **R↓** – it does almost the same. **ROLL↑** equals **R↑**.

¹⁷⁴ The *3LS* of the [HP-9100A](#) was a severe limitation for calculations.

On the [HP-35](#), **T** was added to **X**, **Y**, and **Z** to provide one temporary *register* more, allowing for more complex calculations under the boundary conditions of the early 70's. Actually, that purpose is reflected in its name, maybe also that it is the top stack register. Likewise, we added **A**, **B**, **C**, and **D** on our *WP 34S* since dealing with *complex numbers* on it required two *registers* for each such number (compare the [HP-15C](#) of 1982 or even its *Collector's Edition* of 2023). Our resulting *8LS* turned out being also advantageous for simpler objects (field proven since 2011, → [OMS 1](#)).



Such an **ASL** is not required, of course, if **(↑)** has just been pressed before. This is why **ASL** is disabled after **(ENTER↑)**.

And **ASL** is also disabled after **(CLX)** since the new number entered shall simply replace the one just deleted in **X** – an extra zero on the *stack* makes no sense.

$\Sigma+$ and $\Sigma-$ are dedicated commands for adding (accumulating) and subtracting (removing) statistical, measured, or counted data ($\rightarrow$ [OMS 2](#)). They were designed just for data input since their first appearance on the [HP-80](#) and [-45](#) in 1973 and were not really meant to be used in *stack* calculations. This applies in analogy also to the 6 *Matrix Editor* commands mentioned in *Section 1*.

How Many Stack Levels Are Necessary?

Millions of *HP RPN* pocket calculator users were (and still are) happy working just with a *4LS* and ≥ 1 *GP register* since 1972. *Heathkit* offered an *RPN* pocket calculator for airplane navigation featuring a *5LS* a bit later. There was found, however, no calculator hardware with larger *stacks* until *HP* launched its model [28C](#) featuring an ‘infinite’ *stack* with *RPL* in 1987. Although *HP’s RPL* calculators significantly exceeded shirt pocket size immediately, the *RPL* line lived

until 2015 when production of the [HP-50g](#) ceased.

Per definition you can't have *top level repetition* (→ [OMS 1](#)) with an infinite *stack*, and also **R↓**, **R↑**, and **FILL** make no sense there for the same reason. All in all, *stack* handling in *RPL* requires many more commands (DUP, SWAP, DROP, OVER, ROT, UNROT, ROLL, ROLLD, PICK, UNPICK, PICK3, DEPTH, DUP2, DUPN, DROP2, DROPN, DUPDUP, NIP, and NDUPN) than just **ENTER↑**, **x↗y**, and **R↓** in classic *RPN*. While *RPL* certainly has its merits, many potential users (advanced students and professionals) prefer the easier approach of *RPN* and, last not least, true pocket calculators. Such calculators are made and sold until today.



The size of the traditional *stack* was certainly a compromise between technical resources and engineering cost in 1972. We are not bound to that anymore. Decades of field experience solving real-life problems did show that a *4LS* suffices in almost all cases if you calculate inside out as recommended. Though a *stack* overflow may happen silently without prior warning and the odds are high you will not notice it in time – but the consequences for your calculations may be catastrophic. With an *8LS*, however, you are definitely on the safe side (as demonstrated comprehensively in [OMS 1](#) and field proven by [WP 34S](#) and [WP 31S](#) since 2011).¹⁷⁵ **Calculating *sans soucis!***TM

← Obviously, *HP* had thought about a dedicated statistical calculator in the early 70's

(past [HP-80](#), probably also past [HP-45](#)), but that product never materialized. Only a prototype case was discovered and published by [aj04062](#) (USA) at the [MoHPC](#) in 2024.

The four bottom rows of its keys are all set with $\Sigma+$ bottom right; [RDM](#) probably returned a *pseudo random number*. The primary functions in the top four rows can be guessed based on the yellow prints:

1. $\sqrt{x}$, [LN](#), [LOG](#), x^2 , and [Q](#),
2. $((x^2)^{-1})$, $\dagger^{-1}$, $\bar{x}$, $\bar{s}$, and [yellow](#),
3. F^{-1} in the center of 3rd row,
4. eventually [ENTER†](#), [CHS](#), maybe [CLΣ](#), and [CLx](#). [CLEAR](#) might have meant “clear all” – but that could be reached easier by switching off.

On either side of F^{-1} , both [STO](#) and [RCL](#) as well as $x\bar{z}y$ and $R\downarrow$ were located most probably. [FMT](#) denotes a modified *t*-test.

Outputs of [LIN](#) and [QUAD](#) [REGRESSION](#) were interesting, since r^2 , a_0 , a_1 , and a_2 (if applicable) should have been returned – so [QUAD](#) had overwritten the entire 4LS then. And quadratic fitting required at least 9 (GP) registers for accumulating all the statistical sums necessary ($\rightarrow$ 294).

For better layout coherence, Q^{-1} and [Q](#) should have been swapped though.

Way before PCs became available, HP manufactured even printing programmable scientific ‘pocket’ calculators – you needed a lab smock with extra deep pockets for carrying them. See an [HP-19C](#) pictured here $\rightarrow$



¹⁷⁵ Find a historic discussion about this topic on [Valentin Albillo's](#) excellent HP website <https://albillo.hpcalc.org/threads/HP%20Thread%20VA100b%20-%20202005-07-15%20-%20How%20many%20stack%20levels.pdf>.

Each weighed some 370 g (incl. batteries) and took 8.7 cm × 16.6 cm on your desk where you could secure it with a steel rope to reduce its diffusion rate in the time you were not present.

How Many Stack Levels Shall Be Displayed?

Millions of *RPN* pocket calculators built so far displayed and still display *x* only – *y*, *z*, and *t* are quietly acting unseen. Users were doing all sorts of tricks using these four *registers* – just tracking *y*, *z*, and *t* in their minds. Silicon memory was scarce and expensive – human memory was cheaper for decades.

Even *HP's RPL* calculators, launched between 1987 and 2015, with their 'infinite' *stacks* displayed no more than their bottom four *stack registers*, although many of the tricks making sense on a finite *stack* became complex or obsolete there (→ previous chapter).

With the *8LS* and *RPN*, nobody expects you tracking *y*, *z*, *t*, *a*, *b*, *c*, and *d* in your mind – it is simply not necessary at all. You know how the *stack* contents move, and they will show up automatically when needed (→ [OMS 1](#)). Mathematical functions on your *WP43* operate on the contents of the lowest four *stack registers* (*x*, *y*, *z* and *t*) maximum at once. Thus, and assuming human mental abilities did not deteriorate in last decades, displaying more than four *stack* levels carries no lasting benefit. This holds in particular since the odds for a *stack overflow* in real-world calculations are reduced to zero when using the *8LS*, computing inside out as recommended (→ [OMS 1](#)). For the same reason, we omit heading display indicators *X*, *Y*, *Z*, and *T*. You chose a *WP43* for



yourself, so you are certainly able to remember these four letters naming the bottom *stack registers*.

It is even debatable if more than the bottom 1 or 2 *stack* levels need to be displayed at all. This decision is up to you – **DSTACK** is ready to help.

Being at it, if you want to experience *RPN* on a clean keyboard, a vintage **HP-21** is recommended (pictured on previous page approximately to scale). Exceeding the function set of the **HP-35** in a solid, compact, very handy case weighing some 180 g, it was still built to survive all wear and tear – thus, a working **HP-21** can be found nowadays still. *HP* missed, however, the opportunities to produce tidy successors with the **HP-31** or **-10C**, printing their shifted functions on the keyboard instead of the slanted key fronts. Cost reduction ...

Precision and Accuracy of Calculations Using Finite Real Numbers

Before the **HP-35**, precision of most technical calculations was limited to 3 - 4 significant digits for decades – that was what you could achieve on a quality slide rule operating carefully. Note this sufficed for bringing so-called ‘astronauts’ to the Moon and sending two *Voyagers* to the outer planets of our solar system and into interstellar space beyond, as we all know.

Then the **HP-35** changed personal computing completely – featuring a precision of 10 digits without user ado. No slide rule could compete anymore. In wise self-restraint, however, startup default display format of many *HP* pocket calculators for science and engineering became **FIX 4**.

Since then, paradigms changed again: Now, precision of your **WP43** and its siblings goes far beyond what is needed in most applications even in science. As mentioned above, internal computations are usually carried out with 39 digits.¹⁷⁶ Displayed results (rounded to 34 digits) should show a relative error $< \pm 1 \times 10^{-33}$ per operation. See some random **examples** in **SDC**:

- **n $\frac{1}{x}$ **2n** $\times$ **2** $-$** returns precisely **0**. for almost all primes **$n < 700$** (except **7** $\frac{1}{x}$ **14** $\times$ **2** $-$ **2** $\div$ returning 5×10^{-34} , like **641**; for **67**, **83**, and **683**, results are -5×10^{-34});

¹⁷⁶ There may be even more ($\rightarrow$ [Background Facts and Considerations](#) below). Feel free to use $\left(729^{33.5} / 3^{201}\right) - 1$ and $\left(531\,441^{1072.5} / 9_{.6435}\right) - 1$ for testing any calculator you may get your hands on. Your **WP43** returns precisely **0**. for both problems.

- $7 \text{ [sin] [arcsin] } 7 \text{ [=] } 7 \text{ [/]}$ returns 1.4×10^{-34} ;
- $7 \text{ [ln] } e^x \text{ } 7 \text{ [=] } 7 \text{ [/]} = 2.9 \times 10^{-34}$ while $7 \text{ } e^x \text{ [ln] } 7 \text{ [=]}$ returns plain 0.;
- $7 \text{ [x] [x^2] } 7 \text{ [=] } 7 \text{ [/]} = -2.9 \times 10^{-34}$ while $7 \text{ [x^2] [x] } 7 \text{ [=]}$ returns plain 0;
- $7 \text{ [ENTER] } (\sqrt[y]{x}) \text{ } 7 \text{ [y^x] } 7 \text{ [=] } 7 \text{ [/]}$ returns -4.3×10^{-34} .

Greater inaccuracies may occur next to poles. – For comparison, SLVQ shows an inaccuracy of $\pm 2 \times 10^{-33}$ ($\rightarrow$ [OMS 5](#)) while nested advanced operations like SOLVE(f) may be inaccurate up to some $\pm 6 \times 10^{-33}$ or even more due to many elementary operations executed therein, whose inaccuracies accumulate.

There is a quasi-standard (called *calculator forensics*) established to find out about various processors, firmware, precision, and ‘accuracy’ of different calculator models: compute $\arcsin\{\arccos[\arctan(\tan\{\cos[\sin(9^\circ)]\})]\}$ (although this formula is mathematical nonsense). An ideal brainless number cruncher of infinite internal precision will return exactly 9° or equivalent.¹⁷⁷

Real calculators (computing with a finite number of digits) deviate for obvious reasons. Your WP43 returns

$$8.999\,999\,999\,999\,999\,999\,999\,999\,999\,999\,999\,999\,937\,535 = 9 - 6.246\,5 \cdot 10^{-29}.$$
¹⁷⁸

Far simpler is another test, but very potent: Enter **10 000 001.** and press $[x^2]$ just seven times. Your WP43 will return

$$1.000\,012\,800\,081\,280\,341\,377\,066\,802\,645\,679 \times 10^{896}.$$

¹⁷⁷ $\rightarrow$ [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)))))))

¹⁷⁸ If you want to know how other calculators have performed in this test, look at <http://www.rskey.org/~mwsebastian/miscprj/results.htm>. But be aware that ...

- a) that list was closed in 2007, and
- b) some calculators may cheat or even lie to you based on their guesses what the expected result shall be!

TN: There is a parallel in airliners: until 2016, *Boeing's* airplanes executed the pilot's commands immediately while *Airbus'* execute what its software ‘thinks’ the pilot should have done under the conditions it observes. Alas, the latter may turn out dangerous and caused at least two fatal crashes. Meanwhile, *Boeing* has caught up ($\rightarrow$ *B737max*). Luckily, a calculator can't crash this way – but there remain risks:

Actually, the problem with a calculator lying to the user is not the lie itself but that you never know if it is lying or telling you the truth – the calculator becomes a suspicious device, and trust in its reliability is destroyed reliably.

TN: Parallels in recent administrations are obvious – several are competing 42 years after 1984, and just 77 years after *George Orwell* wrote his vatic novel.

This is the most precise *real* result known for any pocket calculator worldwide.¹⁷⁹ Nevertheless, only its first 32 digits are correct! For comparison, enter **10 000 001** (without decimal point) and press $[x^2]$ seven times again. Your WP43 will return

100 001 280 008 128 034 137 706 680 264 566 942 $\times 10^{861}$

instead, which actually is an 897-digits *long integer*, precisely

100 001 280 008 128 034 137 706 680 264 566 942 362 065 259
381 704 558 672 471 663 975 250 935 637 066 049 668 878 686 323 398
085 866 658 787 740 848 661 692 888 859 950 737 447 654 753 345 168
002 963 740 073 746 995 710 556 007 241 670 099 300 929 440 107 854
408 072 554 911 252 317 789 928 499 808 013 093 965 927 574 941 133
244 381 547 633 598 449 089 802 869 082 118 094 468 776 017 826 651
186 922 820 900 475 103 999 094 028 004 541 163 740 697 539 733 888
428 367 675 593 254 423 047 914 478 918 135 752 102 429 288 143 361
653 459 171 040 814 216 776 401 746 346 343 970 859 728 390 095 407
066 078 152 415 625 359 663 834 513 241 403 709 468 936 478 003 065
366 991 757 932 039 263 495 546 002 578 420 990 239 188 092 834 549
068 962 510 444 861 858 929 119 375 002 542 434 572 320 889 769 109
362 436 235 325 985 158 624 722 403 373 962 502 148 033 363 010 732
930 713 924 088 117 757 151 012 106 967 826 289 242 967 898 516 694
703 304 135 759 892 818 613 724 712 834 463 748 230 478 358 549 260
032 564 540 002 953 869 332 914 454 490 582 230 113 562 493 732 890
458 571 501 560 870 702 174 970 266 185 257 957 423 611 226 456 640
106 680 000 341 376 000 812 800 001 280 000 001

Taking this test further will increase inaccuracy – after 27 presses of $[x^2]$ only the first 25 digits of the returned *real* result will be correct.

Such outputs qualify the concept of ‘*exactness*’ or ‘*accuracy*’ as a whole – they may deviate from your expectations but are the best results nevertheless you can get from a calculator of today featuring finite precision, not lying to you.

At the bottom line, your WP43 will return 34 digits always but not all of them are guaranteed being true.

Furthermore, there are inaccuracies caused by the limited precision of your input – being usually far greater. Note the *least accurate constituent* (LAC) of your computation will determine the resulting accuracy – your result can never have more significant digits than your LAC ($\rightarrow$ 302).

¹⁷⁹ If we were marketeers we would praise the WP43 as *the most precise calculator on the worldwide market – second to none*. Since we aren’t we tell you that, e.g., the WP 34S with DBLON and Free42 are equally precise.

And do not just believe in a result displayed – do some quick order-of-magnitude estimates yourself, check and verify! A critical mind will help always.¹⁸⁰

About the Classical *Mach Number* Formula

Let's return to said vintage formula ($\rightarrow$ *OMS 1*):

$$\sqrt{5 \left(\left[\left(1 + 0.2 \left[\frac{CAS}{661.5} \right]^2 \right)^{3.5} - 1 \right] \{ 1 - 6.875 \times 10^{-6} \times PA \}^{-5.2656} + 1 \right]^{0.286} - 1 \right)}$$

Nowadays, evaluating expressions like this is no real problem anymore also on many *algebraic* calculators. Here is its solution using an *HP-12C Platinum* in *ALG mode*:¹⁸¹

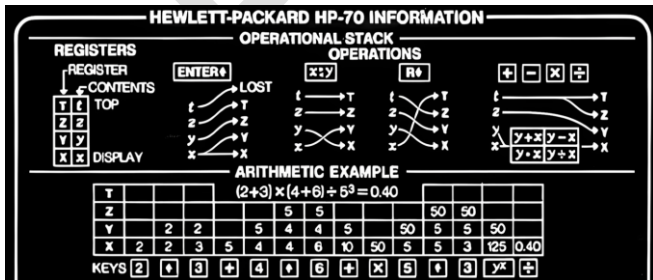
350 / 661.5 = x^2 x .2 = + 1 = y^x 182 3.5 = - 1 = STO 0

¹⁸⁰ Before electronic calculators, the order of magnitude of the result had to be pre-estimated and digits were few. Now, digits are far more than enough but erroneous results in a totally wrong ballpark (due to input errors or operating mistakes) may easily slip through unnoticed.

TN: Parallels: Before the internet, information was scarce and had to be carefully searched. Nowadays, information is abundant and has to be carefully checked and verified, else AI hallucinations, blatant lies, or propaganda – euphemistically called *fake news* by interested parties – can and will spread easily. Entire less than democratic administrations and regimes may be (and were) erected where people were successfully confused by avalanches of *fake news* (also less polite terms are common) so they couldn't tell fake from truth anymore. PRINCIPIIS OBSTA.

181 **ATTENTION:** **ENTER** works as **=** on this calculator in *ALG* mode. Therein, it allows for 13 open parentheses but knows nothing about operator hierarchy beyond – it calculates stubbornly from left to right. The same is observed with the *HP-19BII*.

TN: Seems *HP* thought and thinks this primitive (so-called 'chain calculation') paradigm suits their business / financial users best. The lasting success of the *HP-12C* in these quarters proves this evidently (compare *OMS 2*). Note *HP* were more optimistic in the 1970's.



¹⁸² This operation should rather read $\lceil \cdot \rceil$ in this mode.

1 [=] (6.875 [EEX] [CHS] 6 [x] 25500 [)] [=] (y^x 5.2656 [CHS] [=]
 [x] [RCL] 0 [=] + 1 [=] (y^x .286 [=] - 1 [=] [x] 5 [=] (x

Note this algebraic solution (80 keystrokes) works inside-out as recommended for *RPN* in *OMS 1*, hence using [=] twelve times, though only one pair of parentheses and one *GP register* for solving. And (x²) as well as (x) are postfix operations also in *ALG* mode here, and [CHS] operates on the open input.

For solving this formula from left to right (almost) as written, even 92 keystrokes are required:

5 [x] ((((1 + (.2 [x] (350 [÷] 661.5 [)] (x²) [)] (y^x 3.5 - 1 [)]
 [x] (1 - (6.875 [EEX] [CHS] 6 [x] 25500 [)] (y^x 5.2656 [CHS] + 1 [)]
 (y^x .286 - 1 [)] (x

Not a single [=] is used here.

Feel free to compare with the solution in *RPN* mode on the very same device:

350 [ENTER↑] 661.5 [÷] (x²) .2 [x] 1 + 3.5 (y^x 1 -
 6.875 [EEX] [CHS] 6 [ENTER↑] 25500 [x] [CHS] 1 + 5.2656 [CHS] (y^x [x]
 1 + .286 (y^x 1 - 5 [x] (x

These are 63 keystrokes compared to a minimum of 80 in *ALG*, i.e. 27% more on the same device!

You may, however, find more intelligent implementations of algebraic mode easily. The *CASIO fx-991DEX CLASSWIZ*, for instance, a calculator computing exclusively algebraically,¹⁸³ allows for entering the entire *Mach number* formula from left to right using 82 keystrokes – and you can even replay and edit it (but beware: when this calculator turns off automatically after some minutes of inactivity, your nice formula will be gone without a trace!).

For years, there are also ‘scientific’ calculators available featuring so-called ‘natural textbook display’ or ‘MathPrint’ or alike. Actually those are not targeted to science or engineering at all but to the educational (mass) market¹⁸⁴ where

¹⁸³ It features prefix functions (like $\sqrt{\blacksquare}$ or $x^\blacksquare$), implicit multiplications, and perfectly knows operator hierarchy.

¹⁸⁴ Mainly to schools, although some call themselves ‘high’. Actually, those calculators are targeted to the math teachers there (who stick to their curricula as long as they can) – the parents will buy what the teacher recommends, and the students have to use it.

problems typically culminate in inserting values into formulas provided ('*canned problems*' as [Garth Wilson](#) calls them, → [OMS 1](#)). None of those awesome calculators, however, can display the entire *Mach number* formula in a row so far.

Feel free to assess also their aptitude for *explorational mathematics*, coherence of their user interface, and consistency of their operation (→ [OMS 1](#)).

TN: Don't you ever confuse such a *high school* with a German *Hochschule* – identical names though totally different ballparks!

But since former *Fachhochschulen* (the minor siblings of universities and *Technischen Hochschulen*) were rebaptized in *universities of applied sciences*, the *Technischen Hochschulen* had to become *technical universities* to keep distance. Little substantial improvement at the bottom line – just an unlimited belief in pretty facades and an unbroken trend of euphemizing lower quality. Education marketing ...

APPENDIX I: ADVANCED MATHEMATICAL
FUNCTIONS AND TASKS

Lasciate ogni speranza, voi ch’è entrate!
(Dante Alighieri, Commedia Divina)

There’s neither witchcraft nor sorcery in mathematics.
(unknown contemporary author)

Your WP43 contains several operations covering advanced mathematics. Many are taken over from WP 34S and previous HP calculators, some are implemented here for the first time on an RPN calculator (→ App. D). Find these functions described below in more detail than in the IOI, together with a few traditional pocket calculator functions matching the topic.

As explained in OMS 1, we presume you are able to read and comprehend 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 and apply something you overview and know the background of – else it may even become dangerous for you and your fellow men (¡no toca que desconoce!).¹⁸⁵

Number Generating Functions

The following are all monadic functions except COMB and PERM.

Name	Remarks (→ 14ff for general information)
B_n, 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: $B_n^* = 2 \cdot \frac{(2n)!}{(2\pi)^{2n}} \cdot \zeta(2n)$ → 332 for $\zeta(x)$.

¹⁸⁵ Real world shows many sad events where people full of good will caused large damages by applying tools they did not know sufficiently – or standard tools in areas where those were not applicable. There was, e.g., a major oil spill in the Gulf of Mexico caused this way by American employees (hence wanting to call it Gulf of America on its oily side?). “Wenn Dumme fleißig werden, wird’s gefährlich” (≈ There’s nothing more dangerous than busy fools), a boss of mine used to say.

Name	Remarks (→ 14ff for general information)
COMB, PERM	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 <i>IOI</i> (→ 31 and 69, respectively). Note that $C_{y,0} = 1$, $C_{y,1} = y$, $C_{y,2} = 0.5 y (y - 1)$. $C_{y,x}$ applies to the <i>binomial distribution</i> (→ 273): In a <i>Galton box</i>¹⁸⁶ featuring y rows of pins and fed with 2^y balls, $C_{y,x}$ is the number of balls expected in column x of the box (start column counting with zero). $P_{y,x} = \frac{\Gamma(y+1)}{\Gamma(y-x+1)} \quad \text{and} \quad C_{y,x} = \frac{\Gamma(y+1)}{\Gamma(x+1) \Gamma(y-x+1)}$ are the general formulas working also for non-integer numbers and in <i>complex</i> domain.
FACTOR	... factorizes an integer input number. The output is a <i>real</i> matrix of two rows. Its 1st row contains the prime factors found and the 2nd row their exponents. E.g., 2 443 466 684 015 280 FACTOR returns $\begin{bmatrix} 2. & 3. & 5. & 7. & 11. & 17. & 123456791. \\ 4. & 3. & 1. & 2. & 1. & 1. & 1. \end{bmatrix},$ since it equals $2^4 \times 3^3 \times 5 \times 7^2 \times 11 \times 17 \times 123456791$. ATTENTION: If you suspect a number $> 10^9$ being prime, use PRIME? first – it is significantly faster than FACTOR in this matter. FACTOR operating on actual primes $> 10^9$ may take very long (→ 335) – feel free to use EXIT to abort).

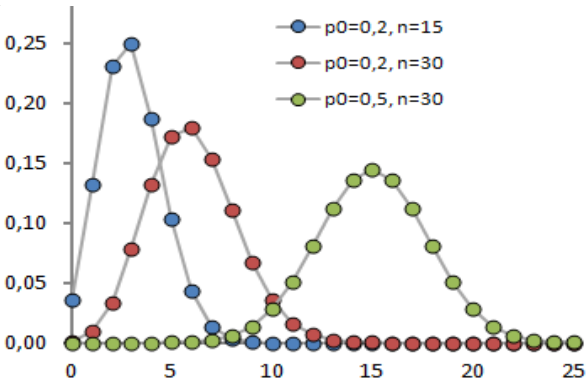
¹⁸⁶ *TN:* This is called «*Planche de Galton*» in French, „*Galtonbrett*” in German, and “*macchina di Galton*” in Italian, for example. Note the subtle differences in naming. *Galton* invented his box or board in 1889.

Name	Remarks (→ 14ff for general information)
FIB	For <i>integers</i>, FIB returns the <i>Fibonacci number</i> f_n with the counting number $n = x$. The <i>Fibonacci numbers</i> 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 WP43. For non-integers, FIB returns the <i>extended Fibonacci number</i> $F_x = \frac{1}{\sqrt{5}} [\Phi^x - \Phi^{-x} \cos(x \pi)]$ for any <i>real</i> or <i>complex number</i> x, with $\Phi = \frac{1+\sqrt{5}}{2}$ denoting the <i>golden ratio</i> (→ 123).

Statistical Distribution Functions (**PMF**, **PDF**, **CDF**, etc.)

Stack-wise, the following functions stored in **PROB** are all *monadic*. Actually, they feature more parameters though, supplied in *registers I, J, and K* as applicable and mentioned below. In the following, the five discrete distributions are covered first, the nine continuous ones thereafter. Plots (taken from *Wikipedia*) show typical *PMF*'s or *PDF*'s.

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



BINOM_p returns

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

(→ **COMB** on p. 272 for explanation of the notation).

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

The *binomial distribution* is fundamental for error statistics in industrial sampling, e.g., for designing sample test plans.¹⁸⁷

Example:

What is the probability for finding no fault in a sample of 15 objects drawn from a batch of 300 wherein you expect 3% defective objects overall? This will tell you:

.03 **STO** **I** **15** **STO** **J** **0** **PROB** **Binom: Binom**_▲

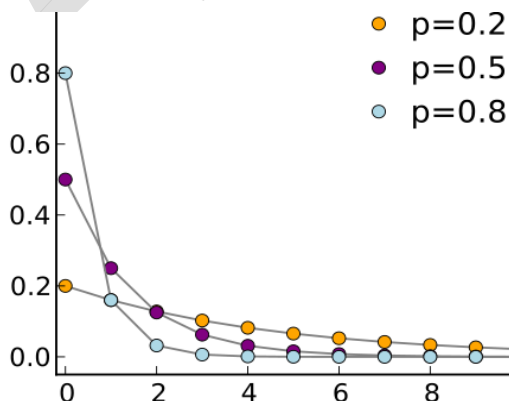
... returning 0.633 – so the odds are almost two out of three that you will not detect any defect in your sample!¹⁸⁸

Geom: Geometric distribution:

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

GEOM_▲ returns $F_{Ge}(m) = 1 - (1 - p_0)^{m+1}$,

being the probability for a first success after **m = x** *Bernoulli* experiments. The probability **p₀** for a success in each such experiment must be specified in **I**.¹⁸⁹

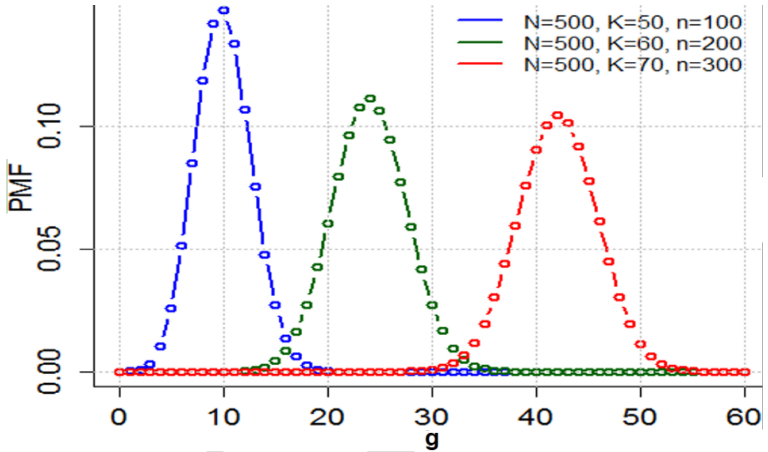


¹⁸⁷ → <http://www.itl.nist.gov/div898/handbook/eda/section3/eda366i.htm>.

¹⁸⁸ The exact result for said boundary conditions is 0.626, calculated using the *hypergeometric distribution* overleaf. For decades, such probabilities were computed applying the *binomial* or even the *Poisson distribution* since calculations are easier there. Don't be afraid – you own a WP43!

¹⁸⁹ → 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 shown here are $K/N = p_0$ and $N = n_0$).



$$\text{HYPER}_p \text{ returns } p_H(g; n; p_0; n_0) = \frac{\binom{n_0 p_0}{g} \binom{n_0(1-p_0)}{n-g}}{\binom{n_0}{n}}$$

(→ **COMB** on p. 272 for 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. It may be neglected, however, in alleged '*large*' batches ($n_0 > 10$) and for '*small*' sample sizes ($< 10\%$ of n_0). Check the differences.¹⁹⁰

NBin: *Negative binomial* distribution with the total number of failures f (in n draws) in X , the gross probability of a success in a single draw p_0 in I , and n in J . **NBIN**_p returns

$$p_{NB}(f; n; p_0) = \binom{n-1}{f-1} p_0^f (1-p_0)^{n-f} = C_{n-1; f-1} p_0^f (1-p_0)^{n-f}$$

(→ **COMB** on p. 272 and compare **BINOM**).¹⁹¹

¹⁹⁰ Start reading here for more: http://en.wikipedia.org/wiki/Hypergeometric_distribution.

¹⁹¹ → http://en.wikipedia.org/wiki/Negative_binomial_distribution.

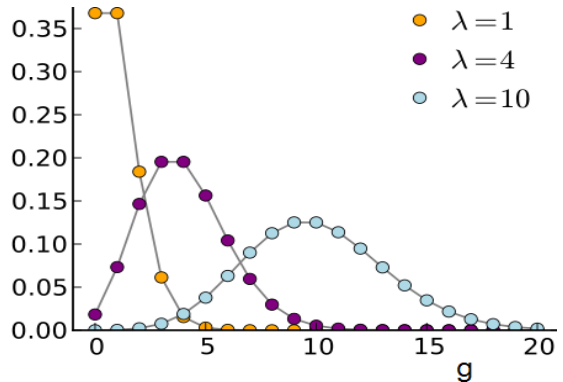
Poiss: *Poisson* distribution with the number of successes g in $\mathbf{X}$ and the *Poisson* parameter λ in $\mathbf{J}$.

POISS_p computes

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

and

POISS_A returns the corresponding *CDF* for the maximum number of successes m in $\mathbf{X}$.



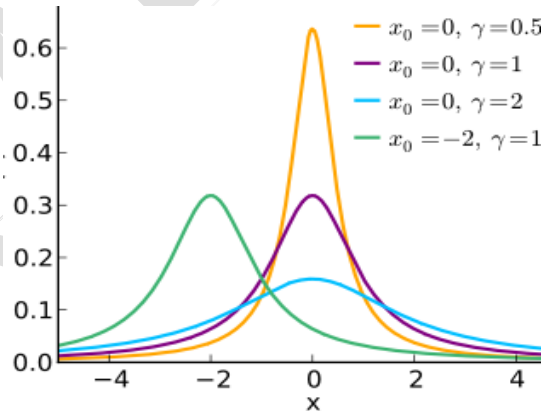
The *Poisson* distribution

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 ($\rightarrow$ **BINOM**).

For the problem introduced with **BINOM** above, **POISS** returns 0.638.¹⁹²

Continuous distributions:

Cauch: *Cauchy-Lorentz* distribution (a.k.a. *Lorentz* or *Breit-Wigner* distribution) with the peak location x_0 specified in $\mathbf{I}$ and the shape γ in $\mathbf{J}$. This distribution is quite popular in physics. It is a special case of *Student's t distribution*.¹⁹³



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

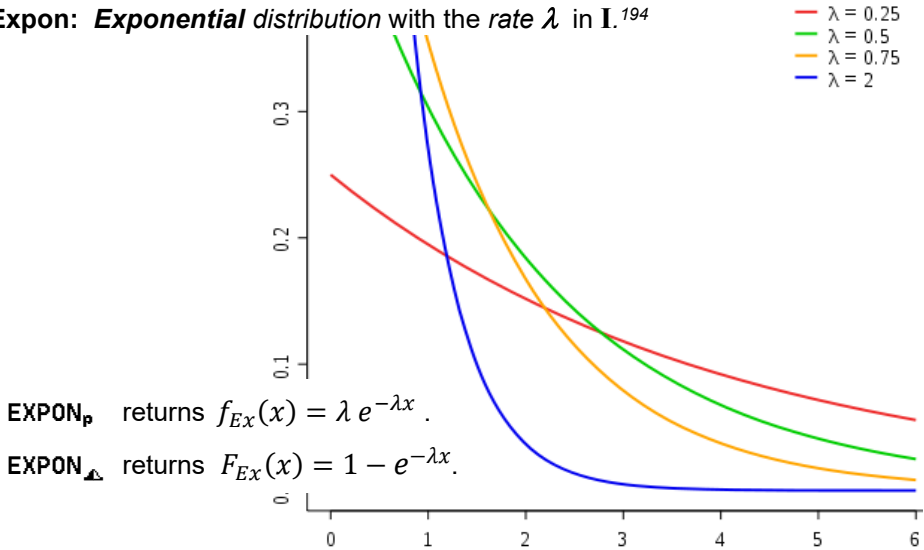
CAUCH_A returns
$$F_{Ca}(x) = \frac{1}{2} + \frac{1}{\pi} \arctan \left(\frac{x - x_0}{\gamma} \right)$$

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

¹⁹² $\rightarrow$ <http://www.itl.nist.gov/div898/handbook/eda/section3/eda366j.htm>

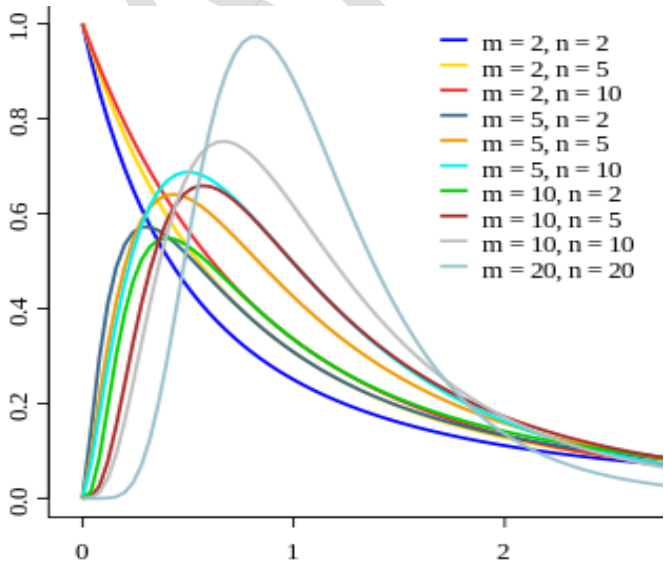
¹⁹³ Start reading http://en.wikipedia.org/wiki/Cauchy_distribution.

Expon: *Exponential* distribution with the rate λ in **I**.¹⁹⁴



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 presents the *PDFs* plotted for different *degrees of freedom* m and n corresponding to i and j .¹⁹⁵



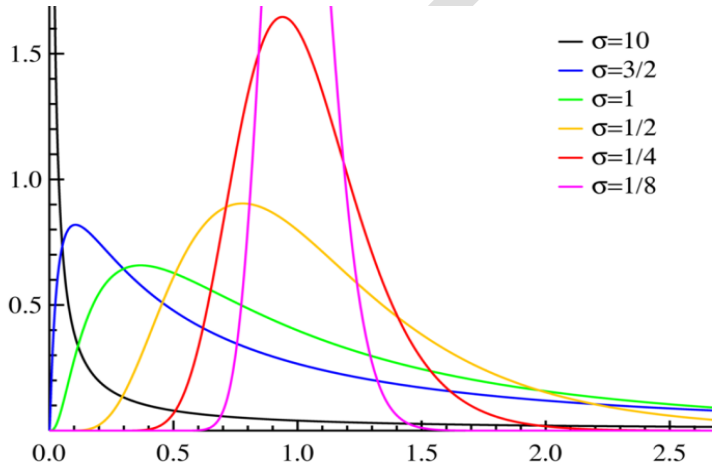
¹⁹⁴ → <http://www.itl.nist.gov/div898/handbook/eda/section3/eda3667.htm>.

¹⁹⁵ → <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** ($\rightarrow$ some *PDF* plots for $\mu = 0$ below).

LGNRM_p 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. 279.¹⁹⁶



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

Substituting $\xi = x - \mu/s$,

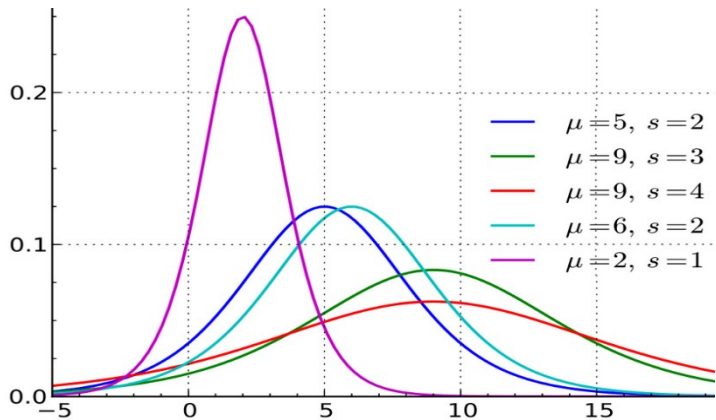
LOGIS_p returns
$$f_{Lg}(x) = \frac{e^{-\xi}}{(1 + e^{-\xi})^2 s}$$

LOGIS_A returns
$$F_{Lg}(x) = 1/(1 + e^{-\xi})$$

LOGIS⁻¹ returns¹⁹⁷
$$F_{Lg}^{-1}(p) = \mu + s \ln\left(\frac{p}{1-p}\right)$$

¹⁹⁶ $\rightarrow$ <http://www.itl.nist.gov/div898/handbook/eda/section3/eda3669.htm>

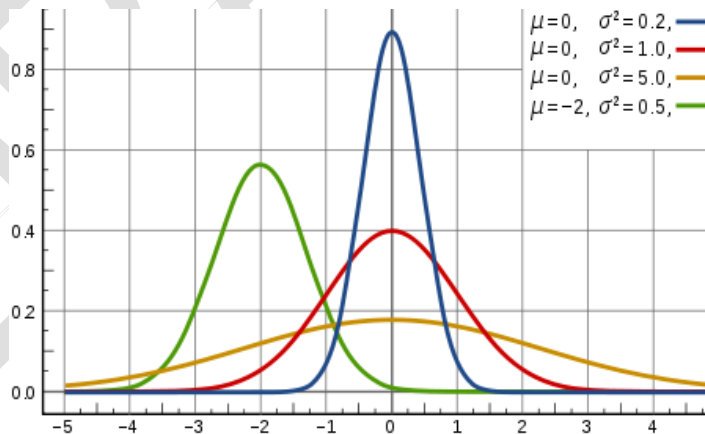
¹⁹⁷ Start reading here for more: http://en.wikipedia.org/wiki/Logistic_distribution



Norml: *Normal* distribution with a *mean* μ given in **I** and an *SD* σ in **J**.

The **red** **NORML_p** returns $f_N(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \varphi\left(\frac{x-\mu}{\sigma}\right)$

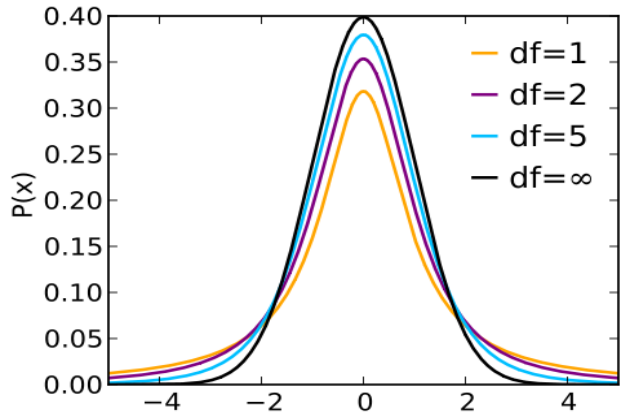
And **NORML_c** returns $F_N(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$ with $\Phi(z)$ denoting the *standardized normal (Gaussian) CDF* ($\rightarrow$ error function on p. 326).¹⁹⁸



t(x): Standardized *Student's t* distribution with its *degrees of freedom* (**df**) in **I**.
It is used for hypothesis testing and calculating confidence intervals, e. g. for

¹⁹⁸ $\rightarrow$ <http://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm>.

means. The diagram shows its *PDFs* plotted for different *degrees of freedom*. For $df \rightarrow \infty$, the shoulders of $t(x)$ shrink and it approaches the **standard-normal distribution** (compare the red curve for **NORML_p** on previous page).¹⁹⁹



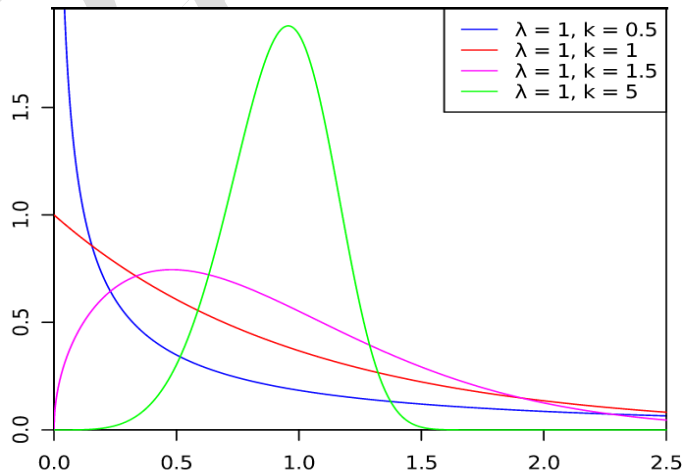
Weibl: **Weibull distribution** with its shape parameter b in **I** and its characteristic lifetime T in **J**.

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, $\rightarrow$ the curves below (with $k = b$ and $\lambda = 1/T$).

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

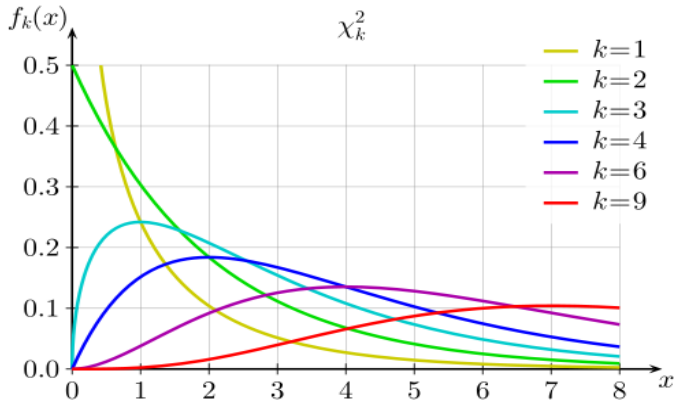
The **Weibull distribution** is widely used for analyzing lifetimes of tools or products ($\rightarrow$ example in **OMS 2**).²⁰⁰



¹⁹⁹ $\rightarrow$ <http://www.itl.nist.gov/div898/handbook/eda/section3/eda3664.htm>.

²⁰⁰ $\rightarrow$ <http://www.itl.nist.gov/div898/handbook/eda/section3/eda3668.htm>. You may even find some more application areas of this distribution mentioned in https://en.wikipedia.org/wiki/Weibull_distribution#Applications.

$\chi^2(x)$: **Chi-square** distribution with its degrees of freedom given in **I**. It is used for calculating confidence intervals for SDs, variances, process and machine capabilities, and the like. The diagram shows PDF's for different degrees of freedom.²⁰¹



More Statistical Formulas, also for Curve Fitting

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

- For samples drawn out of a *normally* distributed (additive) process, the expected value is the *arithmetic mean* (or *average*) of the sample values and its uncertainty is given by its *standard error* ($\rightarrow \bar{x}$ and s_m).
- For samples drawn out of a *log-normally* distributed (multiplicative) process, the expected value is the *geometric mean* and its uncertainty is given by its *scattering factor* ($\rightarrow \bar{x}_g$ 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. Be assured not everything is *normal* (*Gaussian*) in real world.²⁰² Process characteristics can be detected (and should be

²⁰¹ <http://www.itl.nist.gov/div898/handbook/eda/section3/eda3666.htm>

²⁰² In real life, drastic deviations from the (usually assumed *normal*) model distribution are often found – then you can't expect the calculated consequences matching reality any better, though.

Being at it: The *PDF* of a *normal distribution* will never reach zero. So 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.

checked well in advance of calculating, e.g., means) using suitable tests – turn to applicable statistical reference literature.

The following functions as named in left column (sorted alphabetically) are all found in STAT or FIT. All sums Σ and products Π printed in this chapter run from **1** to ***n***:²⁰³

Name	Remarks (→ 14ff 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 <i>OLS</i> . ²⁰⁴ → 276 for shapes of such peaks.
CORR (r)	For any set of data points (x_i, y_i) , the <i>coefficient of correlation</i> is $r = s_{xy} / s_x s_y$. → s_{xy} , s , and dedicated formulas for r or r² in the various fit models on pp. 290ff. For an arbitrary fit model $R(x)$, $r^2 = 1 - \frac{\sum [R(x_i) - y_i]^2}{\sum (\bar{y} - y_i)^2}$

Nevertheless, the *Gaussian distribution* is a very successful model describing a lot of real-world observations very well. Just never forget the limits of such models.

As mentioned in [OMS 2](#), we recommend you look deeply into statistics textbooks to ensure you fully understand what you do with the functions provided in your *WP43*. Remember also f. 164.

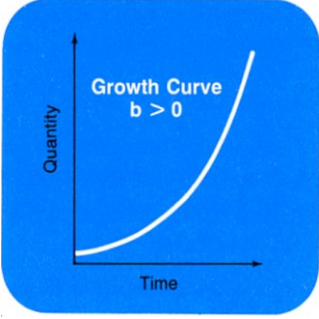
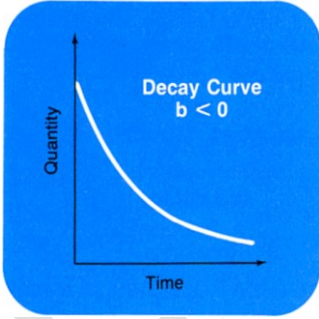
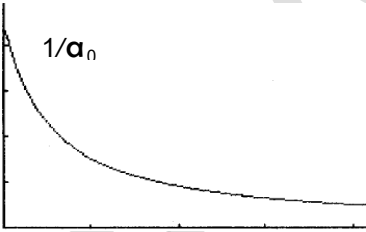
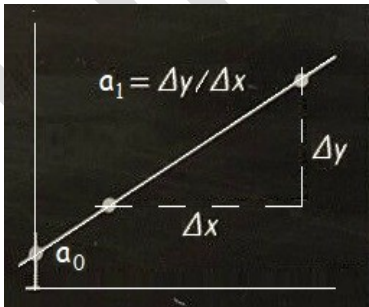
²⁰³ Diagrams in this chapter were created using either vintage *HP* manuals of the 1970's or (if drawn black on white) the book *Curve Fitting for Programmable Calculators* by [W. M. Kolb](#), 1st ed., 1982.

²⁰⁴ Note that an *ordinary least squares regression* (*OLS*) works best for (horizontal) **x** being the independent and (vertical) **y** being the dependent variable. In any case, data point errors in direction **y** shall be significantly greater than those in direction **x**. Swap axes if the situation differs in your problem. For an *OLS*, the sum of the squared (vertical) **y**-distances from the data points to the fit line will be minimized.

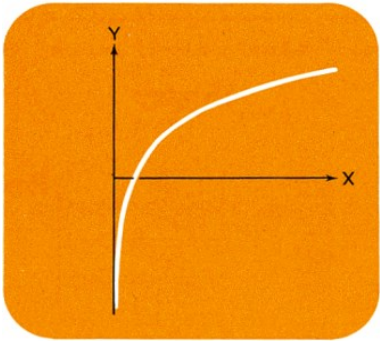
→ **OrthoF** below for a different regression model (*TLS*) and next chapter for more about all the curve fit models provided on your *WP43*.

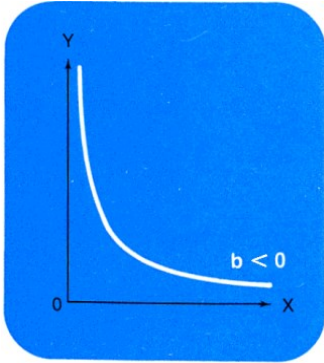
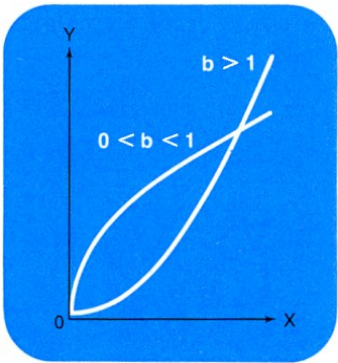
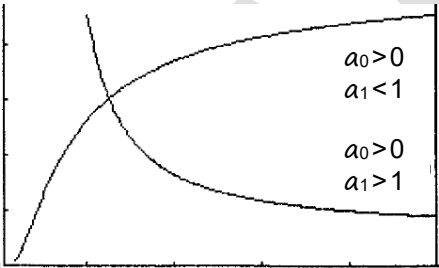
Name	Remarks (→ 14ff for general information)
	is its <i>coefficient of determination</i> (indicating the fraction of the variation of the dependent data y determined by the variation of the independent data x). For $r^2 = 1$, y is fully determined by x; for $r^2 = 0$, y is completely independent of 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 you allowed (→ 25). A 2-parameter regression (7 of the 10 fit models provided on your WP43) is called 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 t <i>distribution</i> for the <i>degrees of freedom</i> $n-2$ (→ 279).
COV, s _{xy}	For any set of data points (x_i, y_i), the <i>population covariance</i> is $COV_{xy} = \frac{1}{n^2} \left(n \sum x_i y_i - \sum x_i \sum y_i \right)$ and the <i>sample covariance</i> is $s_{xy} = \frac{1}{n(n-1)} \left(n \sum x_i y_i - \sum x_i \sum y_i \right).$ Note covariances check linear dependencies only.²⁰⁵

²⁰⁵ → <https://de.wikipedia.org/wiki/Stichprobenkovarianz#/media/Datei:CovarianceCorrelation.svg>

Name	Remarks (→ 14ff for general information)
ExpF	Selects the exponential curve fit model $R(x) = a e^{bx} = a_0 e^{a_1 x}$ for OLS.²⁰⁴ Generally, this will be a good choice if the measured data follow the shape of one of the two curves pictured (think of human population growth or nuclear decay).  
GaussF	Selects a <i>Gauss</i> peak fit model $R(x) = a_0 e^{\frac{(x-a_1)^2}{a_2}}$ for OLS.²⁰⁴ → 279 for shapes of such peaks.
HypF	Selects the hyperbolic fit model $R(x) = 1/(a_0 + a_1 x)$ for OLS.²⁰⁴ 
LinF	Selects the linear fit model $R(x) = a_0 + a_1 x$ for OLS (→ 282). Generally, this will be a good choice if the measured data follow a straight line, raising or falling (but compare OrthoF below).²⁰⁶ 

²⁰⁶ If you want to detect if there is more than just a single linear relation underlying your experimental data (i.e. if the slope changes), look at this: <https://xkcd.com/2701/>

Name	Remarks (→ 14ff for general information)
LogF	 Selects the logarithmic curve fit model $R(x) = a_0 + a_1 \ln(x)$ for OLS (→ 282). Generally, this will be a good choice if the measured data follow a curve looking like drawn at left.
L.R.	Uses the fit model selected and computes the <i>m</i> (i.e. 2 or 3) parameters of the regression curve through the <i>n</i> data points accumulated. Note <i>m</i> must not exceed <i>n</i>. For all fit models provided on your WP43, a regression parameter is (statistically) <i>significant</i> at a 99% confidence level if $ a_i /s(a_i) > t_{n-m}^{-1}(0.995),$ with the right hand side being the inverse of the <i>t distribution</i> for <i>n – m</i> degrees of freedom (→ 279).
OrthoF	Selects a linear fit model $R(x) = a_0 + a_1x$ like LinF, but assumes data point errors both in <i>x</i> and <i>y</i> being of (about) equal size. The sum of the squared (orthogonal) distances from the data points to the fit line will be minimized. Hence, this model is called <i>orthogonal regression</i> (or <i>total least squares</i>, TLS; a.k.a. <i>Deming regression</i>; compare p. 282). The SD of the data points from the fit line is $s_o = s_x s_y \sqrt{\frac{1 - r^2}{s_x^2 + r^2 s_y^2}}$ s_o equals <i>s_{mi}</i> if ≥ 30 points were generated as described in the application examples in OMS 2. → also pp. 290ff and 299ff.
ParabF	Selects a parabolic fit model $R(x) = a_0 + a_1x + a_2x^2$ for OLS (→ 282).

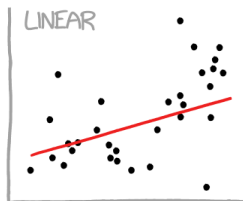
Name	Remarks (→ 14ff for general information)
PowerF	Selects the power curve fit model $R(x) = a x^b = a_0 x^{a_1}$ for OLS (→ 282). This will be a good choice if measured data follow the shape of one of the curves pictured here (look for the <i>Tower of Pisa</i> in the OM).  
RootF	 Selects the root curve fit model $R(x) = a_0 a_1^{1/x}$ for OLS (→ 282). $a_0 > 0$ $a_1 < 1$ $a_0 > 0$ $a_1 > 1$
s, s_m	The sample SD s is the positive square root of the <i>variance</i> of a normally distributed sample: $s_x^2 = \frac{n \sum x_i^2 - (\sum x_i)^2}{n(n-1)} = \frac{\sum x_i^2 - n \bar{x}^2}{n-1}$ And the <i>standard error</i> (i.e. the SD of the <i>mean</i> $\bar{x}$) is $s_m = s / \sqrt{n}$
s_{mi}	The SD of the <i>measuring instrument</i> as measured using the MSA shown on pp. 299ff. → also OrthoF and OMS 2.

Name	Remarks (→ 14ff for general information)
s_w, s_{mw}	The sample SD for <u>weighted</u> or grouped data (where the weight or frequency y_i of each data point x_i was entered via $\Sigma+$) is $s_w = \sqrt{\frac{\sum y_i \sum y_i x_i^2 - (\sum y_i x_i)^2}{\sum y_i (\sum y_i - 1)}}$ And the <i>standard error</i> (i.e. the SD of the <i>weighted mean</i> $\bar{x}_w$) is $s_{mw} = s_w / \sqrt{\sum y_i}$
s_{xy}	→ COV on p. 283.
$\hat{x}$	Forecast for x for given y from the fit model used, → next chapter.
$\bar{x}$	The arithmetic mean of normally distributed data is $\bar{x} = \frac{1}{n} \sum x_i$
$\bar{x}_G$	The geometric mean of log-normally distributed data is $\bar{x}_G = \sqrt[n]{\prod x_i} = \exp\left[\frac{1}{n} \sum \ln(x_i)\right]$
$\bar{x}_H$	The harmonic mean is calculated as $\bar{x}_H = n / \sum \frac{1}{x_i}$
$\bar{x}_{RMS}$	The quadratic mean (root mean squared) is $\bar{x}_{RMS} = \sqrt{\frac{1}{n} \sum x_i^2}$
$\bar{x}_w$	The arithmetic mean of normally distributed <u>weighted</u> or <u>grouped</u> data ²⁰⁷ is $\bar{x}_w = \sum x_i y_i / \sum y_i$
$\hat{y}$	Forecast for y for given x from the fit model used, → next chapter.

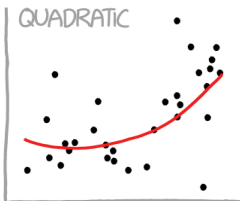
²⁰⁷ where the weight or frequency y_i of each data point x_i was entered via $\Sigma+$

Name	Remarks (→ 14ff for general information)
ε	The <i>scattering factor</i> ε_x for a sample of log-normally distributed collected data is $\ln(\varepsilon_x) = \sqrt{\frac{\sum \ln^2(x_i) - 2n \ln(\bar{x}_G)}{n-1}}$ Compare s.
ε_m	The <i>scattering factor</i> of the <i>geometric mean</i> is $\varepsilon_m = \varepsilon^{1/\sqrt{n}}$ Compare s_m.
ε_p	The <i>scattering factor</i> ε_p for a population of log-normally distributed data is $\ln(\varepsilon_p) = \sqrt{\frac{n-1}{n}} \ln(\varepsilon)$ Compare σ.
σ	The population SD of normally distributed data is $\sigma = \sqrt{\frac{n-1}{n}} s$
σ_w	The population SD of <u>weighted</u> or <u>grouped</u> data ²⁰⁷ is $\sigma_w = \sqrt{\frac{\sum y_i (x_i - \bar{x}_w)^2}{\sum y_i}}$

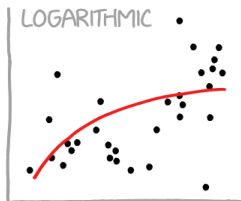
CURVE-FITTING METHODS AND THE MESSAGES THEY SEND



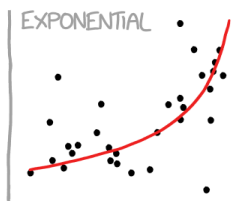
"HEY, I DID A REGRESSION."



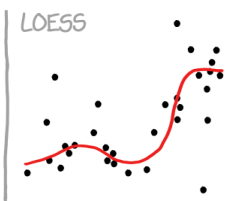
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."



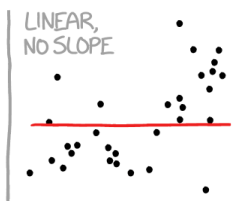
"LOOK, IT'S TAPERING OFF!"



"LOOK, IT'S GROWING UNCONTROLLABLY!"



"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."



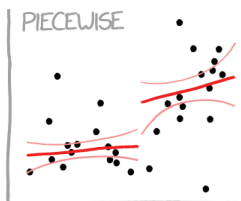
"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."



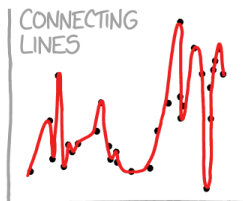
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."



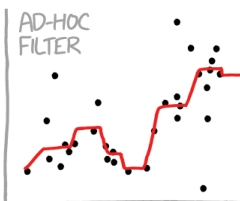
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."



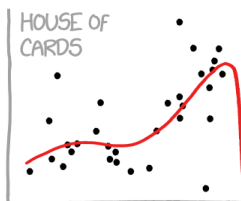
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."



"I CLICKED 'SMOOTH LINES' IN EXCEL."



"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"



"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE- WAIT NO NO DON'T EXTEND IT AAAAAA!!!"

Curve Fitting Models Provided

A proper linear regression is computed for **LinF** and **OrthoF** only. For the other three standard models (**ExpF**, **LogF**, and **PowerF**) the method of **LinF** 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).

Furthermore, your data might follow a straight line if you plot the inverse of your y -data over your x -data (then **HypF** will fit). And there is more below.²⁰⁸

This is what your **WP43** will do when you enter statistical data points and compute the parameters of a fit curve:

1. For each new data point, it puts x and y in a new row of **STATS**, updates the 23 sums listed on pp. 103f (some of them may turn non-numeric for entries ≤ 0 – just don't care), and increments the data point counter n .
2. The subsequent evaluation depends on the fit model you select ($\rightarrow$ 284ff):
 - a. If **LinF** is chosen then the **OLS** ($\rightarrow$ 282) line parameters a_0 and a_1 are computed following the formulas:

$$a_0 = \frac{\sum x_i^2 \cdot \sum y_i - \sum x_i \cdot \sum x_i y_i}{n \sum x_i^2 - (\sum x_i)^2}, \quad a_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} = \frac{s_{xy}}{s_x^2} = r \frac{s_y}{s_x}$$

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

The *standard errors* of a_0 and a_1 are calculated using the formulas

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

$$\text{with } s_x^2 = \frac{n \sum x_i^2 - (\sum x_i)^2}{n(n - 1)}, \quad s_y^2 = \frac{n \sum y_i^2 - (\sum y_i)^2}{n(n - 1)}, \quad \text{and } \bar{x} = \frac{1}{n} \sum x_i.$$

²⁰⁸ The picture on previous page was copied from <https://xkcd.com/2048/>.

The forecast for a given x is $\hat{y} = a_0 + a_1 x$; for a given y it is $\hat{x} = (y - a_0)/a_1$.

The *standard error of estimate* (a.k.a. square root of the *residual variances*) is

$$s_{res} = \sqrt{\frac{n-1}{n-2} (s_y^2 - a_1^2 s_x^2)} = s_y \sqrt{\frac{n-1}{n-2} (1 - r^2)} = s(a_1) s_x \sqrt{n-1}$$

The *standard error of a forecast* $\hat{y}$ as specified above is

$$s(\hat{y}) = s_{res} \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{\sum (x_i - \bar{x})^2}} = s(a_1) \sqrt{\frac{n-1}{n} s_x^2 + (x - \bar{x})^2}$$

Note this error increases with growing distance from the mean.

- b. **If you chose ExpF** then the OLS ($\rightarrow$ 282) line parameters for the transformed data $x_i, \ln(y_i)$ are computed using

$$\begin{aligned} a_{0,tEX} &= \frac{\sum x_i^2 \cdot \sum \ln(y_i) - \sum x_i \cdot \sum x_i \ln(y_i)}{n \sum x_i^2 - (\sum x_i)^2} \\ a_{1,tEX} &= \frac{n \sum x_i \ln(y_i) - \sum x_i \sum \ln(y_i)}{n \sum x_i^2 - (\sum x_i)^2} \\ r_{tEX} &= \frac{n \sum x_i \ln(y_i) - \sum x_i \sum \ln(y_i)}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \cdot \sqrt{n \sum \ln^2(y_i) - [\sum \ln(y_i)]^2}} \end{aligned}$$

The standard errors of $a_{0,tEX}$ and $a_{1,tEX}$ are calculated using the formulas (*) on p. 290 but replacing s_y^2 with

$$s_{ty}^2 = \frac{n \sum \ln^2(y_i) - (\sum \ln(y_i))^2}{n(n-1)} \Rightarrow s(a_{1,tEX}) = \frac{s_{ty}}{s_x} \sqrt{\frac{1 - r_{tEX}^2}{n-2}} \text{ here.}$$

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

The forecast for a given x is $\hat{y} = a_0 e^{a_1 x}$; for a given y it is $\hat{x} = \frac{1}{a_1} \ln(y/a_0)$.

- c. If **PowerF** is chosen then the OLS ($\rightarrow$ 282) line parameters for the transformed data $\ln(x_i)$, $\ln(y_i)$ are computed in analogy to the method shown for **ExpF**. Thus:

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

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

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

The standard errors of $a_{0,tPOW}$ and $a_{1,tPOW}$ are calculated using the formulas (*) on p. 290 but replacing

$$s_x^2 \text{ with } s_{tx}^2 = \frac{1}{n(n-1)} \left[n \sum \ln^2(x_i) - \left(\sum \ln(x_i) \right)^2 \right],$$

$$s_y^2 \text{ with } s_{ty}^2 = \frac{1}{n(n-1)} \left[n \sum \ln^2(y_i) - \left(\sum \ln(y_i) \right)^2 \right],$$

and $\bar{x}$ with $\bar{x}_t = \frac{1}{n} \sum \ln(x_i)$.

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

The forecast for a given x is $\hat{y} = a_0 x^{a_1}$; for a given y it is $\hat{x} = \sqrt[a_1]{y/a_0}$.

- d. If you chose **LogF** then the OLS ($\rightarrow$ 282) line parameters for the transformed data $\ln(x_i)$, y_i are computed in analogy to the method shown for **ExpF**. Thus they are

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

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

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

The standard errors of $a_{0,tLOG}$ and $a_{1,tLOG}$ are calculated using the formulas (*) on p. 290 but replacing

$$s_x^2 \text{ with } s_{tx}^2 = \frac{1}{n(n-1)} \left[n \sum \ln^2(x_i) - \left(\sum \ln(x_i) \right)^2 \right]$$

$$\text{and } \bar{x} \text{ with } \bar{x}_t = \frac{1}{n} \sum \ln(x_i).$$

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

The forecast for a given x is $\hat{y} = a_0 + a_1 \ln(x)$; for a given y it is $\hat{x} = \exp(y - a_0/a_1)$.

- e. **If HypF is chosen** then the parameters of the OLS ($\rightarrow$ 282) curve $R(x) = 1/(a_0 + a_1 x)$ are computed to be

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

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

The forecast for a given x is $\hat{y} = 1/(a_{0,HY} + a_{1,HY} x)$; for a given y it is $\hat{x} = [(1/y) - a_{0,HY}]/a_{1,HY}$.

- f. **If you chose RootF** then the parameters of the OLS ($\rightarrow$ 282) curve are 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 $a_{0,\sqrt{}} = e^B$ and $a_{1,\sqrt{}} = e^C$.

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

The forecast for a given x is $\hat{y} = a_{0,\sqrt{}} \times a_{1,\sqrt{}}^{1/x}$; for a given y it is $\hat{x} = \lg(a_{1,\sqrt{}}) [\lg(y) - \lg(a_{0,\sqrt{}})]$.

- g. **If ParabF is chosen** then the parameters of the OLS ($\rightarrow$ 282) curve are computed using

$$\begin{aligned} A &= n \sum x_i^2 - \left(\sum x_i \right)^2, & B &= n \sum x_i^2 y_i - \sum x_i^2 \cdot \sum y_i, \\ C &= n \sum x_i^3 - \sum x_i^2 \cdot \sum x_i, & D &= n \sum x_i y_i - \sum x_i \cdot \sum y_i, \\ E &= n \sum x_i^4 - \left(\sum x_i^2 \right)^2 \end{aligned}$$

Then the parameters of the fit curve $R(x) = a_0 + a_1 x + a_2 x^2$ are

$$a_2 = \frac{A B - C D}{A E - C^2}, \quad a_1 = \frac{D - a_2 C}{A},$$

$$\text{and } a_0 = \frac{1}{n} \left(\sum y_i - a_2 \sum x_i^2 - a_1 \sum x_i \right).$$

$$\text{And } r_{PA}^2 = \frac{a_0 \sum y_i + a_1 \sum x_i y_i + a_2 \sum x_i^2 y_i - \frac{1}{n} (\sum y_i)^2}{\sum y_i^2 - \frac{1}{n} (\sum y_i)^2}$$

The forecast for a given x is $\hat{y} = a_0 + a_1 x + a_2 x^2$; for a given y it is

$$\hat{x}_{1,2} = \frac{1}{2a_2} \left(-a_1 \pm \sqrt{a_1^2 - 4a_2(a_0 - y)} \right).$$

- h. **If you chose GaussF** then the parameters of the OLS ($\rightarrow$ 282) curve are computed using the auxiliary terms A , C , and E exactly as in **ParabF**. The other terms are

$$B = n \sum x_i^2 \ln(y_i) - \sum x_i^2 \cdot \sum \ln(y_i),$$

$$D = n \sum x_i \ln(y_i) - \sum x_i \cdot \sum \ln(y_i), \quad F = \frac{A B - C D}{A E - C^2}, \quad G = \frac{D - F C}{A},$$

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

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

$$a_2 = \frac{1}{F}, \quad a_1 = -\frac{G}{2} a_2 \quad \text{and} \quad a_0 = e^{H - F a_1^2}.$$

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

The forecast for a given x is $\hat{y} = a_0 e^{(x-a_1)^2/a_2}$; for a given y it is

$$\hat{x}_{1,2} = a_1 \pm \sqrt{a_2 \ln(y/a_0)}$$

- i. **If CauchF is chosen** then the parameters of the OLS ($\rightarrow$ 282) curve are computed using the auxiliary terms A , C , and E exactly as in **ParabF**. The other terms are

$$B = n \sum \frac{x_i^2}{y_i} - \sum x_i^2 \cdot \sum \frac{1}{y_i} \quad \text{and} \quad D = n \sum \frac{x_i}{y_i} - \sum x_i \cdot \sum \frac{1}{y_i}.$$

F and G are calculated as for **GaussF** but with the components computed here; and

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

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

$$a_0 = F, \quad a_1 = \frac{G}{2a_0}, \quad \text{and} \quad a_2 = H - F a_1^2.$$

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

The forecast for a given x is $\hat{y} = 1/[a_0(x + a_1)^2 + a_2]$; and the forecast for a given y is

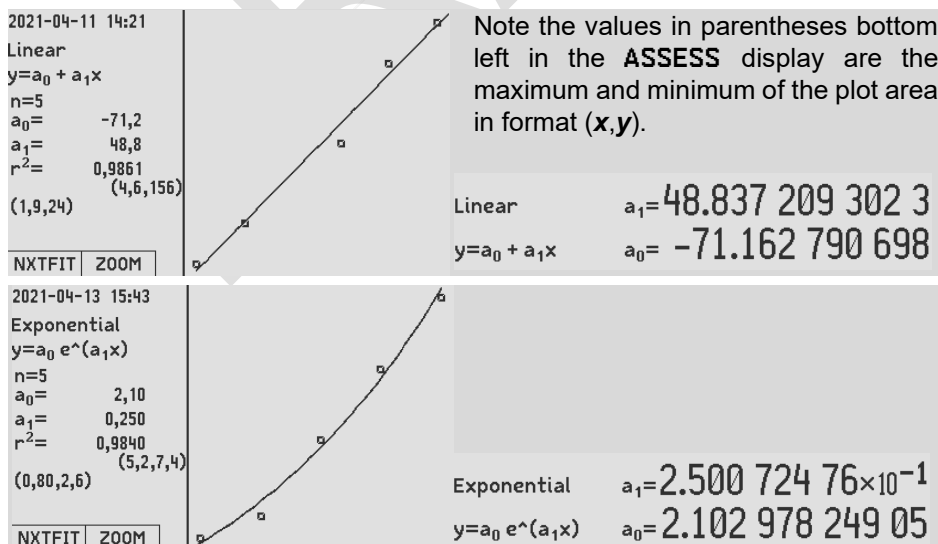
$$\hat{x}_{1,2} = -a_1 \pm \sqrt{\frac{1}{a_0} \left(\frac{1}{y} - a_2 \right)}$$

- j. **If you chose 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 ($\rightarrow$ **IOI**). The model delivering the greatest r^2 will be selected as best fitting model.
- k. **If you chose OrthoF** then the total least squares regression line parameters a_0 and a_1 will be computed following the formulas

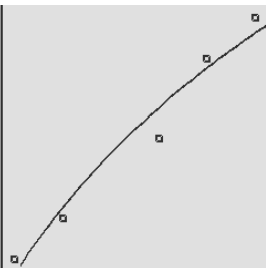
$$a_1 = \frac{1}{2s_{xy}} \left[s_y^2 - s_x^2 + \sqrt{(s_y^2 - s_x^2)^2 + 4s_{xy}^2} \right] \quad \text{and} \quad a_0 = \bar{y} - a_1 \bar{x}$$

assuming the data point errors in direction **y** and **x** are about equal. The other formulas for **OrthoF** can be taken over from **LinF**.

The outputs of ASSESS (left, to scale) and **L.R.** are generally formatted as shown below, allowing for up to 11 digits displayed for each model parameter (far more than usually significant). The outputs below are just dummies for demonstration – some fit models do not make much sense for the data used:

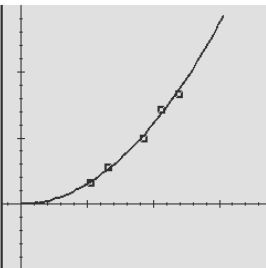


2021-04-11 14:18
 Logarithmic
 $y=a_0+a_1\ln(x)$
 $n=5$
 $a_0=-80,5$
 $a_1=148$
 $r^2=0,9614$
 $(1,9,24)$
 $(4,6,156)$



Logarithmic $a_1=148.222\ 520\ 145$
 $y=a_0+a_1\ln(x)$ $a_0=-80.532\ 162\ 769$

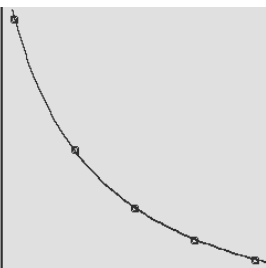
2021-04-18 17:12
 Power*
 $y=a_0x^{a_1}$
 $n=5$
 $a_0=7.72$
 $a_1=1.99$
 $r^2=0.9952$
 $(-0.53,-92)$
 $(7.0,271)$



If zero is included in the drawing range (which may also be achieved by zooming out), the respective axis or axes will be drawn ($\rightarrow$ also **CauchF** and **GaussF**).

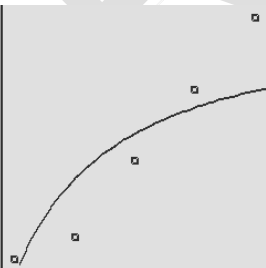
Power $a_1=-8.052\ 852\ 4\times 10^{-1}$
 $y=a_0x^{a_1}$ $a_0=5.229\ 414\ 822\ 53$

2021-04-13 16:01
 Hyperbolic
 $y=(a_0+a_1x)^{-1}$
 $n=5$
 $a_0=0,0645$
 $a_1=0,130$
 $r^2=0,9999$
 $(0,80,1,2)$
 $(5,2,5,3)$



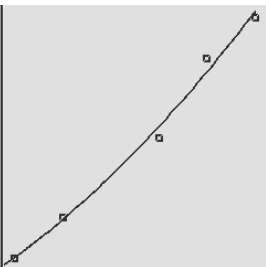
Hyperbolic $a_1=1.302\ 069\ 21\times 10^{-1}$
 $y=(a_0+a_1x)^{-1}$ $a_0=6.452\ 434\ 35\times 10^{-2}$

2021-04-13 15:35
 Root
 $y=a_0a_1^{(1/x)}$
 $n=5$
 $a_0=7,27$
 $a_1=0,342$
 $r^2=0,7634$
 $(0,80,2,6)$
 $(5,2,7,4)$

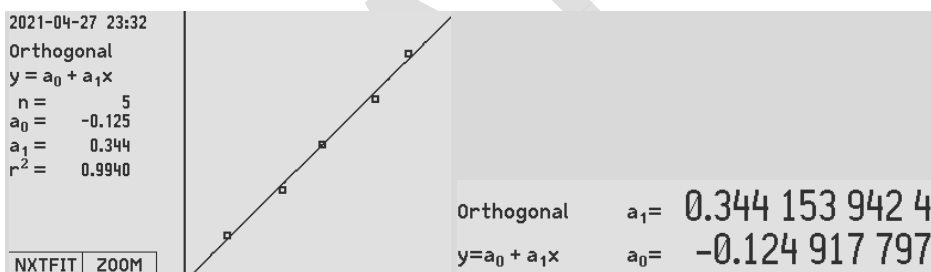
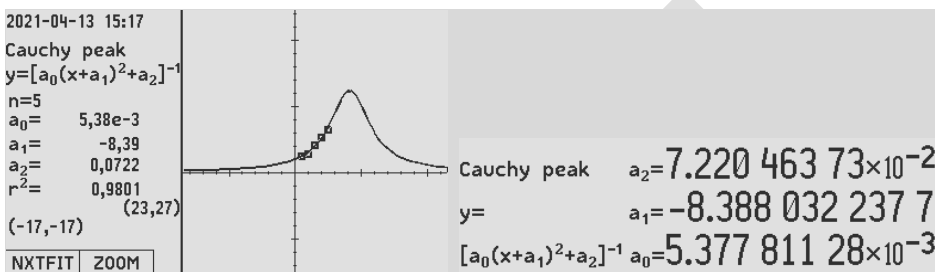
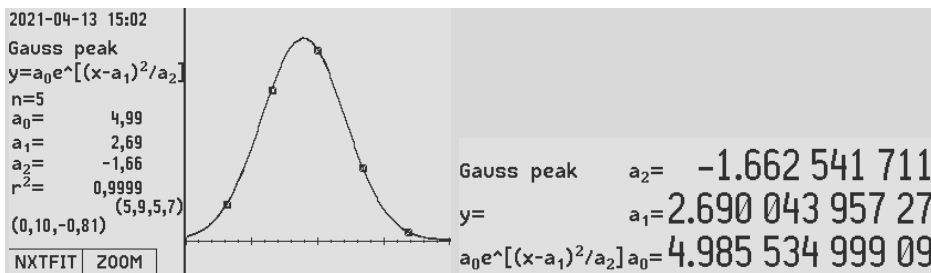


Root $a_1=0.341\ 758\ 18$
 $y=a_0a_1^{(1/x)}$ $a_0=7.270\ 848\ 526\ 85$

2021-04-11 14:26
 Parabolic
 $y=a_0+a_1x+a_2x^2$
 $n=5$
 $a_0=-19,1$
 $a_1=13,6$
 $a_2=5,45$
 $r^2=0,9913$
 $(1,9,24)$
 $(4,6,156)$



Parabolic $a_2=5.454\ 545\ 454\ 55$
 $y=a_0+a_1x+a_2x^2$ $a_1=13.636\ 363\ 636\ 4$
 $a_0=-19.090\ 909\ 091$



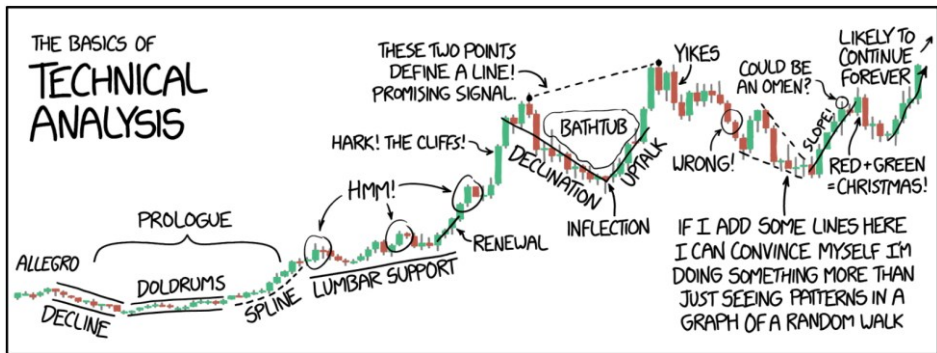
About general polynomial fits to n measured data points:

LinF and **ParabF** above are polynomials of *grade* 1 and 2. For a polynomial $R(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_g x^g$ of *grade* $g < n$, the system of linear equations determining its parameters looks like

$$Y = X A \text{ with } Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^g \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^g \end{pmatrix} \text{ and } A = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_g \end{pmatrix}.$$

Its solution is $A = (X^T X)^{-1} X^T Y$. Your **WP43** will help you solving this matrix equation. Though, do not exaggerate ($\rightarrow$ <https://xkcd.com/2101/>):²⁰⁹

²⁰⁹ *Science* grows in continuous competition between experiment and theory. Theory



The Pedestrian's Measurement System Analysis (MSA)

Knowing the measuring instrument or system you use is essential for assessing its indications and results (→ f. 71 on pp. 116f). To measure its actual precision under real-world conditions easily, proceed as explained below:

1. Create a square diagram with equal axes.

Some fundamentals: Assume a constant object **A** and an ideal measuring instrument in constant environment. Measure a particular dimension **x** of **A** twice independently under unchanged conditions using this instrument; you will get two readings **x₁** and **x₂**. Both will be equal (due to the ideal instrument and constant conditions), so the data point (**x₁**, **x₂**) will be located perfectly on the (45°) diagonal of said diagram.

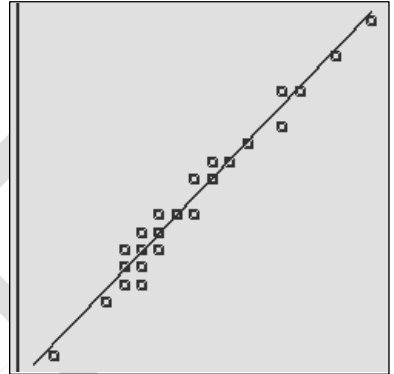
But all technical processes scatter; real measuring processes do as well (actually, uncertainty is inherent in any physical measurement). Thus, taking a large number of independent high resolution measurements **x** of **A**, using the same measuring system, combining every two subsequent measurements to one data point (**x₁**, **x₂**), and plotting these points will result in a circular cloud of normally distributed points centered on the diagonal. The cloud diameter will be the smaller the better the measuring system is.

interprets observations but also creates predictions in a range where no observations exist so far. Experimental observations may corroborate a theory or falsify it.

Did you ever read or hear of a *Technical Analysis* comparing its predictions to what really happened? Please think about it. Such methods may be called *voodoo* at other places – but don't tell financial people or you might be shot (shoot the messenger!).

Investigating two different, constant objects this way will result in two clouds of said kind, three in three, etc. Instead of measuring just very few different objects many times, however, it is advantageous measuring many different objects just two times (the returns will be more diversified and deliver more information). Thus:

2. Perform two successive, independent series of measurements of a set of constant objects $A_1 \dots A_n$ under at most unchanged conditions using the same instrument.
3. Plot the data points (x_{1j}, x_{2j}) for all A_j in said diagram. They should build an ant trail scattering around the diagonal like here (→ also [OMS 2](#)).
4. If its center line coincides with the 45° line (→ [OMS 2](#)) and the trail is 'narrow enough' then the instrument employed is considered 'sufficiently precise'. For usual narrow trails, ≥ 30 samples are required to achieve a reliable result for the uncertainty of the measuring system by determining the trail width



$$W^2 = s_x^2 s_y^2 \frac{1 - r^2}{s_x^2 + r^2 s_y^2} = s_y^2 \frac{1 - r^2}{1 + a_1^2} = s_{mi}^2$$

If $30 s_{mi} \leq T$ then this system is precise enough (a.k.a. capable) for controlling a full tolerance band width T under said conditions.²¹⁰

Further analyses of the measuring system investigated are viable by interpreting such diagrams in more detail. They require just plain logic and common sense. Think! Undeniable advantage of this method: objects will never lie to you.

If you collect your ($\geq$) 30 test objects from a steady state production process, you can determine not only the real-life precision of the measuring system used but also the *SD* of this production process. Then the ratio $(L - W)/W$ (with L being the trail length) shall be greater than a fixed limit B to warrant this measuring system being capable to control this production process (→ [OMS 2](#)).

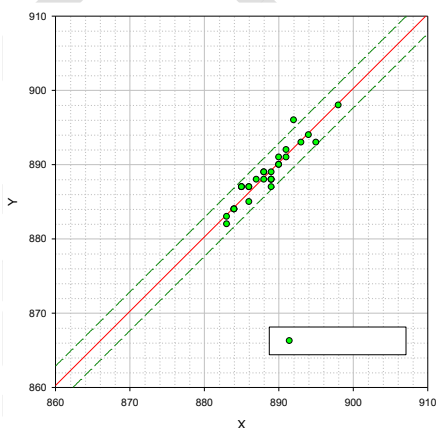
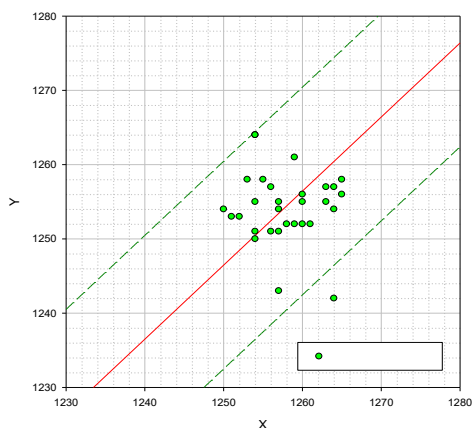
²¹⁰ The number 30 appears twice here but independently.

Let's emphasize: s_{mi} measured this way reflects the conditions prevalent during the measurements. This is wanted since you strive for knowing a realistic precision, e.g., at the place the instrument will be deployed later on. So you do not have to rely on the values in its data sheet (usually determined under 'ideal' conditions you don't know).

Automotive industry demands **B = 9.5**. Other industries may be less demanding here – a minimum **B = 7.5** shall be achieved.

With many samples approximately uniformly distributed over a certain range the measured points of the overall trail will fill a sausage (i.e. an oval, not an ellipse!).

Plots of this kind may also support you monitoring industrial processes, directly comparing various instruments, machine settings, etc. Find below, for example, a pair of plots of parts produced on an automated high-speed precision machine set up in two different ways (one small square of the grid represents $2 \times 2 \mu\text{m}$, the dashed lines are $\pm 2 s_{mi}$ from the red center line fitted through the 30 points measured).²¹¹ Evidently the production precision could be improved by a factor 5, applying a relatively simple change in the machine setup:



This method of displaying measured data is very efficient. Although taught in seminars and workshops (and advertised under various names), applicable literature is scarce.²¹² Note you can use this analysis method for qualifying any measuring system or process (mechanical, electronical, whatever) at an arbitrary location in the universe – the one and only hard requisite is it shall leave your test samples constant (but it will even tell you when this condition was not fulfilled).

²¹¹ These diagrams were created using [SigmaPlot](#), an excellent software for scientific graphing and drawing. I can highly recommend it – it really does what you tell it to do (unlike [MS Excel](#)) and also has some powerful analysis features implemented.

²¹² → <https://blog.minitab.com/en/real-world-quality-improvement/orthogonal-regression-testing-the-equivalence-of-instruments>, for example. Actually, our MSA diagram grew out of a [Shainin](#) workshop about industrial process improvement I attended in 1996.

Error Propagation in Calculations

Experimental data always go with errors, caused by, e.g., the uncertainties of the measuring method, the instrument used, and environmental variations (→ f. 71 on p. 116). Even under carefully controlled environmental and measuring conditions, random errors remain (→ previous chapter). These errors must be taken into account for a proper estimation of the uncertainties 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 n independent experimental parameters $x_1 \dots x_n$. Each such parameter x_i shows an uncertainty or 'error' Δx_i . Now, if $R = f(x_1, \dots, x_n)$ then its total 'error' is

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

Often, the differential terms under the square root are either unknown or tedious to determine analytically. Luckily, however, this root can be simplified:

$\Delta R = \pm \sqrt{\Delta f_1^2 + \dots + \Delta f_n^2}$. And with your *WP43*, the following will do for computing ΔR , even if f should be 'very curvy':

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 *WP43* compute $R = f(x_1, x_2, \dots, x_n)$.
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. Repeat the last two steps for each parameter from x_2 up to x_n .

In four decades of professional experience, I found such correlation diagrams being the most powerful although yet simple tools for easily assessing the quality of real-life processes in arbitrary environments. They ease convincing other people as well, even non-tech. Such plots support decisions far better than staring at numbers. Mankind has got simply far longer experience with pictures (diagrams) than with numbers.

Be very careful when employing *MS Excel* for plotting in this matter: it is unable to draw square diagrams reliably; plots vary depending on the particular output device used.

Being through with all n parameters, you will eventually get

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

The summands under the square root have become simple differences which are determined most easily using your *WP43*.

For 'small errors' or less curvature of f , a 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 *WP43* compute $R = f(x_1, x_2, \dots, x_n)$.
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 parameter from x_3 up to x_n .

Being through with all n parameters, you will end with

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

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

At the bottom line, we have replaced a formula requiring a lot of symbolic mathematics and few numeric calculations by an equivalent formula requiring no symbolic mathematics at all but many numeric calculations. Programmed, your *WP43* can do all these repetitive calculations for you automatically. Determining derivatives is not required anymore.

Solving Differential Equations (1)

The method applied to the examples in [OMS 4](#) develops as explained below: ²¹³

First, we are going to solve a **1D** problem of the kind

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

²¹³ The procedure coarsely follows [R. M. Eisberg](#), *Applied Mathematical Physics with Programmable Pocket Calculators*, McGraw-Hill, New York, 1976.

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

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

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

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

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

Using the following, so-called *half-step method*, however, a better approximation of the slope of f is achieved principally:

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

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

Proceeding from time 0 in small steps Δt again, we get $\left(\frac{df}{dt}\right)_{1/2}$ as above, and

$$f_1 \approx f_0 + \left(\frac{df}{dt}\right)_{1/2} \Delta t, \quad \left(\frac{df}{dt}\right)_{3/2} \approx \left(\frac{df}{dt}\right)_{1/2} + a\Delta t,$$

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

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. 303, we will get the following new set:

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

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

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

Proceeding from time zero in small steps Δt again, we get $\left(\frac{df}{dt}\right)_{1/2}$ as above, as well as

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

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

This *half-step method* as explained above can be applied easily to all ordinary differential equations of 2nd order which can be written like

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

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

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

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

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

For solving a **2D** problem, e.g., finding the orbit of a satellite in the gravitational field of Earth, we need two differential equations, one for $\mathbf{x}$ and one for $\mathbf{y}$.

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

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

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

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

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

Example:

Despite of the crosstalk (a.k.a. coupling) between **x** and **y**, we can compute the coordinates **x** and **y** of the satellite as functions of time, letting the *WP43* proceed like we did in the *1D* examples in *OMS 4*:

For ease of handling in a first computation, we set $GM = 1 = a$ and the start values

$$x_0 = 1, \quad \left(\frac{dx}{dt}\right)_0 = 0, \quad y_0 = 0, \quad \left(\frac{dy}{dt}\right)_0 = 1.$$

These 'variable' start values shall be entered in our program using INPUT here:

```
LBL 'Satell'
  REM 'Begin of variable initialization'
  INPUT 'x'
  INPUT 'dxbydt'  remember a slash is disallowed in a name
  INPUT 'y'
  INPUT 'dybydt'
  .1
  STO 'Δt'        initialize the remaining fixed start values
  1
  (for Earth satellites, take GM⊕ from CONST which
  STO 'a'          would look like # GM⊕ instead of 1 here)
  0
  start at time zero
  STO 't'
  REM 'End of variable initialization'

LBL 01
  REM 'Begin of the (infinite) time loop'
  RCL 'y'
  RCL 'y'
  x2              y2
  RCL 'x'
  x2
  +               y2+x2
```

-1.5

y^x

RCL $\times$ 'a'

$\times$

RCL L

RCL $\times$ 'x'

RCL 't'

$x > 0.?$

GTO 02

$(y^2+x^2)^{-1.5}$

$a (y^2+x^2)^{-1.5}$

$y a (y^2+x^2)^{-1.5} = -K_y$

$a (y^2+x^2)^{-1.5}$

$x a (y^2+x^2)^{-1.5} = -K_x$. The stack is $[-K_x, -K_y, \dots]$ now.

check time – it will be zero in 1st run

from 2nd run on go to *local label* 02

REM 'The next steps are for the first run only'

DROP $\downarrow$

forget t

2

/

$-K_x / 2$

$x \rightrightarrows y$

The stack is $[-K_y, -K_x / 2, \dots]$ after $x \rightrightarrows y$

2

/

$-K_y / 2$

$x \rightrightarrows y$

The stack is $[-K_x / 2, -K_y / 2, \dots]$ after $x \rightrightarrows y$

GTO 03

go to common part of time loop

LBL 02

REM 'This is used from second run on'

DROP $\downarrow$

forget t

LBL 03

REM 'The common part of the time loop resumes here'

RCL $\times$ ' Δt '

$-K_x \times \Delta t$ (or half of it in 1st run)

STO- ' $dxbydt$ '

calculate the new dx/dt

DROP $\downarrow$

bring y to the front

RCL $\times$ ' Δt '

$-K_y \times \Delta t$ (or half of it in 1st run)

STO- ' $dybydt$ '

calculate the new dy/dt

RCL ' Δt '

STO+ ' t '

calculate the new time

RCL $\times$ ' $dxbydt$ '

$dx/dt \times \Delta t$

STO+ ' x '

calculate the new x

VIEW ' x '

and show it for the new point (x, y) to be plotted

STOP

RCL ' Δt '

RCL $\times$ ' $dybydt$ '

$dy/dt \times \Delta t$

STO+ ' y '

calculate the new y

VIEW ' y '

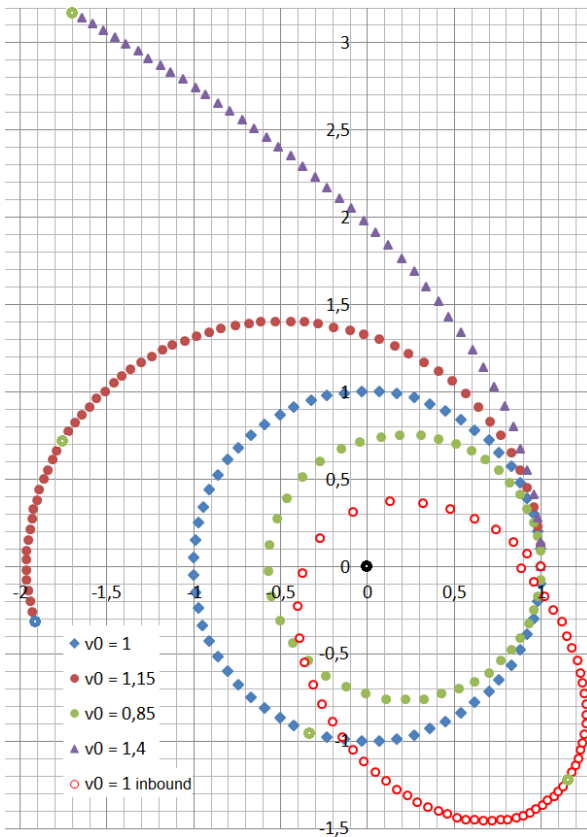
and show it for the new point (x, y) to be plotted

STOP

GTO 01

REM 'End of time loop'

END



Plotting the points calculated for these start values will result in a perfect circle as shown by the blue symbols in the diagram below – taking $64 \Delta t$ for one full orbit. We added examples with slightly different start velocities for comparison. The green elliptical orbit takes just 46 time steps, the dark red one 116. Green and blue marks in the other curves highlight the positions after 46 and $64 \Delta t$ for comparison. For a 40% higher start velocity, violet triangles build a hyperbola.

The innermost red ellipse starts with velocity 1 again but directed 45° inwards ('NW'). Note this curve does not close – why? Here, the perigee velocity is too high for the Δt

chosen. We recommend watching the limits of such numeric models always.

For a smooth descent to a planet or one of the many moons of Jupiter or Saturn, you can introduce a decelerating force even depending on height over ground. Just your imagination is the limit – and your abilities in mathematical modeling.

Solving Differential Equations (2)

Here comes a program solving 1st- and 2nd-order differential equations by the fourth-order *Runge-Kutta* method, translated from *HP-34C Advanced Mathematical Applications*:

A first-order equation is of the form $y' = f(x, y)$, with initial values x_0, y_0 , a 2nd-order equation is of the form $y'' = g(x, y, y')$, with initial values x_0, y_0, y'_0 .

The solution is a numerical solution which calculates y_n for $x_n = x_0 + n s$ (with $n = 1, 2, 3, \dots$), where s is an incremental step specified by the user. The value for s may be changed at any time during the program's execution. This allows solution of the equation for values of x arbitrarily close to a pole ($y = \pm \infty$), or a zero ($y = 0$).

The initial values shall be provided on the *stack*, e.g., $[s, y'_0, y_0, x_0, \dots]$. Results are delivered as sequence of x , y , and – if applicable – y' values.

Equations for 1st order:

$$y_{i+1} = y_i + \frac{1}{6} (c_1 + 2c_2 + 2c_3 + c_4)$$

with $c_1 = s f(x_i, y_i)$

$$c_2 = s f\left(x_i + \frac{s}{2}, y_i + \frac{c_1}{2}\right)$$

$$c_3 = s f\left(x_i + \frac{s}{2}, y_i + \frac{c_2}{2}\right)$$

$$c_4 = s f(x_i + s, y_i + c_3)$$

```

LBL 'RKdeq1'
  REM 'Runge-Kutta for 1st order differential equations'
LBL 'Init1'
  STO 00          Begin with [s, y0, x0, ...]
  R↓
  2
  STO/ 00         R00 contains s / 2 now
  R↓
  STO 02          [y0, x0, ...]
  R↓
  STO 01          x0
  STOP
LBL 'Repet1'
  RCL 02          y
  RCL 01          [x, y, ...]
  XEQ 'funct1'
  STO 'C1'        (s/2)×f(x, y) = c1/2
  RCL 02
  +              c1/2 + y
  RCL 01          [x, c1/2 + y, ...]
  RCL 00
  +              [x + s/2, c1/2 + y, ...]

```

```

XEQ 'funct1'
STO 'C2C3'
RCL 02
+
RCL 01
RCL 00
+
XEQ 'funct1'
STO+ 'C2C3'
ENTER↑
+
RCL 02
+
RCL 01
RCL 00
ENTER↑
+
STO 's'
+
XEQ 'funct1'
RCL 'C1'
+
RCL 'C2C3'
ENTER↑
+
+
3
/
STO+ 02
RCL 's'
STO+ 01
RCL 01
STOP
RCL 02
STOP
GTO 'Repet1'

```

$$(s/2) \times f(x + s/2, c_1/2 + y) = c_2/2$$

$$c_2/2 + y$$

$$x$$

$$[x + s/2, c_2/2 + y, \dots]$$

$$(s/2) \times f(x + s/2, c_2/2 + y) = c_3/2$$

R4 contains $c_2/2 + c_3/2$ now

$$[c_3/2, c_3/2, \dots]$$

$$c_3$$

$$c_3 + y$$

$$[x, c_3 + y, \dots]$$

$$[s/2, s/2, x, c_3 + y, \dots]$$

$$[s, x, c_3 + y, \dots]$$

$$[x + s, c_3 + y, \dots]$$

$$(s/2) \times f(x + s, c_3 + y) = c_4/2$$

$$c_1/2$$

$$c_1/2 + c_4/2$$

$$[c_2/2 + c_3/2, c_1/2 + c_4/2, \dots]$$

$$[c_2 + c_3, c_1/2 + c_4/2, \dots]$$

$$[c_1/2 + c_2 + c_3 + c_4/2, \dots]$$

$$(c_1 + 2c_2 + 2c_3 + c_4)/6$$

R02 contains $y + (c_1 + 2c_2 + 2c_3 + c_4)/6$ now

$$s + x = \text{new } x$$

$$\text{new } y$$

ad infinitum

```

LBL 'funct1'
NOP
RCL 00
×
RTN
END

```

← Store your function here (→ the example below)

Multiply the function result times $s/2$

How to proceed:

1. Enter *PEM* and key in the program as listed above
2. Leave *PEM*. Press **GTO 'func1'**
3. Enter *PEM* and key in the function, replacing NOP. Remember *x* shall be in *X* and *y* in *Y* at begin.
4. Leave *PEM*. Push the initial values for *s*, *y*, and *x* on the *stack*. Then call **XEQ 'Init1'**
5. Get *x*₁, *y*₁, *x*₂, *y*₂, *x*₃, ...
(Feel free to store a new value for *s*/2 into **R00** while the program stops.)

Example:

Solve numerically the first-order differential equation

$$y' = \frac{\sin(x) + \arctan(y/x)}{y - \ln(\sqrt{x^2 + y^2})}$$

where *x*₀ = *y*₀ = 1. Let *s* = 0.5. The angular mode must be set to radians.

Keystrokes:

1. Enter *PEM* and key in the program as listed above
2. **[P/R]** leave *PEM* and **[GTO] [PROG] (func1)**
3. **[P/R]** for entering the specific differential equation



[MODE] RAD

[STO] 05 [x<=>y] [STO] 06 [x<=>y]

[→P] [ln] [RCL] 06 [-] [+/-] [1/x]

[RCL] 06 [RCL] 05 [/] [TRI] arctan

[RCL] 05 sin [+] [x]

[P/R] leave *PEM* eventually

4. **1 [ENTER↑] 1 [ENTER↑] 0.5**

[XEQ] [PROG] (Init1)

5. **1.50 [R/S] 2.06 [R/S]**

2.00 [R/S] 2.78 [R/S]

2.50 [R/S] 3.28 [R/S] ...

Equations for 2nd order:

$$y_{i+1} = y_i + s \left[y'_i + \frac{1}{6} (k_1 + k_2 + k_3) \right]$$

$$y'_{i+1} = y'_i + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4)$$

with $k_1 = s \ g(x_i, y_i, y'_i)$

$$k_2 = s \ g\left(x_i + \frac{s}{2}, y_i + \frac{s}{2} y'_i + \frac{k_1}{8}, y'_i + \frac{k_1}{2}\right)$$

$$k_3 = s \ g\left(x_i + \frac{s}{2}, y_i + \frac{s}{2} y'_i + \frac{k_2}{8}, y'_i + \frac{k_2}{2}\right)$$

$$k_4 = s \ g\left(x_i + s, y_i + s y'_i + \frac{k_3}{2}, y'_i + k_3\right)$$

LBL 'RKdeq2'

REM 'Runge-Kutta for 2nd order differential equations'

LBL 'Init2'

STO 00

Begin with [s, y'₀, y₀, x₀, ...]

R↓

2

STO/ 00

R00 contains s / 2 now

R↓

STO 03

[y'₀, y₀, x₀, ...]

R↓

STO 02

[y₀, x₀, ...]

R↓

STO 01

x₀

STOP

LBL 'Repet2'

RCL 03

y'

RCL 02

y

RCL 01

[x, y, y', ...]

XEQ 'funct2'

STO 'K1'

(s/2)×f(x, y, y') = k₁/2

STO I

RCL 00

[s/2, k₁/2, ...]

XEQ A

Decoding the rest of this program is left to the reader.

STO 04

STO I

RCL 00

RCL I

```

RCL 03
+
RCL 'K1'
XEQ B
STO+ 04
ENTER↑
+
STO I
RCL 00
ENTER↑
+
XEQ A
RCL 04
ENTER↑
+
+
RCL 'K1'
+
3
/
RCL 03
+
STO 03
RCL L
RCL 04
RCL 'K1'
+
3
/
+
RCL 00
ENTER↑
+
STO 'K1'
×
RCL 02
+
STO 02
RCL 'K1'
STO+ 01
RCL 01
STOP
RCL 02

```

new **x**

new **y**

```
STOP
GTO 'Repet2'
```

ad infinitum

```
LBL A
  RCL I
  RCL 03
  +
  RCL I
LBL B
  2
  /
  RCL 03
  +
  R↑
  ×
  RCL 02
  +
  RCL 01
  R↑
  +
LBL 'funct2'
  NOP
  RCL 00
  ×
  RTN
END
```

← Store your function here (→ the example below)

Multiply the function result times $s/2$

How to proceed:

1. Enter *PEM* and key in the program as listed above
2. Leave *PEM*. Press **GTO 'funct2'**
3. Enter *PEM* and key in the function, replacing NOP. Remember x shall be in X , y in Y , and y' in Z at begin.
4. Leave *PEM*. Push the initial values for s , y' , y , and x on the *stack*. Then call **XEQ 'Init2'**
5. Get $x_1, y_1, x_2, y_2, x_3, \dots$
(Feel free to store a new value for $s/2$ into **R00** while the program stops.)

Example:

Solve numerically the 2nd-order differential equation $(1 - x^2) y'' + x y' = x$
Where $x_0 = y_0 = y'_0 = 0$ and $s = 0.1$.

Rewrite the equation as

$$y'' = \frac{x(1-y')}{1-x^2}$$

Keystrokes:

1. Enter *PEM* and key in the program as listed above

2. **P/R** leave *PEM* and **GTO** **PR0G** (funct2)

3. **P/R** for entering the specific differential equation



STO 05 **R↓** **R↓** 1 **−** **RCL** 05 **x**

RCL 05 **EXP** **x²** 1 **−** **/**

P/R leave *PEM* eventually

4. 0 **FILL** 0.1 **DISP** **FIX** 4

XEQ **PR0G** (Init2)

5. 0.100 0 **R/S** 0.000 2 **R/S**

0.200 0 **R/S** 0.001 3 **R/S**

0.300 0 **R/S** 0.004 6 **R/S** ...

Areas, Curve Lengths, and Volumes

The following program²¹⁴ computes:

- the derivative of a given function specified in rectangular or polar coordinates ($f(x)$ or $r(\theta)$) at a specific point,
- its integral between two given points,
- the length of its curve between them,
- the volume of the *solid of revolution* enclosed by said curve between two points, and
- the surface of said solid.

The program provides specific entry points for these tasks, → below.

LBL 'ACLVol'

REM 'Areas, Curve Lengths, Volumes'

LBL 'f'

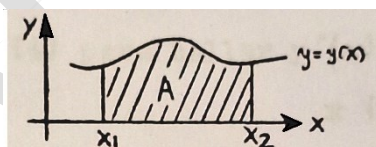
REM 'computes the integral in rectangular coordinates'

XEQ 'limits'

PGMINT 'func'

∫fd 'x'

RTN



LBL 'limits'

frequently used

STO ↑Lim

R↓

STO ↓Lim

R↑

leave the stack as it was

RTN

LBL 'curvef'

REM 'computes the curve length in rectangular coordinates'

XEQ 'limits'

PGMINT 'Lf'

∫fd 'x'

RTN

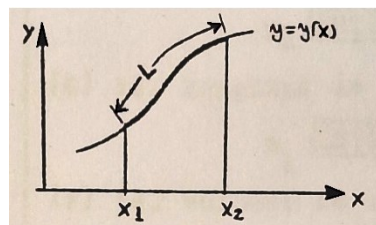
$$L = \int_a^b \sqrt{1 + f'(x)^2} dx$$

LBL 'Lf'

this is an integrand

RCL 'x'

REM 'computes $\sqrt{(1+f'^2)}$ '



²¹⁴ Taken from *HP-34C Matemática avanzada* by [Valentin Albiño](#) and [Fernando del Rey](#) (from Spain), available on [Eric's](#) website. The sketches were refined for better readability. The program is carefully modernized by using alpha labels, comments, variables, etc.

XEQ 'f'

1

→POL

RTN

LBL 'volums'

REM 'volume of the solid of revolution in rectangular coordinates'

XEQ 'limits'

π

$$V = \int_a^b \pi f^2 dx$$

GTO A

LBL 'r'

REM 'the integral in polar coordinates'

XEQ 'limits'

0.5

$$A = \int_a^b \frac{1}{2} r^2 d\theta$$

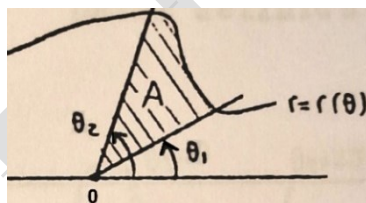
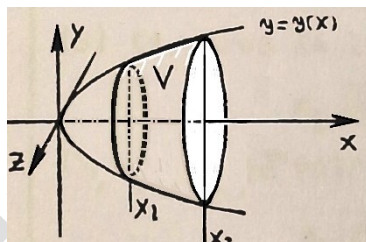
LBL A

STO I

PGMINT 'subf'

∫fd 'x'

RTN



LBL 'subf'

this is an integrand once again but x is not needed here yet

REM 'computes $\pi \cdot f^2$ or $f^2/2$ '

XEQ 'func'

x^2

RCL I

either 0.5 or π

x

RTN

LBL 'curver'

REM 'computes the curve length in polar coordinates'

XEQ 'limits'

PGMINT 'Lr'

∫fd 'x'

RTN

$$L = \int_a^b \sqrt{r^2 + r'^2} d\theta$$

LBL 'Lr'

this is an integrand again

RCL 'x'

REM 'computes $\sqrt{(r^2 + r'^2)}$ '

LocR 01

STO .00

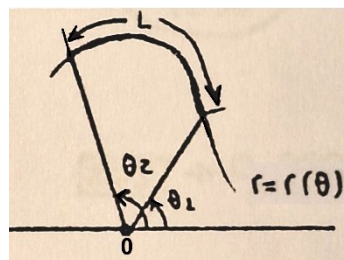
XEQ 'f'

RCL .00

XEQ 'funct'

RCL I

I was fed by subroutine f'



→POL
RTN

LBL 'surfac'

REM 'surface of the solid of revolution in rectangular coordinates'

XEQ 'limits'

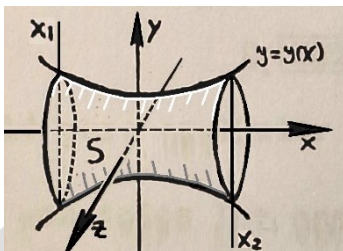
PGMINT 'surf'

fd 'x'

RTN

$$S = \int_a^b 2\pi f \sqrt{1 + f'^2} dx$$

this is an integrand again



LBL 'surf'

RCL 'x'

REM 'computes $2\pi f \sqrt{1 + f'^2}$ '

STO J

XEQ 'funct'

x₂ J

XEQ 'f''

1

→POL

RCL J

x

2

x

π

x

RTN

LBL 'f''

REM 'computes the derivative of f or r'

LocR 02

STO .00 = x

8-10-17 = δ

STO .01

STO+ .01 = 2δ

STO- .00 = x - δ

+ = x + δ

XEQ 'funct'

STO I = funct(x + δ)

RCL .00 = x - δ

XEQ 'funct'

STO- I = funct(x + δ) - funct(x - δ)

RCL .01

STO/ I $f'(x) = [f(x + \delta) - f(x - \delta)] / 2\delta$

RCL I

RTN

```

LBL 'func'
RCL 'X'
LBL 'func'
REM 'computes f(x) or r(θ) with the argument given in X'
RTN
END

```

since func may be an integrand

$f(x)$ or $r(\theta)$ with the argument given in X

insert the function $f(x)$ or $r(\theta)$ just behind the REM step.

Note this 110-step (973-bytes) program uses the *local label A*, the *global registers I* and *J*, as well as the *global variables* $\downarrow\text{Lim}$ and $\uparrow\text{Lim}$.

How to benefit from **ACLVol**:

1. Enter above program as usual (**GTO** $\square\square$, **P/R**, type, **P/R**). Having done this, do the following for using it:

2. **GTO** **PROG** [**func**]

P/R $\blacktriangledown$ and enter the function to be investigated just behind the REM step there (if there was any previous function, delete it before).

Note the argument of the function shall be given in X and its result shall be returned in X as well.

Thereafter, press **P/R** to return to *RUM*. Set **ACC** to the proper value.

3. Now there are the following options:

a. To get the value of the function at an arbitrary location, enter said location and **XEQ** **PROG** [**func**].

To get the derivative of the function at an arbitrary location, enter said location and **XEQ** **PROG** [**f'**].

b. To get the integral of the function $f(x)$ between a and b , key in a **ENTER** b **XEQ** **PROG** [**$\int f$**].

c. To get the curve length of the function $f(x)$ between a and b , key in a **ENTER** b **XEQ** **PROG** [**curvef**].

d. To get the volume of the *solid of revolution* enclosed by $f(x)$ between a and b , key in a **ENTER** b **XEQ** **PROG** [**volums**].

e. To get the surface of the *solid of revolution* enclosed by $f(x)$ between a and b , key in a **ENTER** b **XEQ** **PROG** [**surfacc**].

- f. To get the integral of the function $r(\theta)$ between α and β , key in
 α **(ENTER)** β **(XEQ)** **(PROG)** **($\int r$)**.
- g. To get the curve length of the function $r(\theta)$ between α and β , key in
 α **(ENTER)** β **(XEQ)** **(PROG)** **(curver)**.

In all cases but a, the uncertainty of the result is returned in Y.

Example 0:

Find the integral of $f = 1/\sqrt{x}$ between 0 and 1.

Solution:

With **ACLVol** keyed in,

(GTO) **(PROG)** **(funct)**

(P/R) **(∇)** **($\sqrt{x}$)** **(1/x)** **(P/R)**

0 **(ENTER)** 2 **(π)** **(x)** **(XEQ)** **(PROG)** **($\int f$)**

returns

$2. \pm 1.4 \times 10^{-35}$.

Example 1 (with **RAD** and **FIX 6**):

Find the length of the first turn of an Archimedean spiral defined by $r = 3\vartheta$.

Solution:

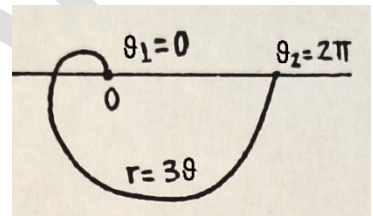
(GTO) **(PROG)** **(funct)**

(P/R) **(∇)** 3 **(x)** **(P/R)**

0 **(ENTER)** 2 **(π)** **(x)** **(XEQ)** **(PROG)** **(curver)**

returns

$63.768\ 882 \pm 1 \times 10^{-35}$.



Example 2:

Find the area of $1/4$ of the lemniscate $r = 3 \times \sqrt{\cos(2\vartheta)}$.

Solution:

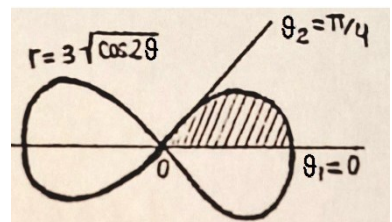
(GTO) **(PROG)** **(funct)**

(P/R) **(∇)** 2 **(x)** **(TRI)** **(cos)** **($\sqrt{x}$)** 3 **(x)** **(P/R)**

0 **(ENTER)** **(π)** 4 **(/)** **(XEQ)** **(PROG)** **($\int r$)**

returns

$2.25 \pm 4 \times 10^{-35}$.

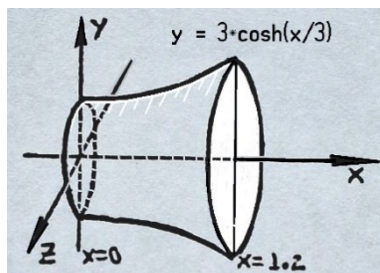


Example 3:

Find the volume of the body created by a full turn of $f = 3 \times \cosh(x/3)$ around the x -axis between $x = 0$ and 1.2 .

Solution:

GTO **PROG** [funct]
P/R **▼** 3 **/** **TRI** **cosh** 3 **x** **P/R**
0 **ENTER** **↑** 1.2
XEQ **PROG** [volums]
 returns $35.797\ 554 \pm 3 \times 10^{-35}$.

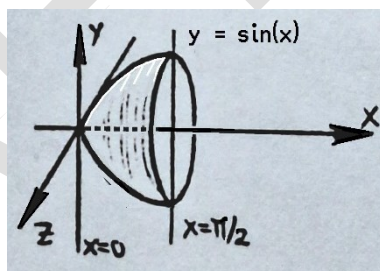


Example 4:

Find the lateral surface of the body created by a full turn of $f = \sin(x)$ around the x -axis between $x = 0$ and $\pi/2$.

Solution:

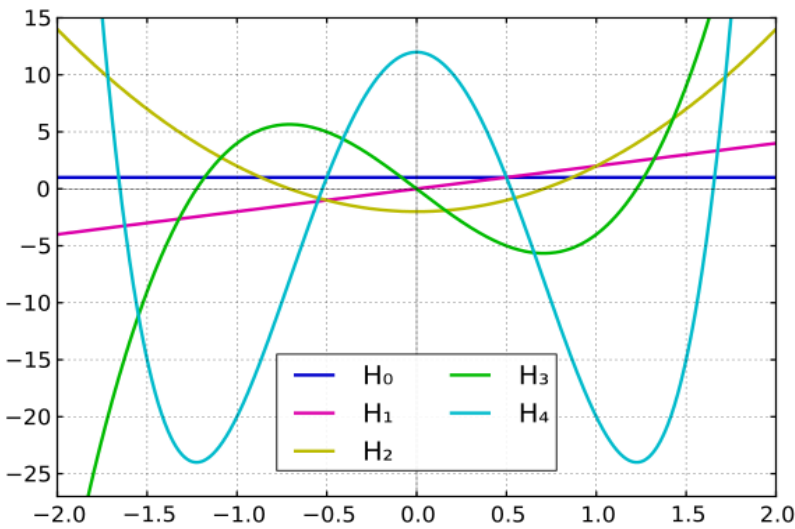
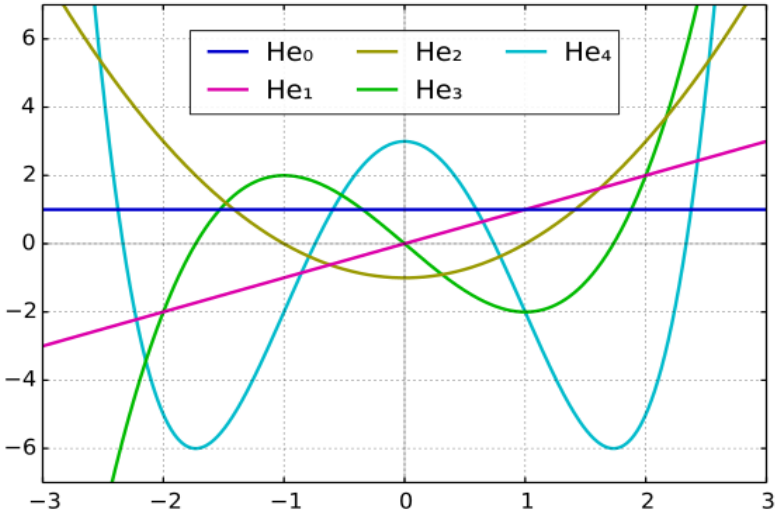
GTO **PROG** [funct]
P/R **▼** **TRI** **sin** **P/R**
0 **ENTER** **↑** **π** 2 **/**
XEQ **PROG** [surfacc]
 returns $7,211\ 800 \pm 6 \times 10^{-23}$.

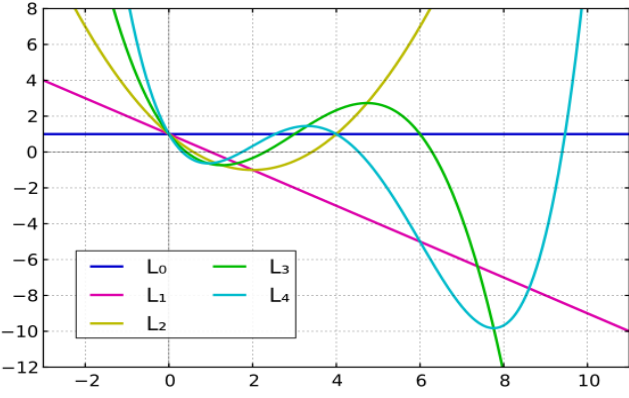
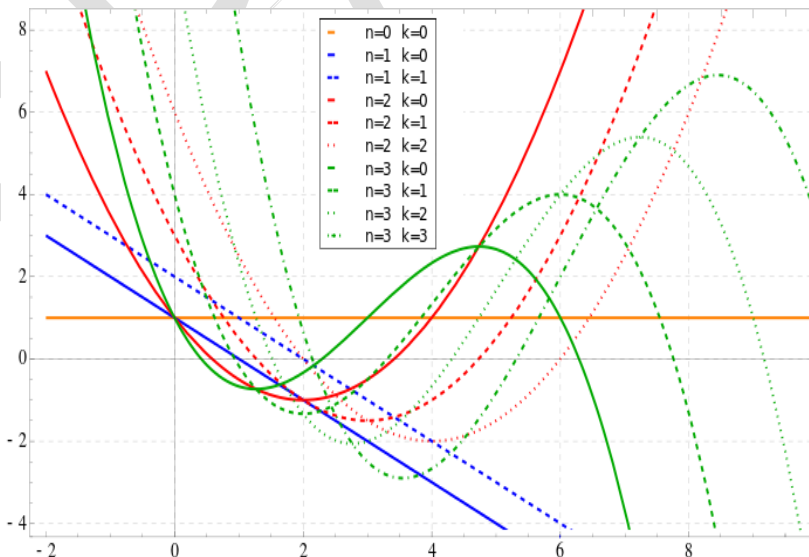


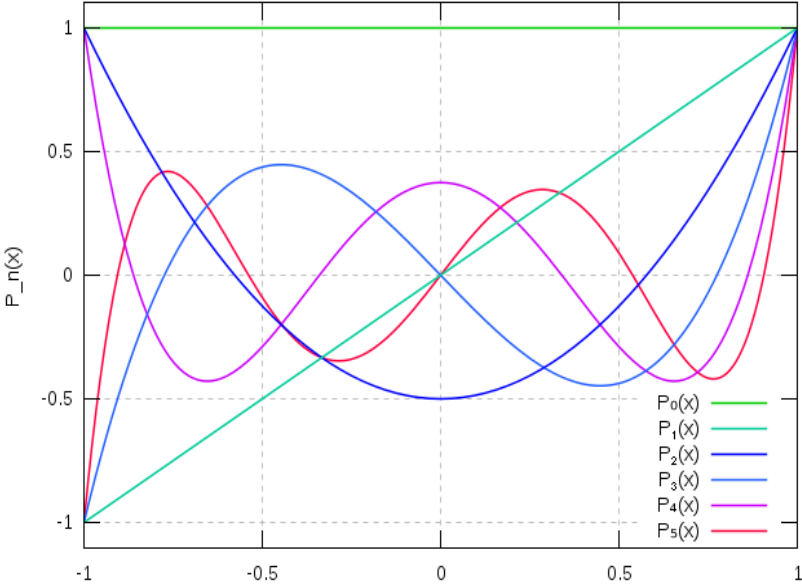
Orthogonal Polynomials

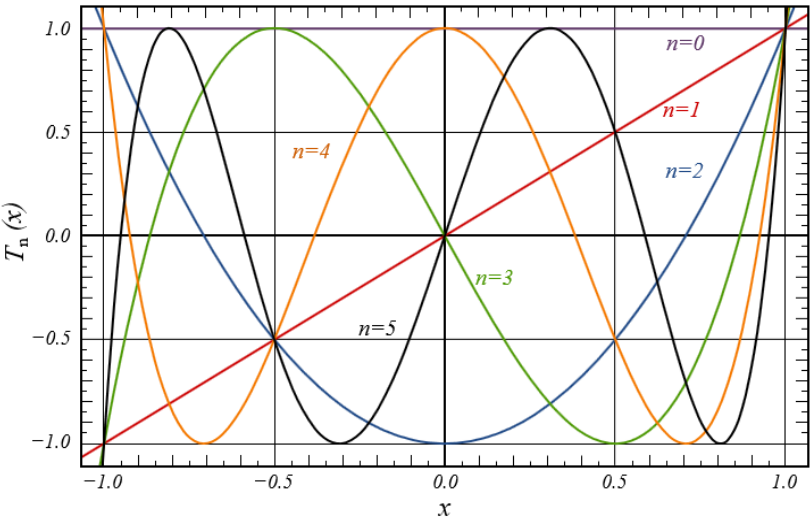
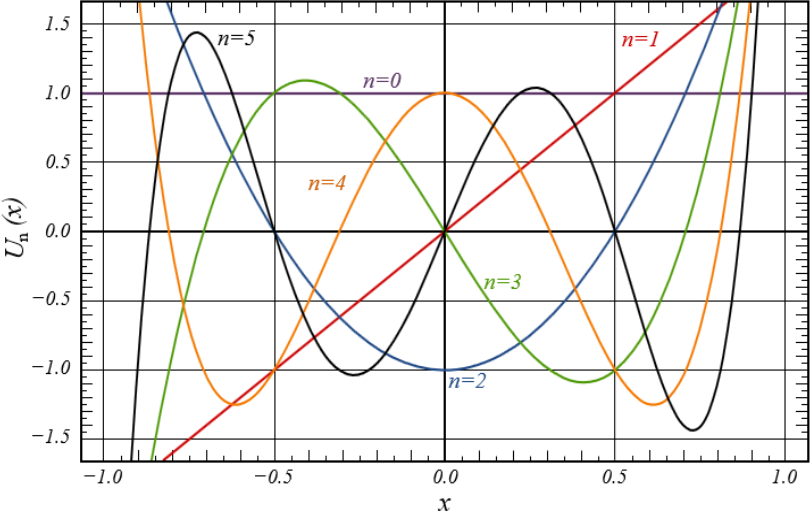
The following seven sets of polynomials are all collected in [X.FN'ORTHOG](#):

Name	Remarks (→ 14ff for general information)
H_n	<i>Hermite</i> polynomials for <u>probability</u>: $H_n(x) = (-1)^n e^{x^2/2} \frac{d^n}{dx^n} (e^{-x^2/2})$ with n in Y solve the differential equation $f''(x) - 2x \cdot f'(x) + 2n \cdot f(x) = 0.$ See the first five polynomials plotted overleaf.

Name	Remarks (→ 14ff for general information)
	
H_{np}	<u>Hermite</u> polynomials for <u>physics</u>: $H_{np}(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}(e^{-x^2})$ with n in $\mathbf{Y}$ solve the same differential equation. See the first five polynomials plotted: 

Name	Remarks (→ 14ff for general information)
L_m	Laguerre polynomials: $L_m(x) = \frac{e^x}{m!} \frac{d^m}{dx^m}(x^m e^{-x}) = L_m^{(0)}(x)$ with m in $\mathbb{Y}$ solve the differential equation $x \cdot f''(x) + (\alpha + 1 - x) \cdot f'(x) + m \cdot f(x) = 0 \quad \text{with } \alpha = 0$ (compare $L_{m\alpha}$ below). See the first five <i>Laguerre polynomials</i> plotted here. The first three you will find below again. 
$L_{m\alpha}$	Laguerre's generalized polynomials (compare L_m): $L_m^{(\alpha)}(x) = \frac{x^{-\alpha} e^x}{m!} \frac{d^m}{dx^m}(x^{m+\alpha} e^{-x})$ with m in $\mathbb{Y}$ and α in $\mathbb{Z}$. Some are plotted below ($n = m$, $k = \alpha$). 

Name	Remarks (→ 14ff for general information)
P_n	Legendre polynomials: $P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} [(x^2 - 1)^n]$ with n in $\mathbb{Y}$ solve the differential equation $\frac{d}{dx} \left[(1 - x^2) \cdot \frac{d}{dx} f(x) \right] + n(n - 1) f(x) = 0.$ The first six polynomials are plotted here: 
T_n	Čebyšëv (a.k.a. <i>Chebyshev</i>, <i>Tschebyschow</i>, <i>Tschebyscheff</i>, originally <i>Чебышёв</i>) <u>polynomials of first kind</u> $T_n(x) = \begin{cases} \cos(n \arccos(x)) & \text{for } -1 \leq x \leq 1 \\ \cosh(n \operatorname{arcosh}(x)) & \text{for } x > 1 \\ (-1)^n \cosh(n \operatorname{arcosh}(-x)) & \text{for } x < -1 \end{cases} \quad \text{with } n \text{ in } \mathbb{Y}$ solve the differential equation $f''(x) - \frac{x}{1 - x^2} f'(x) + \frac{n^2}{1 - x^2} f(x) = 0.$ The plot overleaf shows $T_0(x) \dots T_5(x)$.

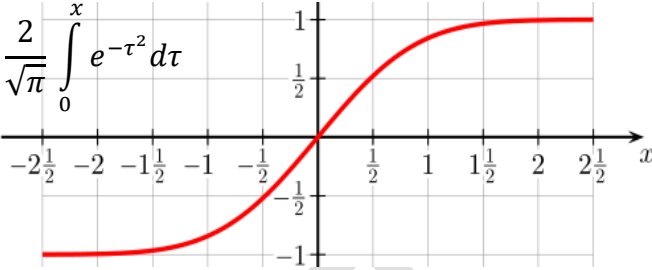
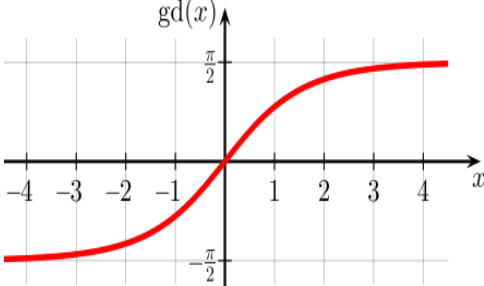
Name	Remarks (→ 14ff for general information)
	
U_n	Čebyšev polynomials of <u>second</u> kind $U_n(x)$ with n in $\mathbb{Y}$ solve the differential equation $f''(x) - \frac{3x}{1-x^2} f'(x) + \frac{n(n+2)}{1-x^2} f(x) = 0$ The plot shows $U_0(x) \dots U_5(x)$: 

Even More Mathematical Functions

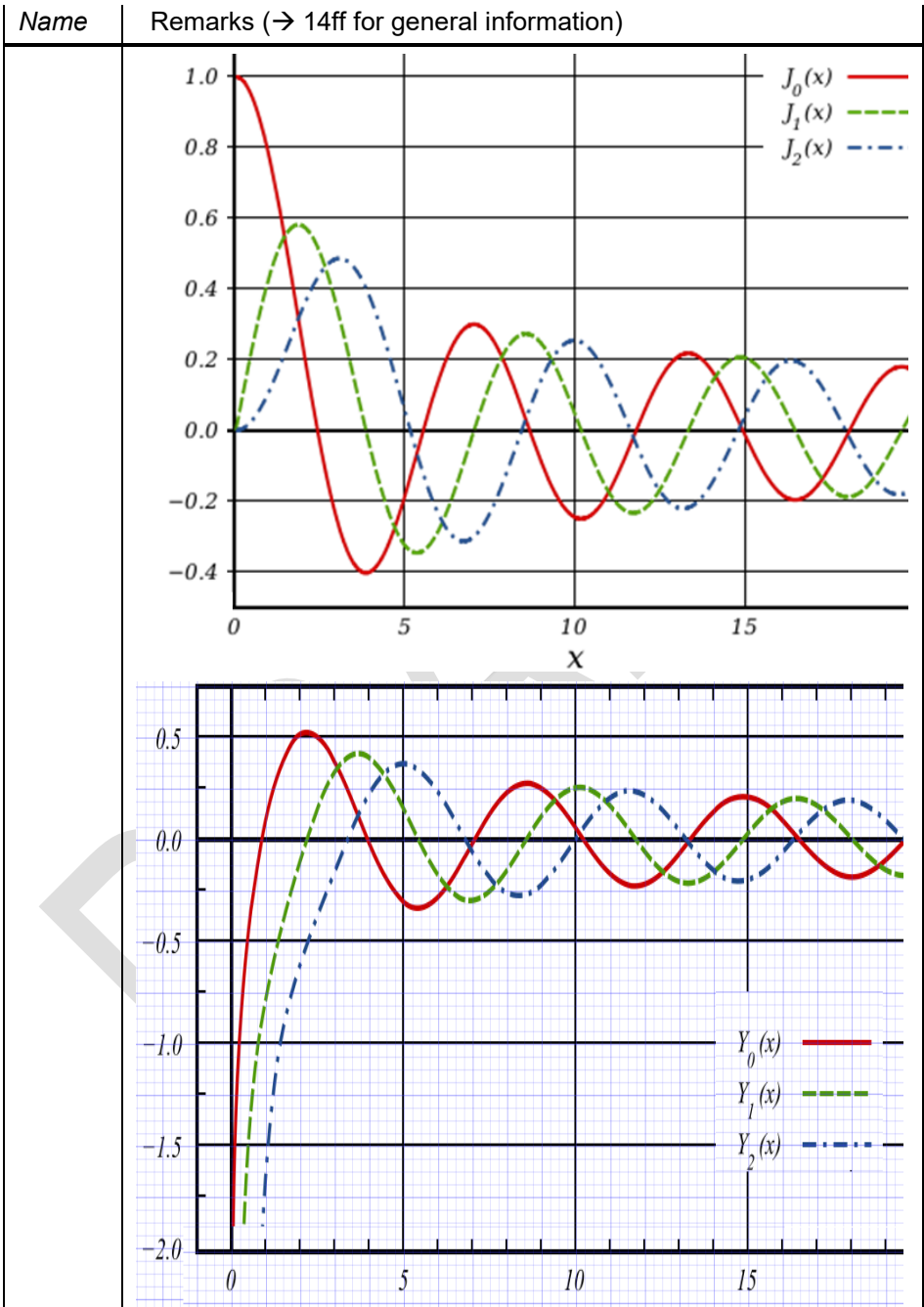
Also all the following functions are found in X.FN. Some of them are for pure mathematics only but were useful at some stages of the WP 34S or WP43 projects, so we made these functions accessible for the public.

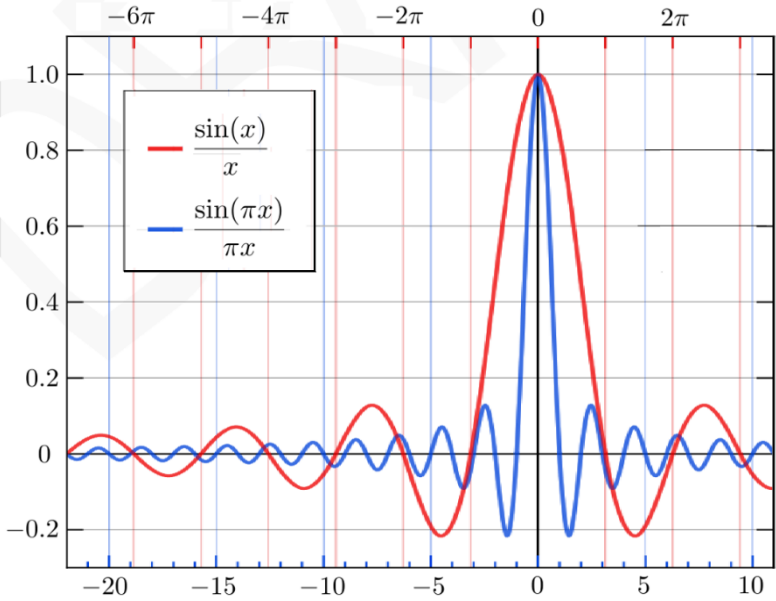
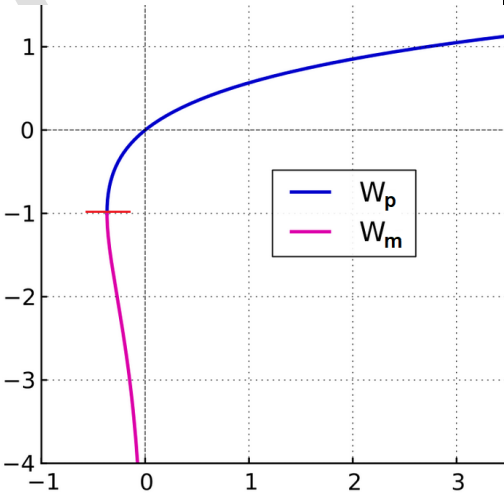
Name	Remarks (→ 14ff for general information)
AGM	Returns the <i>arithmetic-geometric mean</i> . The <i>AGM</i> is very useful in computing the values of complete elliptic integrals (→ 327) and can also be used for finding the inverse tangent. Read more about it at http://mathworld.wolfram.com/Arithmetic-GeometricMean.html .
atan2	Returns the expanded <i>arcus tangent</i> . It is explained most easily by this picture showing the returns of arctan2 for five points (x, y):

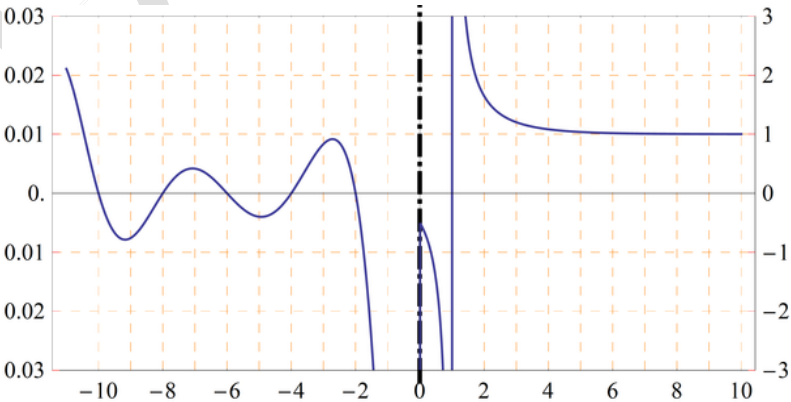
Name	Remarks (→ 14ff for general information)
$\text{cn}(u,m)$, $\text{dn}(u,m)$, $\text{sn}(u,m)$, $\psi(u,m)$, $K(m)$, $F(\varphi,m)$, $E(m)$, $E(\varphi,m)$, $\Pi(n,m)$, $Z(\varphi,m)$	$K(m) = F\left(\frac{\pi}{2}; m\right) = \int_0^{\pi/2} \frac{d\vartheta}{\sqrt{1 - m \sin^2 \vartheta}} \quad \text{with } m \in \mathbb{R}$ is called the <i>complete elliptic integral of the 1st kind</i>. The <i>incomplete elliptic integral</i> of this kind is: $F(\varphi; m) = \int_0^{\varphi} \frac{d\vartheta}{\sqrt{1 - m \sin^2 \vartheta}} = \int_0^{\tan \varphi} \frac{d\omega}{\sqrt{(1 + \omega^2)[1 + \omega^2(1 - m)]}} = u$ Using u, m, and the ‘amplitude’ ψ, the following four <i>elliptic functions</i> are defined: $\begin{aligned} \text{sn}(u; m) &= \sin(\psi), & \psi(u; m) &= \arcsin[\text{sn}(u; m)], \\ \text{cn}(u; m) &= \cos(\psi), & \text{and } \text{dn}(u; m) &= \sqrt{1 - m \sin^2 \psi} \end{aligned}$ The (<i>in-</i>) <i>complete elliptic integral of the 2nd kind</i> looks like this: $\left(E(\varphi, m) = \int_0^{\varphi} \sqrt{1 - m \sin^2 \vartheta} \, d\vartheta \right) \quad E(m) = \int_0^{\pi/2} \sqrt{1 - m \sin^2 \vartheta} \, d\vartheta$ <i>Jacobi's Zeta</i> is $Z(\varphi; m) = E(\varphi; m) - F(\varphi; m) \frac{E(m)}{K(m)}$. The <i>complete elliptic integral of the 3rd kind</i> is $\Pi(n, m) = \int_0^{\pi/2} \frac{d\vartheta}{(1 - n \sin^2 \vartheta) \sqrt{1 - m \sin^2 \vartheta}}$ You find more about the <i>elliptic functions and integrals</i> provided here at https://en.wikipedia.org/wiki/Jacobi_elliptic_functions and https://en.wikipedia.org/wiki/Elliptic_integral (the German texts at https://de.wikipedia.org/wiki/Jacobische_elliptische_Funktion and https://de.wikipedia.org/wiki/Elliptisches_Integral contain a few applications at least). Further examples of usage are printed, e.g., in https://dlmf.nist.gov/19.30 ff, https://dlmf.nist.gov/22.18 and ...19; some interesting original texts of <i>Jacobi</i> et. al. (of 1849ff) are accessible via http://sites.mathdoc.fr/cgi-bin/rbsm?cc=F_8_h%2a

Name	Remarks (→ 14ff for general information)
erf	... returns the <i>error function</i> $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\tau^2} d\tau$  Note that $\operatorname{erf}\left(x/\sqrt{2}\right) = 2 \Phi(x) - 1$ where $\Phi(x)$ represents the <i>standardized normal CDF</i> as described on p. 279. Beyond pure statistics, the <i>error function</i> may be helpful in heat conduction and diffusion problems, for instance.
erfc	... returns the <i>complementary error function</i> $\operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-\tau^2} d\tau$ This function is related to the <i>error probability</i> of the <i>standardized normal distribution</i>.
g_d, g_d^{-1}	g_d returns the <i>Gudermannian function</i> $g_d(x) = \int_0^x \frac{d\xi}{\cosh(\xi)}$ linking hyperbolic and trigonometric functions. See the plot for its <i>real</i> values. The <i>inverse of this function</i> is $g_d^{-1}(x) = \int_0^x \frac{d\xi}{\cos(\xi)}$ Start reading here for more: http://en.wikipedia.org/wiki/Gudermannian_function . 

Name	Remarks (→ 14ff for general information)
I_{xyz}	... returns the <i>regularized (incomplete) Beta function</i> $I_{xyz} = \beta_x(x, y, z) / \beta(y, z)$ 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 $\beta(y, z)$ being <i>Euler's Beta function</i> (→ 101 and https://en.wikipedia.org/wiki/Beta_function).
$J_y(x)$, $Y_y(x)$	Generally, the <i>Bessel functions</i> solve the differential equation $x^2 f''(x) + x f'(x) + (x^2 - y^2) f(x) = 0$ with $y \in \mathbb{C}$. $J_y(x)$ returns the <i>Bessel function of first kind</i> and order y. For arbitrary y, this is $J_y(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m! \Gamma(m + y + 1)} \left(\frac{x}{2}\right)^{2m+y}$ For integer y, this is also $J_y(x) = \frac{1}{\pi} \int_0^{\pi} \cos[y t - x \sin(t)] dt$ $Y_y(x)$ returns the <i>Bessel function of second kind</i> and order y. For arbitrary y, this is $Y_y(x) = \frac{J_y(x) \cos(y \pi) - J_{-y}(x)}{\sin(y \pi)}$ Watch the different vertical scales of the plots overleaf. Start reading here for more information: http://en.wikipedia.org/wiki/Bessel_function.



Name	Remarks (→ 14ff for general information)
sinc, sincπ	 The graph displays two sinc functions. The x-axis ranges from -20 to 10, with major ticks every 5 units. The y-axis ranges from -0.2 to 1.0, with major ticks every 0.2 units. The red curve represents $\frac{\sin(x)}{x}$ and the blue curve represents $\frac{\sin(\pi x)}{\pi x}$. Both curves have a maximum value of 1.0 at $x=0$. The blue curve oscillates much more rapidly than the red curve. Vertical grid lines are present at $x = -6\pi, -4\pi, -2\pi, 0, 2\pi$ (labeled at the top) and at integer values of x (labeled at the bottom).
W_p, W_m	Return Lambert's W with its principal branch (called W_p here) and its negative branch (called W_m for <u>minus</u>). The connecting point is $(x, y) = (-1/e, -1)$. The graph shows the <i>real</i> values of both branches.  The graph shows the real branches of Lambert's W function. The x-axis ranges from -1 to 3, and the y-axis ranges from -4 to 1. The blue curve represents the principal branch W_p, which starts at $(-1/e, -1)$ and increases towards $y=1$ as x increases. The magenta curve represents the negative branch W_m, which starts at $(-1/e, -1)$ and decreases towards $y=-4$ as x increases. A horizontal dashed line is drawn at $y=-1$. Start reading here for more information: http://en.wikipedia.org/wiki/Lambert_W_function. Learn more here: http://mathworld.wolfram.com/LambertW-Function.html .

Name	Remarks (→ 14ff for general information)
Γ_p	Returns the <i>regularized gamma function</i> $P(x, y) = \gamma(x, y) / \Gamma(x) \quad (\text{see } \gamma_{XY})$
Γ_q	Returns the <i>regularized gamma function</i> $Q(x, y) = \Gamma_u(x, y) / \Gamma(x) \quad (\text{see } \Gamma_{XY})$
γ_{XY}	Returns the <u>lower incomplete gamma function</u> required for Γ_p above: $\gamma(x, y) = \int_0^y t^{x-1} e^{-t} dt$
Γ_{XY}	Returns the <u>upper incomplete gamma function</u> required for Γ_q above: $\Gamma_u(x, y) = \int_y^\infty t^{x-1} e^{-t} dt$
$\zeta(x)$	Returns <i>Riemann's Zeta</i> for real x , with $\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x} \quad \text{for } x > 1,$ and its analytical continuation for $x < 1$: $\zeta(x) = 2^x \pi^{x-1} \sin\left(\frac{\pi}{2} x\right) \Gamma(1-x) \zeta(1-x).$ 

→ 101 for $\Gamma(x)$ and the following for more:
https://en.wikipedia.org/wiki/Incomplete_gamma_function

Name	Remarks (→ 14ff for general information)
	Watch the different vertical scales for the negative and positive x in this plot! And note $\sqrt{6} \zeta(2) = \pi$. → http://mathworld.wolfram.com/RiemannZetaFunction.html.

Note that the *incomplete gamma* and *error functions*, the *Laguerre*, *Legendre*, and *Bessel functions*, as well as some matrix operations were provided on the Commodore²¹⁵ M55 (*The Mathematician*, a pocket calculator with 55 keys as pictured below) in 1976 already. At the same time, the S61 (*The Statistician* with 60 keys!) featured the *binomial*, *hypergeometric*, *Poisson*, *chi-square*, *Gaussian*, *Student's t*, and *Fisher's F* distributions as well as the *chi-square test for fits* (→ [OMS 2](#)). The latter calculator was accompanied by the N60 (*The Navigator* with 60 keys as well), dedicated to flight navigation.



²¹⁵ This is the company which produced the C64 as well.

Taking into account that no *HP* pocket calculator ever provided orthogonal polynomials nor the *hypergeometric distribution* until today, *Commodore* offered an impressive early algebraic triple for some 100 US\$ each in 1976/7 (while you had to spend 125 US\$ for an *HP-21* featuring far less functions, → 264).

At the other end of the quality scale, see one of the worst *RPN* implementations pictured next to it. Admitted, it was in 1974 and called for only 49.95 £ (some 80 US\$ – while *HP*'s calculators did cost a fortune, in particular beyond the *USA*), but nevertheless precision, power, and memory of *Sir Sinclair's* calculator left a lot of space for improvement (featuring merely a 2LS!); though it was definitely the smallest pocket 'scientific' of its time. So it represented nothing more than a relatively cheap replacement of a very good slide rule (showing comparable precision) before prices of better scientific calculators plummeted.

Another less expensive calculator of that time was the *Novus Mathematician* featuring a 3LS (→ 259) – calling for some 70 US\$ in '76.

Concerning the special functions implemented in your *WP43*, you will find lots of information in the internet beyond what is printed in this appendix. 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 e-Handbook of Statistical Methods* (→ 273ff) is a competent source. And for pure mathematics, the *NIST Digital Library of Mathematical Functions* and *Mathworld* (→ 326f for both) may tell you more details than you ever wanted to know.



For the functions of *WP43*, we used the *GNU multiple precision arithmetic library* (→ <https://gmplib.org/manual/>). The *WP43* firmware has been written in *C* by various authors (→ introduction of the *OM*) and has grown for years. Major parts of its mathematical algorithms date back to the *WP 34S* project (→ 353) and were coded by *Paul Dale*. Feel free to browse the firmware files at <https://gitlab.com/rpncalculators/wp43/-/tree/master>. You might remember *Dante* though.

APPENDIX J: 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 – feel free to use STOSTK for saving it before you call a subroutine and RCLSTK for recovering it thereafter.

Below is a recursive implementation of the factorial. This is an **example** for demonstration purposes only, since this routine will neither set the *stack* correctly nor will it work at all for input greater than some 100 but you can get the principle:

```
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**.

A Few Words About Factorizing

In a nutshell, FACTOR uses the prime sieve of [Eratosthenes](#) (some 200 B.C., though the algorithm is much older). Thus, it starts with 2 and continues with 3, 5, 7, 11, 13, 17, 19, 23, etc. to check if the source number is a factor of the check prime number. This is it – there are no shortcuts, no stored list of prime numbers, nor any other tricks. With a huge source number being a product of two big prime factors, it may take very long to find them (advantage in cryptography).

For quick estimates of the time required for factorizing, the *WP43 Simulator* on my laptop needed ca. 90 minutes to check prime factors up to 10^9 in 2025-05; i.e. it would need some 900 minutes (15 hours) to check prime factors up to 10^{10} etc. – your computer may be a lot faster, but your calculator is way slower. Feel free to press **EXIT** for aborting FACTOR.

A Few Words About the Solver

In a nutshell, the *Solver* takes the start values for the unknown for calculating the function to be solved, and proceeds as outlined in the *HP-42S OM* on pp. 186f.

In most situations, the calculated root is an accurate estimate of the theoretical, infinitely precise root of the function. An ideal solution is one for which $f(x)$ exactly equals 0. However, a nonzero value for $f(x)$ is often also acceptable, because it results from approximating the root with limited (...) precision.²¹⁶

While solving, as soon as SOLVE has found two results $f(x_1)$ and $f(x_2)$ of opposite signs, it divides the interval $|x_2 - x_1|$ in two halves and looks in which of them the sign change occurs. You can observe this process ‘live’ on the screen of your *WP43*:

6.098 349 456 332 522 567 791 462 547 182 976×10 ⁻¹	+
6.098 349 456 332 522 571 235 202 903 916 494×10 ⁻¹	-

SOLVE will stop when $|f(x_{root})| \leq u$ with u being the least displayable number within the screen format selected.²¹⁷ Thus, for SCI 15 solving may end with a solution like:

```
2023-05-21 16:40 R/L x^r /max 64:2 4
0.000 000 000 000 000
3.993 488 263 668 089×10-16
6.098 349 456 332 523×10-1
r = 6.098 349 456 332 52×10-1
```

With 34 significant digits, the difference between the last two approximations to the location of the root turns out being 5×10^{-33} here.

²¹⁶ Quoted from *HP-42S PET*, p. 108.

²¹⁷ This applies if **SOLVE** returns claiming ‘**root found**’. It does not for poles, constant curve shapes, local or asymptotic extrema (→ *HP-42S* manuals).

Feel free to look up [Section 12](#) of the [HP-42S OM](#). See the [HP-34C OH](#) ([Section 8](#) and [App. A](#)) or the [HP-15C OH](#) ([Section 13](#) and [App. D](#)) for more information and some caveats.²¹⁸ Additionally, finding roots at the limits of function domain may be a challenge.

More Than a Few Words About the Integrator

If you know the analytical solution of an integral, you should use this knowledge. When an integral turns out being difficult to evaluate analytically, however, it may be easier solving it numerically.

Remember $\left[\int \right]$ and its siblings return the approximation for the integral in **X** and an estimated upper bound for its uncertainty in **Y**.

We recommend looking at the [HP-34C OH](#), [Section 9](#) and [App. B](#), of 1979 when the *Integrator* was first featured in a pocket calculator.²¹⁹ The [HP-15C CE OH](#), [Section 14](#) and [App. E](#), and in particular the [HP-15C CE Advanced Functions Handbook](#) of 2023, [Section 2](#), pp. 45 - 68, may be even better reads, also about difficult integrals. This topic is also covered in the [HP-42S OM](#), [Section 13](#), and its [PET](#), [Section 4](#). Note, however, that *HP*'s integration algorithm differs from ours.

WP43 uses the *tanh-sinh quadrature* integration algorithm published by [Michalski](#) and [Mosig](#), actually its version written for the *WP 34S* by [César Rodríguez](#) (→ <https://newtonexcelbach.com/2020/10/29/numerical-integration-with-tanh-sinh-quadrature-v-5-0/>). This is a very capable algorithm. Nevertheless, functions containing poles, narrow peaks or drops, and/or rapid oscillations in a wider range within the integration interval may carry it beyond its limitations.

We recommend checking and analyzing the integrand and applying suitable methods, e.g., subdividing the integration interval or transforming variables.

²¹⁸ Solving as described above was introduced with the [HP-34C](#) by [William Kahan](#) consulting *HP*.

²¹⁹ → *Handheld Calculator Evaluates Integrals* by [William Kahan](#) in the *HP Journal* of 1980-08 (<http://hparchive.com/Journals/HPJ-1980-08.pdf>), pp. 23 - 32. A highly recommended introduction by a mathematician!

There is a recent survey and comparison of various numeric integration methods in https://axiomelab.gitlab.io/hp-integral/article_hp_en.html. In it, the author implemented six classical numerical integration methods in C++ and subjected each to five carefully chosen test functions (→ also 2 pages below). Thanks to [Steve Simpkins](#) for pointing to this article.

Example 1:

$$\int_0^{1000} e^{\sin(x) \cdot \cos(x)} dx = ?$$

Solution (with ACC = 10^{-33}):

a) Let your WP43 integrate straight ahead. The result will be 1 038.923 with $y = 0.016$.

b) Note this integrand is oscillating a lot – but luckily it is periodic. Thus,

i. Let your WP43 compute one period $\int_0^{2\pi} e^{\sin(x) \cdot \cos(x)} dx$. It will return 6.682... with $y = 1.47 \times 10^{-35}$.

ii. Divide the integration interval by 2π . This returns 159.15... Hence multiply the result i times 159.

iii. Let your WP43 compute the integral remaining at the upper end of the integration interval $\int_{318\pi}^{1000} e^{\sin(x) \cdot \cos(x)} dx$. It will return 1.398... with $y = 8.91 \times 10^{-34}$.

iv. Add the results ii and iii, returning

1063.846 894 403 590 671 456 757 649 301 007

with a total $y = 3.24 \times 10^{-33}$.

Note that the result a) deviates far more from the truth than the estimated error returned in y there.

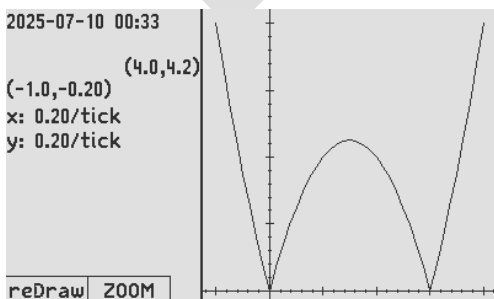
Example 2:

$$\int_{-15}^{+15} |(x-3)x| dx = ?$$

Solution (with ACC = 0):

a) Let your WP43 integrate straight ahead. The result will be 2 258.993 38... with $y = 1.4 \times 10^{-5}$ (the crucial interval $[0; 3]$ is not seen automatically).

b) DRAW the integrand (the plot here shows it between $x = -1$ and $+4$ – its bump is hardly seen when



drawing it between -15 and +15). It has two roots, 0 and 3. Due to the $|\dots|$ function, the integrand has sharp corners at these locations – many automatic algorithms do not like sharp corners. So, let's split the integral in three intervals:

- i. $\int_{-15}^0 |(x-3)x| dx = 1462.5$ with $y = 5 \times 10^{-36}$
- ii. $\int_0^3 |(x-3)x| dx = 4.5$ with $y = 3 \times 10^{-36}$
- iii. $\int_3^{15} |(x-3)x| dx = 792.$ with $y = 2.1 \times 10^{-35}$
- iv. Add the results i – iii, returning $2259. \pm 2.2 \times 10^{-35}$, i.e. **2259** exactly.

c) This integral is easily evaluated analytically, on the other hand:

$$\int (x^2 - 3x) dx = \frac{x^3}{3} - \frac{3}{2} x^2$$

$$\text{i. } \Rightarrow \int_{-15}^0 \dots = -\left[\frac{1}{3} (-15)^3 - \frac{3}{2} (-15)^2\right] = 1462.5$$

$$\text{ii. } \int_0^3 \dots = \frac{1}{3} 3^3 - \frac{3}{2} 3^2 = -4.5$$

$$\text{iii. } \int_3^{15} \dots = \frac{1}{3} 15^3 - \frac{3}{2} (15)^2 - (-4.5) = 792$$

$$\text{iv. } \Rightarrow \int_{-15}^{15} |(x-3)x| dx = 1462.5 + 4.5 + 792 = \mathbf{2259}$$

So we know the numeric *Integrator* works correctly here for the three intervals. Note it was deceived, however, by the 'pathetic' function when trying to integrate in one step – again, the result **a)** deviates far more from the truth **c)** than the estimated error returned in **y** there. Beware!

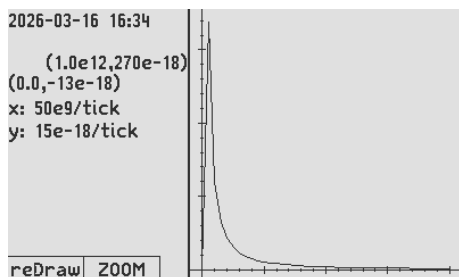
Example 3:

$$2 \int_0^{\infty} \frac{\sqrt{x}}{x^2 + 4} dx = ?$$

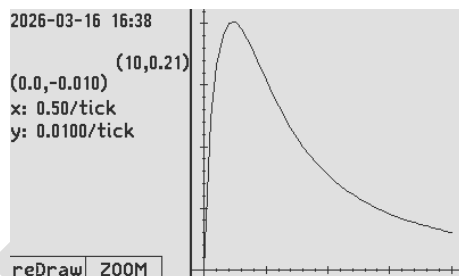
Solution (with ACC = 0):

Integrating straight ahead up to 10^{12} returns 3.141 588 653 58... with

$y = 2.5 \times 10^{-10}$. The expected result is π , so the result is exactly 4×10^{-6} off, far further than y indicates. What happened?



Let's look at the integrand – seems it shows a peak close to zero (note the plot is based on 41 calculated points only).



So let's draw it again but just between 0 and 10 →

Aha, here we are!

The main contribution to this integral comes from the part at low x -values. Thus, let's integrate in steps:²²⁰

- a) $2 \int_0^{10} \frac{\sqrt{x}}{x^2+4} dx = 1.886\ 582\ 047\ 698\ 276 \dots$ with $y = 1.5 \times 10^{-35}$
- b) $2 \int_{10}^{100} \dots dx = 0.855\ 042\ 598\ 782\ 374 \dots$ with $y = 1.6 \times 10^{-35}$
- c) $2 \int_{100}^{10^4} \dots dx = 0.359\ 968\ 007\ 429\ 142 \dots$ with $y = 8.5 \times 10^{-35}$
- d) $2 \int_{10^4}^{10^6} \dots dx = 0.035\ 999\ 999\ 680\ 003 \dots$ with $y = 1.07 \times 10^{-34}$
- e) $2 \int_{10^6}^{10^8} \dots dx = 0.003\ 599\ 999\ 999\ 996 \dots$ with $y = 1.04 \times 10^{-34}$
- f) $2 \int_{10^8}^{10^{10}} \dots dx = 0.000\ 36$ with $y = 9.8 \times 10^{-35}$
- g) $2 \int_{10^{10}}^{10^{12}} \dots dx = 0.000\ 036$ with $y = 1.17 \times 10^{-34}$

$$\begin{aligned} \Rightarrow 2 \int_0^{\infty} \frac{\sqrt{x}}{x^2+4} dx &= \sum_a^e I_i + 0.000\ 36 \sum_0^{\infty} \frac{1}{10^i} \\ &= 3.141\ 192\ 653\ 589\ 793 \dots + 0.000\ 36 \times 10/9 \\ &= 3.141\ 592\ 653\ 589\ 793 \dots = \pi \text{ exactly, q. e. d. }^{221} \end{aligned}$$

²²⁰ Thanks to [Jean-François Garnier](#) for pointing to this method.

²²¹ Integrating over infinite intervals can't be solved in such an easy way always. Look up http://fmnt.info/blog/20180818_infinite-integrals.html for another method – thanks to [Albert Chan](#) for bringing this to my attention.

There are seven finite integrals recommended by [Valentin Albillo](#) just for testing the capabilities of numeric integration algorithms. These read:

Integral	<i>Valentin's</i> result / <i>WP43</i> result with ACC = 0
11. $\int_0^1 \cos(x) \ln(x) dx$	-0.946 083 070 367 183 014 941 353 313 823 1... - 0.946 083 070 367 183 014 941 353 313 823 179 6 with $y = 2 \times 10^{-35}$
12. $\int_0^1 \frac{1}{\sqrt{x}} dx$	2 2. exactly, with $y = 1 \times 10^{-35}$
13. $\int_0^1 \ln(\Gamma(x)) dx$	$\ln(\sqrt{2\pi}) =$ 0.918 938 533 204 672 741 780 329 736 405 617 5 0.918 938 533 204 672 741 780 329 736 405 617 6 with $y = 1 \times 10^{-35}$
14. $\int_0^1 \frac{1}{\sqrt{-\ln(x)}} dx$ ²²²	$\sqrt{\pi} = 1.772 453 850 905 516 027 298 167 4...$ 1.772 453 850 905 516 013 219 ... with $y = 3 \times 10^{-19}$
15. $\int_0^1 x^{-0.99} dx$	100 99.998 345 660... with $y = 7.1 \times 10^{-7}$ ²²³
16. $\int_{-1}^1 e^{x + \sin\left(e^{e^{e^{x + \frac{1}{3}}}}\right)} dx$	2.996 339 811 976 535 379 182 861 556 235 501 887 668 096 456 254 829 159 48... 2.969 857 ... with $y = 0.019$ ²²³
17. $\int_0^1 \sin^2\{\tan[\tan(\pi x)]\} dx$	$[1 - e^{-2 \tanh(1)}]/2 = 0.390 992 162 151 53...$ 0.398 ... with $y = 0.06$

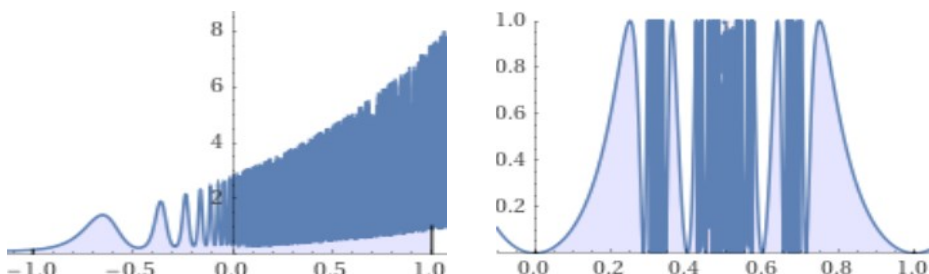
Computing these integrals on your *WP43* may take some time – *USB* power supply is recommended; then you will be done in less than two minutes. Above results are just for your information. Note **15** is easily solved analytically but its numeric computation is hampered by the pole at $x = 0$.

Look at the plots of the last two integrands below and you see why computing

²²² *SPCRES* shall be set here to avoid overflow errors. Note the integrand shows a narrow peak near $x = 1$.

²²³ Note the true value is not covered by y here.

I6 and I7 is so difficult – both are rapidly oscillating but not periodic in a useful way. Luckily, integrands like these are far from usual in real-world problems.



At the bottom line, we recommend taking good care by analysing each integrand before throwing it to the *Integrator*. It may be deceived easily by particular ‘pathetic’ features of the integrand. No automatic on earth can solve every integral for you correctly – you have to check in advance and set up the problem properly. Draw what can be drawn, your *WP43* will support you. Circumvent possible pitfalls by applying suitable methods.

And check automatic results for their plausibility as well, e.g., by dividing intervals. Like other automatisms provided on your *WP43*, also numeric integration will be useful and beneficial based on your solid understanding and experience only.

Differentiating Real Functions Using Complex Numbers

The complex-step derivative formula²²⁴ tells $f'(x) \approx \text{Im}[f(x + i h)]/h$ for a small h . Thus, we can use this method for differentiating and looking for extrema ($f'(x) = 0$) of a function.

Example:

Differentiate $f(x) = [\sin(x)]^{e^x}$

Solution:

Let's write a little program:

²²⁴ → prerequisites in https://en.wikipedia.org/wiki/Numerical_differentiation#Complex-variable_methods. Thanks to [Thomas Klemm](#) for pointing me to this.

```

LBL 'dF'
  MVAR 'x'
  MVAR 'h'
  RCL 'x'
  RCL 'h'
  RE→CX
  XEQ 'fx'
  CX→RE
END

```

compose $x + i h$.

→ below for **fx**

decompose the result, returning the imaginary part in x .

```

LBL 'fx'
  REM 'All functions within this routine operate on complex numbers.'
  sin
  RCL L
  ex
  yx
END

```

Starting with **SDC**, enter

MODE **RAD**

DISP **FIX** **5**

ADV **SOLVE** **PROG** **('dF')**

(E) **6** **÷** **h** **14** **x** **14.25** **x** **x** returns **x= 14.137 17** for the location of the extremum in this interval.

Executing 'fx' three times you will find out if that is a maximum or minimum.

A Few Words About Random Numbers

No fixed algorithm can generate true random numbers. What can be created are pseudo random numbers (*PRN*), i.e. typically *real numbers* equally distributed in a fix range which look like random. Once started, the sequence of these numbers will only repeat after a very long interval, i.e. after very many calls of the *PRN* generator.

The algorithm for **RAN#** on your WP43 starts either

1. from **SDC** (e.g., after **RESET**) or
2. with an initial 'seed' number $s \in [0; 1)$ you enter explicitly via **s SEED** or
3. with a seed taken from the calculator clock at execution time of **0 SEED**.

For equal seeds, the following sequence of *PRN* is fully determined. So you will get repeatable sequences of *PRN* for options 1 and 2 above – advantageous for debugging. If you want to get more random sequences of *PRN*, however, choose option 3.

A Few Words About Uncertainty, Significance, and Confidence

There are two ways to look at this:

1. **Multiples of σ (sigma) are the agreed way of indicating uncertainties in science.** 1σ is generally used from physics to psychology, assuming the underlying data are distributed normally (a.k.a. *Gaussian*).²²⁵ Yes, this corresponds to only 68% confidence, but as long as you know 1σ , you can determine 2 or 3σ most easily if you want or need them, corresponding to some 95% or 99% confidence. Physicists at *CERN* require 5σ or even more before publishing the discovery of a new particle, but that's extreme.
2. **95%, 99%, and 99.9% are common confidence limits or significance levels in statistics**, also applied to testing of hypotheses. Which level is called *significant* may depend on country, discipline, or industry. The examples in *HP's* manuals took 95%, so that seems to be (or have been) the standard in the *USA*. Automotive industry was also known to work with 95%.²²⁶ Looking at *DQS* (the leading German quality institution, known in the *USA* as well), 99% are the level for significance, and 99.9% for high significance. The remainder to 100%, i.e. 5%, 1%, or 0.1%, are the respective error probabilities (you find examples in [OMS 2](#)).

A Few Words About Curve Fitting

The world is full of jokes about curve fitting since there is a whole lot of mistakes one can make (→ [App. I](#)) – one of the most popular is taking more parameters than the underlying data points support. There was a saying: *Using four parameters, you can fit an elephant*; but, of course, the underlying hypotheses or theories determine the number of parameters needed – just don't use more.

²²⁵ This assumption should be checked and verified – but this is 'forgotten' frequently. Then, however, the consequences (a.k.a. results) miss a solid foundation at least or may be even simply wrong (→ [OMS 2](#)).

²²⁶ I skip some barbed remarks here. This industry suffers enough.

Fitting makes most sense based on many data points. If these are acquired experimentally, their errors should be known, usually plotted as error bars – exceeding the sensible capabilities of *WP43*. Nevertheless note that if your error bars represent $\pm s_m$ (as usual) then a ‘good’ fit curve will go through $2/3$ of all your error bars ‘only’ (compare previous chapter), and the other bars should scatter symmetrically around the curve. If your model curve hits significantly more than $2/3$ then it is either too complex for your data or you overestimated the experimental errors – if it hits significantly less then it is either too simple or you underestimated the errors. Thus, a fit curve going through all your error bars shall be eyed.

A Few Words About TVM

In business life, the *HP-12C* is the gold standard for computing financial transactions, being on the market for over 40 years so far (it is so popular in those quarters that even several clones are manufactured and sold by other companies).. The *TVM* user interface of *HP-12C* slightly deviates from the one of your *WP43* since we wanted to spell out the meanings of those equation parameters we found ambiguous. See the following **example** of an annuity due (20 000 US\$ for 4 years and 3%/a, payments at the end of each month):

<i>HP-12C</i>	<i>WP43</i>	Remarks
f 2	DISP FIX 2 FIN 12 per/a	Standard for financial problems Almost standard
CLEAR FIN END	(FIN) End	Clearing is dispensable on <i>WP43</i>
48 n 3 12÷	48 n_{PER} 3 i%/a	Total number of payment periods In offers, the <u>annual</u> interest rate is generally specified. The <i>HP-12C</i> , however, needs the interest per payment period (<u>don't</u> press i in this example there!).
2 EEX 4 PV 0 FV PMT returns - 442.69	2 E 4 PV 0 FV PMT returns PMT = -442.69	<i>TVM</i> automatically solves for the <i>variable</i> called without any numerical input heading.
17 keystrokes	14 - 16 keystrokes	

Generally, the *TVM* equation

$$0 = PV + (1 + i p) \frac{PMT}{i} [1 - (1 + i)^{-n_{PER}}] + FV (1 + i)^{-n_{PER}}$$

is solved for one of its *variables* (the unknown) based on knowing all its other parameters (→ [OMS 6](#)). *TVM* actually uses the *Solver* of your *WP43* for this. Note all *TVM* parameters are kept by your *WP43* until they are overwritten either by input or solving.

Translation of *WP43* Characters Deviating from *UTF-8*

For coding and writing *WP43* programs in human-readable text files, `.rtf` files could be used, just containing one line of text per program step. Each line may encompass a mix of

- plain text,
- *italic* text,
- superscripts and subscripts, optionally *italic*,

and shall be closed by CR and LF.

Any *rich text* editor will do (e.g., *WordPad*) as long as all *WP43 characters* are printable using common fonts. This holds for all of them complying with *UTF-8*. For the *characters* deviating, the following table lists alternative *characters* (altern.) to be used for entering these:²²⁷

<i>WP</i>	<i>UTF</i>	altern.	<i>WP</i>	<i>UTF</i>	altern.	<i>WP</i>	<i>UTF</i>	altern.
̅	00a9					ȳ	0232	→ below
̄	0378	<i>m</i> (mean)	̂	0379	<i>f</i> (forecast)	̄	037f	
1	2027		-1	2072	/	E	2073	E ²²⁸

²²⁷ Some reasoning is given in parentheses. – A few *WP43 characters* required for spelling *items* appear in **dual combinations** only, and for some of them the alternatives are just **dual combinations** as well.

The colors emphasize **superscript** and **subscript** letters. *WP43 characters* printed in light grey fields appear in *menus* only (neither in command *names* nor constants), those in light blue fields are just used on the keyboard, those in grey fields are exclusively for system output, those in dark grey fields are designed but not used presently yet.

WP	UTF	altern.		WP	UTF	altern.		WP	UTF	altern.
*	208f	*		∞	209e	00		∞	209f	00 ²²⁹
e	2147	Call from CONST		i	2148	Call from CONST				
0	2160	0	...	9	2169	9		+	216a	+
-	216b	-		→	21c0	→ ²³⁰				
Δ	21c8			▽	21ca			≡	2263	
↶	21b6			▫	220e					
√x	221a	2r (√)		∛x	221b	3r (∛)		√y	221c	xr (√)
⊕	2295	E (Earth)								
α	2296	α or a		δ	2297	δ or d		μ	2298	μ or m
☉	229a	S (Sun)		☼	229b					
10	23e8			⌘	2427					
🔋	242a	b (battery)		⚡	242b			U	242c	U
UK	242d	k		US	242e	s		Ⓢ	242f	!
↶	2430	p		↷	2431	→ left				
↶	2432	q		↷	2433	→ left		↻	2434	
1	2460	1	...	9	2468	9				
10	2469		...	16	246f					
a	2482	a	...	z	249b	z				

²²⁸ This is for entering exponents. To be valid, either a digit or a blank must precede this **E**, and it shall be directly followed by either a digit or a minus sign (-) and a digit.

²²⁹ Feel free to type italic **00** for the regular *character* ∞.

²³⁰ This alternative can be used for entering a common right arrow as well – it will be translated in the arrow suiting the target.

WP	UTF	altern.		WP	UTF	altern.		WP	UTF	altern.
a	249c	a	...	z	24b5	z				
À	24b6	A	...	Z	24cf	Z				
Ä	24d0	A	...	Z	24e9	Z				

The following *characters* are not allowed in any file names:

blank space, ", &, *, comma, point, /, :, <, >, ?, \, and |.

The following *characters* must not appear in any user-defined variables, *global labels*, or *menu names* (→ 343):

blank space, !, (,), +, comma, -, /, :, ;, =, ^, |, and x.

((Italic digits and letters used so far for translation: *0, 2, 3, a, b, d, E, f, k, m, m, p, q, r, S, s, U, x, and !.*))

Where to Find Programs Readily Available?

At <https://technical.swissmicros.com/dm42/fat/PROGRAMS/>, SM offers a collection of programs for their DM42. Although WP43 is not code compatible to DM42, these routines may be helpful since the keystroke programming paradigm is the same.

Furthermore, there are sections in both <https://forum.swissmicros.com/index.php> and <https://www.hpmuseum.org/forum/index.php> where user programs for the keystroke programmables of HP, DM, and WP calculators can be found, copied, and adapted, if applicable.

Where to Find More about WP43, its Ancestors and Offsprings?

For the predecessors, please look up HP's pocket calculators until the *HP-42S* (information about them is found at <https://literature.hpcalc.org/> or at the MoHPC most easily). For its immediate predecessors, the *WP 34S* and *WP 31S*, see <https://sourceforge.net/projects/wp34s/>. For them and WP43, feel free to check <https://forum.swissmicros.com/viewforum.php?f=44> as well – or turn to <https://forum.swissmicros.com/viewtopic.php?f=44&t=1816> for WP43 exclu-

sively. And for the *WP43* offsprings *C47* and *R47*, you find almost everything at <https://forum.swissmicros.com/viewforum.php?f=41>.

All these models are covered on <https://clones.phweb.me/index.php> to some extent (the author prefers anonymity apparently but is called *Pierre* elsewhere). And there is a big calculator comparison table for *HP*, *SM*, and *WP* models at https://drive.google.com/file/d/1CNxktf9obofCWqHE_7H1cv7vUCVRQmMO/view from the *HP-11C* on, compiled by *Steve Simpkin*.

Watch it – not every second source may be up-to-date nor accurate.

Presently, there are more projects (ab-) using *DM42*, *DM42n*, or *DM32*. Also activities tweaking *HP-12C* or *HP-15C* (*CE*) calculators (or their clones) by various authors can be observed. Just browse the net.

APPENDIX K: THE MOST HONORABLE MEMBERS OF THE PILOT RUN

We know of 30 pilot samples distributed in 2022-10 and the weeks thereafter. *SM* tells the pilot run delivered 35 *WP43* calculators. The following serial numbers are owned by people communicating on the *SM* forum sometimes, listed with their member names:

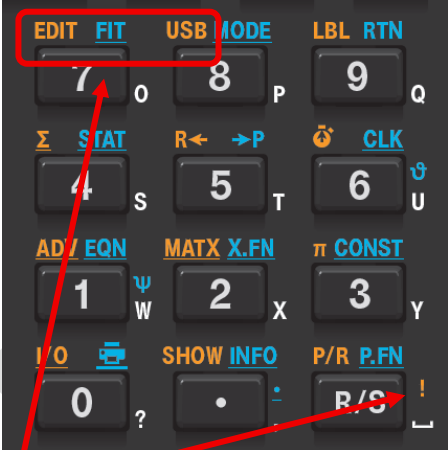
00000	<i>Michael,</i> <i>Walter</i>	00025	<i>toml_12953</i>
00001		00026	<i>gmac42</i>
00002	<i>DA74254</i>	00027	
00003	<i>(Walter)</i>	00028	
00004		00029	<i>Mihail</i>
00005	<i>Over_score</i>	00030	<i>JanneF</i>
00006		00031	<i>Mihail</i>
00007	<i>Over_score</i>	00032	<i>ctrlkws</i>
00008		00033	<i>sca</i>
00009	<i>Thomas_ER</i>	00034	
00010	<i>tuxfish</i>	00035	<i>srk00</i>
00012	<i>RJvM</i>	00036	
00013	<i>rprosperi</i>	00037	
00014	<i>rprosperi</i>	00038	
00015	<i>Jaymos</i>	00039	
00016	<i>Dany R.</i>	00040	<i>HP-Collection</i>
00017		00041	
00018		00042	<i>dlachieze</i>
00019		00043	
00020		00044	<i>Patrick</i> ²³¹
00021	<i>pauli</i>	00045	
00022	<i>pcscote</i>		
00023	<i>HP-Collection</i>	000xx	<i>akaTB</i>
00024	<i>cdmackay</i>	000xx	<i>ben.titmus</i>

Meanwhile, the following surface changes were implemented:

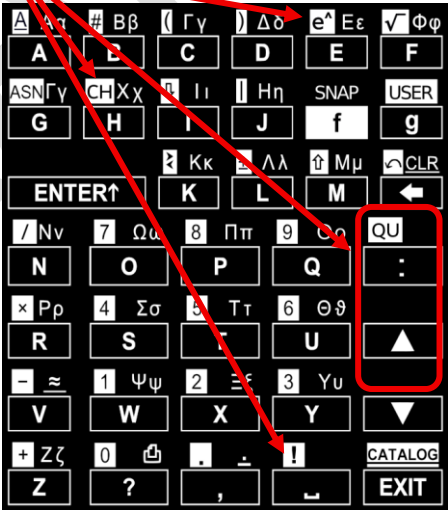
Pilot

Today

Keyboard bezel:



Rear alpha helper:



That's all – in > 40 months (→ [App. M](#)). WP43 is quite settled meanwhile.

²³¹ 28 pilot serial numbers are known. 2 more were distributed, so we know of 30. Michael tells 35 were produced and delivered. The max. serial number we know is 00044 – this looks like a scrap/rework rate of ≥ 20% for that run featuring standard DM42 electronic hardware in the summer of '22.

APPENDIX L: MAKING A WP43 STAND FOR LESS THAN \$2 COST

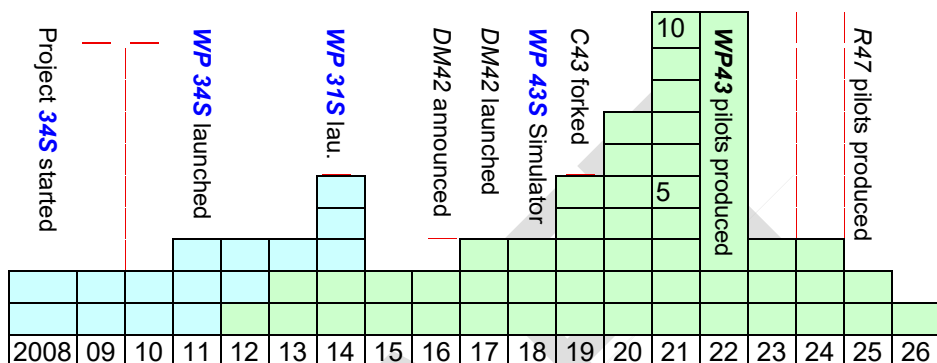
Your WP43 is conveniently carried in your pocket and comfortably held in your hand. Operating it on your desk, depending on actual lighting conditions, you may want it a bit more inclined.

The stand pictured here was made of a bamboo smartphone stand distributed by a big Swedish furniture shop (offered for less than 2 US\$ in 2022 or 1,59 € in 2026), using a flat file. Feel free to glue or stick 3 or 4 little rubber snippets or the like on its bottom to keep it from sliding on slippery surfaces, drill a decent 1 mm hole for the inevitable reset pin, and you will be done.



APPENDIX M: RELEASE NOTES

As mentioned in the OM, this hobbyist's project started as a pipe dream. I just wrote specs for 4.5 years before WP 43S got its first software guy (*Martin*) writing code. This table visualizes the maximum number of WP calculator project team members and some important events in all those years:



Version	Date	Release notes
0	29.11.12	Official project start with the first publication of the <i>43S</i> concept and its layout (https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234685#234685). Though internal discussions started in September 2011, and there are even far older traces of a ' <i>43S</i> ' denoting a ' <i>Super HP-42S</i> ', though in various more or less fictional cases – pure <i>vapourware</i> ™.
0.1	2.2.14 23.5.15	Manual set up based on the one of <i>WP 34S</i> . Passed to <i>Jake Schwartz</i> , <i>Eric Smith</i> , and <i>Richard Ottosen</i> for first information. See the vivid discussions starting in February 2014 in https://www.hpmuseum.org/forum/thread-593.html (it was closed at 10.1.16 but it still is among the three threads most viewed on that forum after over ten years).
0.2	3.10.15	Update based on <i>Jake</i> 's feedback and further thoughts, distributed to <i>Eric</i> , <i>Jake</i> , <i>Marcus von Cube</i> , and <i>Paul Dale</i> .
0.3	21.3.16	Split the manual in three; moved LBL onto the keyboard, renamed STOM to STOCFG , RCLM to RCLCFG , SERR to s_m , and SERR_w to s_{mw} ; refined the <i>Key Response Table</i> . Passed to <i>Michael Steinmann</i> for information.

Version	Date	Release notes
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 . Added <i>multiples of π</i> . Changed keyboard layout.
0.6	22.8.17	Merged the Applications and Owner's Manual. Changed the input order of <i>complex number</i> parts on <i>Paul's</i> request. Changed keyboard layout introducing D.MS, SST, BST, and % while removing $\hat{y}$, 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 <i>character</i> sets for better use of display space. Added the QRG .
0.7	2.4.18	Changed keyboard layout. Replaced the labels BST by $\Xi\Delta$, SST by $\Xi\nabla$, and UNDO by ; added some alpha input mode reminders on the keyboard. Added AGRAPH , CLLCD , EQ.xxx , HYP , J/G , M.GOTO , ORTHOF , PIXEL , POINT , TDISP , and USER . Moved the background considerations out of <i>ReM App. D</i> . Introduced K as <i>alpha register</i> for alphanumeric constants in programs. Removed <i>fraction data type</i> . Extended <i>items</i> from 6 to 7 characters to match <i>HP-42S</i> . Specified <i>DTs</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. Put SUMS in STAT . Renamed the trigonometric and hyperbolic functions according to mathematical standards, and CHR to CHAR . Redistributed the chapter about constants. Modified STATUS display. Refined the unit conversions to ensure <i>S/</i> on one side. Specified 0 SEED . Expanded <i>App. A</i> . Added formula output for L.R. Modified CPX? , DBL? , and REAL? . Changed output of binary tests for compatibility with HP-42S.
0.8	7.5.18 20.9.18	Changed keyboard layout: introduced TRG containing trigonometric functions, removed HYP into EXP and Π to g-shifted $\angle$, swapped some shifted labels. Refined the chapters about <i>register arithmetic</i> , <i>Command Parameter Input</i> , <i>Alphanumeric Input</i> , <i>Matrix Calculations</i> , and <i>Orthogonal Polynomials</i> . Introduced CLCVAR and more vintage examples. Rearranged <i>temporary information</i> on the screen. Renamed REGS to RBR and CLx to CLX . Deleted ANGLE . Corrected errors and inconsistencies. Added one more example. Moved the key response table into an appendix.

Version	Date	Release notes
0.9		Martin set up the project in <i>GitLab</i> in 2018-12. Removed <i>angle DT</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. Added the overlay drawing. Modified functionalities of EXIT and $\frac{1}{x}$ to match <i>HP-42S</i> . Added a chapter about curve fitting. Modified functionalities of ENTER and $\leftarrow$. Expanded <i>App. K</i> . Renamed DOUBLE to $\rightarrow DP$. Added $\rightarrow SP$ and conversions of <i>quarts</i> . Rearranged X.FN . Replaced USR by UM . Changed keyboard moving UM , $\frac{1}{x}$, and TRI . Moved $\frac{1}{x}$ to f(R/S) . Added XIN and XOUT . Added a chapter in <i>App. E</i> and information about infinite integers. Extended the domain of GCD and LCM . Refined and corrected.
0.10	3.1.19	Returned <i>angle DT</i> and αSR . Added IDIVR and VANGLE . Refined FP , IP , IMPFRC , PROFRC , SDIGS? , $\rightarrow DP$, $\rightarrow HR$, $\rightarrow INT$, $\rightarrow REAL$, $\rightarrow SP$, explanation of ALL , the summary of integer functions, and handling of long alpha strings. Modified contents of CPX , MATX , and $\alpha \bullet$. Added a summary of matrix functions. Removed the ON -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 $\frac{1}{x}$ from the keyboard. Renamed X_u to X_e for the distributions.
0.11	8.5.19	Changed keyboard making CC primary and user mode shifted, removing x^2 , $x\rightleftharpoons$, and DSP , adding $ x $, DROP , and SHOW , and moving some shifted labels. Modified BITS , CLREGS , CNST , CPX , DISP , EXP , INTS , MODE , PARTS , SHOW , STAT , $\rightarrow$, $\alpha MATH$, the division matrix, <i>DT</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 <i>variables</i> . Refined <i>Sect. 6</i> . Added $\bar{x}H$, $\bar{x}RMS$, nine statistical sums and five curve fit models. Split STAT in STAT and SUMS ; renamed RMDR to RMD , L_n to L_m , $L_{n\alpha}$ to $L_{m\alpha}$, Π to Π_n , Σ to Σ_n , and some constants to avoid search ambiguities. Refined <i>App. J</i> , <i>Sect. 3 and 4</i> , $\rightarrow INT$, CLR , and the functions of $\blacktriangle$ and $\blacktriangledown$. Put SUMS instead of RMD on the keyboard, moved ADV , BITS , CATALOG , EQN , FILL , INTS , MATX , MODE , PROB , RTN , SHOW , STAT , and αFN . Rearranged A...Q and <i>Sect. 2</i> of the <i>OM</i> .
0.12		Rearranged the appendices of the <i>ReM</i> from <i>App. D</i> on. Expanded <i>App. A</i> of the <i>OM</i> and <i>App. K</i> . Deleted the <i>standardized normal distribution</i> Φ and rearranged PROB . Updated CNST following <i>CODATA 2018</i> . Renamed the angular conversions. Changed the composing and cutting functionality of CC . Refined exiting <i>short integer</i> input. Expanded <i>App. D</i> . Specified max. size of <i>long integers</i> .

Version	Date	Release notes
	16.10.19	Changed keyboard adding $\mathbb{A}$, moving <u>CPX</u> , <u>FIN</u> , <u>RBR</u> , <u>R†</u> , and <u>SHOW</u> , removing <u>%</u> . Renamed <u>VANGLE</u> to <u>V$\mathbb{A}$</u> . Modified <u>CPX</u> , <u>MATX</u> , <u>TRI</u> , and <u>X.FN</u> . Rearranged <u>OMS 1</u> . Added some internal <i>DTs</i> to <i>App. B</i> ; reduced the range of <i>long integer</i> results and <i>DP</i> real inputs to $10^{\pm 999}$. Defined the domains of e^x-1 , <u>IDIVR</u> , <u>LN(1+x)</u> , <u>MOD</u> , and <u>RMD</u> according to the <u>HP-42S</u> ; modified <u>PL0T</u> and <u>$\Sigma+$</u> . Refined the <i>Addressing Tables</i> . Added a <i>DT</i> matrix for <u>IDIVR</u> . Refined the <i>Special Results</i> in <i>App. B</i> .
0.13	30.11.19	Expanded the alpha keyboard and <i>App. I</i> . Modified <u>CPX</u> , <u>INTS</u> , <u>MODE</u> , <u>PROB</u> , <u>STK</u> , <u>TEST</u> , <u>$\alpha\bullet$</u> , <u>SHOW</u> , and <u>STATUS</u> . Refined the sorting order of <i>items</i> , <u>ALL</u> , <u>CX$\rightarrow$RE</u> , <u>MEM?</u> , <u>RE$\rightarrow$CX</u> , <u>RBR</u> , <u>RM</u> , <u>SLVQ</u> , and <u>U$\rightarrow$</u> . Started filling <i>App. F</i> and <i>G</i> . Refined <i>App. 2</i> . Added a <i>long integer</i> example, <u>CPXR?</u> , <u>LZ?</u> , <u>Δv_{cs}</u> , 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 <i>variables</i> , and an index in <i>App. I</i> . Changed keyboard swapping <u>MODE</u> and <u>FLAGS</u> , <u>U$\rightarrow$</u> and <u>$\mathbb{A}\rightarrow$</u> , moving <u>CPX</u> , <u>FILL</u> , <u>RBR</u> , <u>R†</u> , <u>USER</u> , <u>α.FN</u> , <u>αINTL</u> , <u>$\vee$</u> , and <u>$\ominus$</u> , displaying <u>PRINT</u> , <u>RMD</u> , <u>STATUS</u> , <u>x^2</u> , and <u>$\cdot$</u> , and removing <u>$\frac{c}{d}$</u> , <u>$\ominus x$</u> , <u>$\rightarrow$SP</u> , and <u>$\rightarrow$DP</u> . Renamed <u>DISP</u> to <u>DSP</u> and <u>SUMS</u> to <u>Σ</u> , changed <u>$\boxplus$</u> to <u>$\boxtimes$</u> . Refined the addressing tables and <i>catalog</i> access, <u>$\alpha b/c$</u> , <u>ADV</u> , <u>BATT?</u> , <u>BITS</u> , <u>CATALOG'CHARS</u> and <u>'MENUS</u> , <u>CLALL</u> , <u>CLFALL</u> , <u>CPX</u> , <u>EXP</u> , <u>GAP</u> , <u>INTS</u> , <u>I/O</u> , <u>MODE</u> , <u>NEIGHB</u> , <u>PARTS</u> , <u>PRIME?</u> , <u>P.FN</u> , <u>SHOW</u> , <u>STAT</u> , <u>STK</u> , <u>X.FN</u> , <u>αINTL</u> , and <u>$\alpha\bullet$</u> . Deleted all 16-digit (i.e. <i>SP</i>) <i>DTs</i> as well as <u>A...Z</u> and the commands <u>CLK12</u> , <u>CLK24</u> , <u>CPXi</u> , <u>CPXj</u> , <u>CPXRES</u> , <u>CPXR?</u> , <u>DBL?</u> , <u>DENANY</u> , <u>DENFAC</u> , <u>DENFIX</u> , <u>ENG0VR</u> , <u>FAST</u> , <u>IMPFRC</u> , <u>LZOFF</u> , <u>LZON</u> , <u>LZ?</u> , <u>MULT$\times$</u> , <u>MULT$\cdot$</u> , <u>POLAR</u> , <u>PROFRC</u> , <u>QUIET</u> , <u>RDX$\cdot$</u> , <u>RDX$\cdot$</u> , <u>REALRE</u> , <u>RECT</u> , <u>SCIOVR</u> , <u>SLOW</u> , <u>SSIZE4</u> , <u>SSIZE8</u> , <u>$\rightarrow$DP</u> , and <u>$\rightarrow$SP</u> . Corrected.
0.15	14.6.20	Added <u>BESTF?</u> , <u>RANGE</u> , <u>RANGE?</u> , <u>REGIST</u> , <u>SNAP</u> , and <u>s(a)</u> , as well as errors 28 and 31 – 35. Changed <u>DSZ</u> and <u>ISZ</u> to comply with <u>HP-16C</u> . Changed keyboard shifting <u>N</u> , <u>O</u> , <u>P</u> , and <u>Q</u> , swapping <u>?</u> and <u>Z</u> , moving <u>CNST</u> , <u>CPX</u> , <u>FLAGS</u> , <u>RBR</u> , <u>RTN</u> , <u>R†</u> , <u>VIEW</u> , and <u>$\ominus$</u> , removing <u>$\cdot$</u> , and adding <u>MOD</u> , <u>$\checkmark$</u> , and <u>SNAP</u> . Renamed <u>DSP</u> to <u>DISP</u> , <u>CNST</u> to <u>CONST</u> , <u>CONST</u> to <u>CNST</u> , <u>ASL.BLK</u> to <u>ASLIFT</u> , <u>SSIZE</u> to <u>SSIZE8</u> , <u>TDM</u> to <u>TDM24</u> , and the left and right sided probabilities. Refined <u>ASSIGN</u> , <u>CATALOG</u> , <u>CNST</u> , <u>DISP</u> , <u>INFO</u> , <u>NEXTP</u> , <u>PRIME?</u> , <u>PROB</u> , <u>RBR</u> , <u>RESET</u> , <u>SHOW</u> , <u>SINC</u> , <u>STAT</u> , <u>U$\rightarrow$</u> , <u>VIEW</u> , <u>$x=+0?$</u> , <u>$x=-0?$</u> , <u>$y\mathbb{x}$</u> , <u>$\alpha\rightarrow x$</u> , <u>$\mathbb{A}$</u> , as well as <i>OMS 6</i> , <i>App. B</i> , <i>C</i> , and <i>E</i> , and some looping and statistical explanations. Reduced the maximum number of <i>local</i>

Version	Date	Release notes
		<i>registers</i> from 100 to 99. Changed ALLSCI to ALLENG and RECTN to POLAR. Added <i>DT</i> matrices for powers. Corrected.
0.16	3.11.20	Added torque and mmHg conversions, ISM , LOADV , $x_{\max}$ and $x_{\min}$. Added UNDO to the <i>IOI</i> . Refined $\sqrt[3]{x}$ and the descriptions of LOAD , LOADSS , RESET , and UNDO . Marked the not-undoable <i>items</i> in the <i>IOI</i> . Renamed the constants according to the <i>OM</i> and kicked them out of the <i>IOI</i> . Added USB . Refined <i>Basic Kinds of Program Steps</i> , <i>App. E</i> and <i>F</i> . Changed bit numbering from 1...64 to 0...63. Renamed SMODE? to ISM? . Refined IM , RE , SINC , TRI , WSIZE , X.FN and the labels of torque conversions. Added SINCπ and more vintage pictures. Expanded <i>App. G</i> . Refined the power matrix. Corrected.
0.17	16.3.21	Empty menus show up. Refined $\sqrt[3]{x}$, AUTOFF , BATT? , BITS , CLFALL , CLP , CLREGS , DSZ , e^x , FP , GTO. , IP , ISZ , J/G , LOAD , LOCR , LOCR? , LN , MEM? , M.EDI , M.EDIN , NaN? , RCLL , RDP , RSD , SAVE , SDL , SDR , TDISP , WSIZE , $x=+0?$, $x=-0?$, $\sqrt[3]{y}$, y^x , $\Sigma+$, $\Sigma-$, MOD , $\sqrt{x}$, $\frac{1}{x}$, labels in $\frac{1}{x}$, the distributions, the <i>DT</i> matrices, <i>Sect. 3</i> and <i>4</i> and <i>App. 1</i> , as well as <i>App. B</i> , <i>E</i> , <i>F</i> , and <i>I</i> . Renamed ST.X etc. to X etc. Increased the number of local flags to 32. Added Chinese units of length, area, and mass to $\frac{1}{x}$. Modified many conversions. Deleted M.OLD , added DELITM , J/G? , and $\frac{1}{x}$. Moved 'no' from $\frac{1}{x}$ to $\frac{1}{x}$. Corrected.
0.18	18.4.21	Refined <i>Times</i> , the commands EXPON... , M.GET , M.PUT , M.R$\rightarrow$R , NBIN... , SAVE , STAT , and TDISP , <i>Sect. 3</i> of the <i>OM</i> , <i>App. B</i> , <i>E</i> , <i>F</i> , <i>H</i> , and <i>I</i> . Removed error 27, CATALOG'DIGITS , REGIST , and SYS.FL . Added ASSESS , CENTRL , NXTFIT , PLOT , and s_{mi} . Brought <i>wrap</i> and <i>grow</i> in the <i>status bar</i> . Corrected.
0.19	2.5.21	First release. Introduced STATS . Refined <i>App. E</i> and <i>F</i> . Corrected.
0.20	25.9.21	Changed keyboard (re-) introducing DSP , $x\rightarrow$, and $\frac{1}{x}$, making EXP and $\sqrt[3]{x}$ primary, moving BITS , CPX , π , CONST , FILL , STK , TIMER , CLK , and SHOW , swapping $\sqrt[3]{x}$ and LOOP , PRINT and TEST , SHOW , PARTS , and INTS , removing y^x and x^2 into EXP , and deleting RMD from the keyboard. Removed error 30; added errors 38, 39, and 40, the elliptic functions, HIDE , HIDE? , M.QR , M$\pi$$\rightarrowR , R\rightarrowM\pi$, and Y_y(x) . Refined ASSESS , BITS , CLK , COV , EIGVAL , EIGVEC , INDEX , LOADSS , LOADΣ , MATX , M.DIM , M.LU , M.NEW , RANGE , SAVE , SETDAT , SETTIM , s_{xy} , s(a) , TONE , VIEW , X.FN , $\Sigma+$, $\Sigma-$, $\frac{1}{x}$, two constants, the chapters about matrices, integration, some <i>DT</i> matrices, <i>App. 1</i> , <i>B</i> , <i>H</i> , and <i>I</i> . Corrected.

Version	Date	Release notes
0.21	14.1.22	Replaced the keyboard label TIMER by  ; in <i>AIM</i> , changed the label for <u>αMATH</u> , brought $\sqrt{}$ up (instead of $\sqrt{}^{}$), $e^{}$, () , and $ $ for equations. Swapped <u>CONST</u> and π as well as <u>STAT</u> and <u>Σ</u> . Added a chapter to <i>Section 2</i> , an example for differential equations, ATAN2 , CALC , CLBKUP , DRAW , ENDPMT , REDRAW , SQRT , ZOOM , and errors 47 - 50. Refined alphanumeric input, <i>local labels</i> , ALPHA , CLCVAR , FIN , INPUT , KEYG , KEYX , M.EDIN , PARTS , RCELEL , ROUND1 , SOLVE , TRI , TVM , VIEW , <u>α</u> , systems of simultaneous equations, three constants, <i>OMS 4</i> , <i>App. 3</i> , <i>A</i> , <i>B</i> , <i>H</i> , and <i>I</i> . Removed SIGN from CPX . Renamed CEIL , CONJ , CROSS , DOT , FLOOR , LN , LNβ , LNΓ , LN(1+x) , LOG_xy , and SIGN to lower case Latin, LOG₁₀ to lg , LOG₂ to lb , DENMAX to D.MAX , and PRINT to PRINTS . Defined <i>names</i> more precisely. Corrected.
0.22	13.9.22	Moved CPX , FLAGS , LBL , MODE , STK , U→ , and α.FN on the bezel, put SF and CF on the free space. Renamed BITS to BIT , FLAGS to FLAG , PARTS to PART , and relabeled PRINT to  . Labeled the menu keys. Removed the labels of the prefixes. Added FRCSRN , READP , WRITEP , histogram commands, an appendix about graphics, a chapter about I/O, examples for solid angle, elastostatics, and tool life. Renamed GT0.▲ to GT0▲ and GT0.▼ to GT0▼ , TIMER RESET to T.RES . Removed RECV , SEND , XIN , XOUT , and the <i>program top</i> indicator. Refined 0.1s , ADD , AGRAPH , ASSIGN , AUTXEQ , ATAN2 , CENTRL , CLLCD , LOADP , SOLVE , STATUS , TIMER , VARMNU , equation editing, 3 letters, 4 conversions, fraction display, <i>names</i> , <i>AIM</i> , functionalities of  , (EXIT) , and (R/S) , <i>App. 1</i> , <i>5</i> , <i>E</i> , <i>F</i> , <i>G</i> , and <i>H</i> . Corrected.
0.23	3.12.22	Relabeled WP 43S to WP43 .  is replaced by  on the keyboard. Added omicron in <i>AIM</i> . Pilot run performed by SwissMicros; pilots distributed and sold in 2022-10/11. Introduced $\sqrt{1+x^2}$, status indicators 4 and 8 , and <i>virtual keyboards</i> for RBR , FBR , STATUS , and TIMER . Elaborated about histograms. Renamed TIMER ADD to T.Σ+ . Refined <i>App. 4</i> , AGRAPH , CPX , erf , erfc , EXITALL , EXPT , J→D , LNΓ , MANT , MATR , PART , V4 , x! , <u>αMATH</u> , Γ(x) , ζ(x) , 4 , and  . Corrected.
0.23.3	21.1.23	Implemented DELITM , SAVE and LOAD of <i>variables</i> , confirmation for CLP , RBR for <i>variables</i> , and A...D and I...L keys in RBR . Reintroduced (a new) error 27. Fixed simulator crash when restarted after closed while a browser was open.
0.23.4	6.2.23	Implemented FASTFN and CLP menu . Corrected DELITM , PIXEL , AGRAPH , string concatenation, Σ+ memory leakage, CNST in-direction, and possible crash of GT0/XEQ after LOAD . Optimized some common functions.

Version	Date	Release notes
0.23.5	23.2.23	Implemented VARMNU , NUM.IN and ALP.IN . Fixed possible crash of CLP , W_p , and matrix division; corrected TAN in MULπ , KEYG and KEYX (removed K as <i>alpha register</i> , compare 0.7), DECOMP , M.LU ; corrected <i>complex</i> matrix subtraction, eigenvalues and eigenvectors, y^x , and β ; cube root with FASTFN set, COMB and PERM , special cases of lnΓ , g_d , $\rightarrow R$, and DECOMP .
0.23.6	23.2.23	As 0.23.5 but with updated manuals.
0.23.7	11.3.23	Added ACTUSB (also on the calculator keyboard), DMCP , and DUMP . Refined NEXTP ; also VARMNU , x$\rightarrow\alpha$, and handling of empty strings to meet the specs of HP-42S . Elaborated on I/O. Corrected.
0.23.8	11.3.23	Corrected a fatal bug in the simulator. Else unchanged.
0.23.9	12.3.23	Added the angular unit <i>mil</i> .
0.23.10	25.3.23	Implemented $\rightarrow BIN$, $\rightarrow DEC$, $\rightarrow OCT$, and $\rightarrow HEX$ in INTS . Refined (-1)^x , VARMNU , and the QRG . Revised and corrected.
0.23.11	8.4.23	Refined BIT , CLPALL , INTS , PAUSE , RCL , VARMNU , VIEW , # , the comparisons, <i>App. 4</i> , and the QRG . Corrected.
0.23.12	5.5.23	Added LOADST and SAVEST . Refined handling of undefined variables, I/O , SOLVE , VARMNU , x$\rightarrow\alpha$, $\int$, the <i>status bar</i> , OMS 1, 3, and 4 , <i>ReMS 3, App. B and J</i> . Corrected.
0.23.13	1.6.23	Added error 53. Removed error 19, FLASH , RAM , PRCL , and PST0 . Reorganized access to <i>FM</i> . Implemented READP and WRITEP . Refined CLREGS , LOADP , LOADR , LOADSS , LOADV , LOADΣ , PROGS , SNAP , SOLVE , <i>App. I</i> , creating <i>user menus</i> , and displaying global labels. Corrected.
0.24	1.7.23	Added DISK? and <i>character</i> translation for editing programs on a computer. Refined head text of IOI , EXITALL , INFO , INT? , J/G? , SNAP , x! , $\alpha\bullet$, some <i>names</i> of conversions, and <i>stack</i> diagrams. Removed FLASH? and $\overline{f}$. Corrected.
0.24.1 and .2	8.9.23	Expanded the <i>character</i> translation table. Removed $\square LCD$. Added BUZZ , CONFIG , MyPFN , PLAY , VOL , VOL+ , VOL- , VOL? , and V.ALL . Refined BACK , CASE , EXITALL , INFO , I/O , MVAR , NOP , P.FN , P.FN2 , SKIP , TICKS , and VARMNU . Filled <i>App. H</i> . Changed the virtual keyboard for <i>AIM</i> . Corrected.

Version	Date	Release notes
0.24.3	2.12.23	Specified <i>startup default</i> contents for <u>Myα</u> and <u>MyPFN</u> . Added <u>CLKEYS</u> , <u>CPXSLV</u> , <u>MEDDEV</u> , <u>MEDIAN</u> , <u>PERCTIL</u> , <u>Q₃-Q₁</u> , <u>REM</u> , <u>XEQ+1</u> , <u>x_{ma}-x_{mi}</u> , and appendices for <i>Sound</i> and <i>history</i> . Refined <u>ASSIGN</u> , <u>CLR</u> , <u>STAT</u> , <u>U$\rightarrow$</u> , <u>x_H</u> , <u>x_{max}</u> , <u>x_{min}</u> , <u>x_{RMS}</u> , <u>J</u> , <i>user mode</i> , the <i>IOI</i> , <i>App. A</i> , <i>F</i> , and <i>J</i> . Renamed <u>LOAD...</u> to <u>Load...</u> and <u>HIST</u> to <u>HISTOG</u> . Deleted <u>PRINTS</u> and <u>SLOW</u> . Removed printer hardware references. Corrected.
0.25	23.12.23	Added <u>CLMALL</u> , <u>CLMyM</u> , <u>CLMyP</u> , <u>CLMyα</u> , <u>CLMy...</u> , <u>CLVALL</u> , <u>day$\rightarrow$s</u> , <u>DELMALL</u> , <u>DELVALL</u> , <u>s$\rightarrow$day</u> , and <u>TIME:</u> . Refined <i>dates</i> and the backplate print, <u>CLALL</u> , <u>CLR</u> , <u>CLMENU</u> , <u>CPXSLV</u> , <u>DELITM</u> , <u>RESET</u> , and <u>U$\rightarrow$</u> . Renamed <u>XEQ+1</u> to <u>XEQ.SKIP</u> . Corrected.
0.25.1	25.1.24	Added <u>SLVC</u> . Refined lettered <i>flags</i> , <u>DELITM</u> , <u>DELMALL</u> , <u>DELVALL</u> , <u>STATUS</u> , <u>U$\rightarrow$</u> , <u>x$\rightarrow$$\alpha$</u> , and <i>App. G</i> . Corrected.
0.25.2	30.1.24	Firmware corrected.
0.26	7.7.24	Added <u>MEDIAN</u> and related commands. Split <u>STAT</u> in two <i>menus</i> . Created <u>FIT</u> . Refined and corrected. Modified <u>1/x</u> .
0.26.1	28.7.24	Expanded yes/no decisions. Refined median commands. Added <u>:</u> , <u>CH</u> , and <u>QU</u> for <i>AIM</i> . Rearranged and corrected.
0.26.2	21.8.24	Added <u>AVIEW</u> and <u>PROMPT</u> , refined <u>P.FN</u> , <u>P.FN2</u> , <u>VIEW</u> , error 11, and <i>App. E</i> . Limited calculator input for <u>NEXTP</u> and <u>PRIME?</u> . Updated bezel and backside . Refined and corrected.
0.26.3	22.9.24	Added <u>M.FIND</u> , <u>M.CMAX</u> , and <u>M.CMIN</u> . Refined <i>App. E</i> , <i>H</i> , <i>I</i> , and <i>J</i> , <u>AGRAPH</u> , <u>CONVG?</u> , <u>ERR</u> , <u>INDEX</u> , <u>INPUT</u> , <u>MSG</u> , <u>M.DIM</u> , <u>M.EDIN</u> , <u>NEXTP</u> , <u>RANGE</u> , <u>RDP</u> , <u>RESET</u> , <u>SIM_EQ</u> , <u>SOLVE</u> , <u>%ile</u> , <u>J</u> , and the <i>QRG</i> ; renamed <u>IΓ</u> to <u>Γ</u> ; rearranged <i>ReM</i> chapters and corrected.
0.27	9.11.24	Expanded the virtual keyboard for labels. Made 8 <i>stack</i> levels default. Swapped some <i>OM</i> chapters. Refined <i>App. 1</i> , <i>B</i> , <i>E</i> , <i>F</i> , and <i>I</i> . Added <u>e</u> and <u>i</u> . Refined and corrected.
0.27.1	16.12.24	Added <u>FACTOR</u> , many kitchen volumes, two examples to <i>OMS 5</i> , one to <i>App. 4</i> , and a chapter to <i>OMS 2</i> and <i>App. J</i> . Refined <u>U$\rightarrow$</u> , <u>X.FN</u> , and <i>App. A</i> . Rephrased p. 1. Refined and corrected.
0.27.2	19.12.24	Corrected.
0.28	1.5.25	Updated <u>CONST</u> following CODATA 2022. Renamed <u>RCL$\uparrow$</u> to <u>RCLMAX</u> , <u>RCL$\downarrow$</u> to <u>RCLMIN</u> , <u>RCLS</u> to <u>RCLSTK</u> , and the respective store commands. Removed page background color. Refined and corrected.

Version	Date	Release notes
0.28.1	9.5.25	Refined the <i>IOI</i> to contain (almost) all <i>items</i> . Added another example. Refined and corrected.
0.28.2	15.7.25	Split LoadPE into LoadPr and LoadEq . Extended LoadR and the <i>OM</i> and <i>ReM</i> indices. Changed RTN+1 to RTN.SKIP . Deleted LOADST , refined LOAD... and I/O . Put CONST in CATALOG . Added Indian output format, generic EDIT , auxiliary constants, keystroke numbers, <i>App. 4</i> about PLOT , two chapters to <i>App. H</i> , two to <i>App. I</i> , one to <i>App. J</i> , <i>App. K</i> about the pilots, and <i>App. N</i> . Improved the explanations of CPX? , OFF , SAVEST , and <i>App. C</i> . Refined and corrected.
0.28.3	15.9.25	Added EDIT and ! to the bezel, moved USB and DMCP . Expanded <i>App. K</i> . Elaborated about angles. Refined and corrected.
0.29	3.1.26	Refined. More about angles, constants, and the <i>Integrator</i> .
0.29.1	20.2.26	Added two new error messages. Refined in particular <i>OMS 1</i> , <i>ReMS 1</i> , <i>App. G</i> , <i>H</i> , and <i>N</i> , and the font tables. Got rid of 3 colors. Corrected.
0.30	8.6.26	Renamed most commands of INFO from xxx? to ?xxx . Introduced M.C.C. . Refined the fonts reducing abuse of <i>UTF</i> codes. Added 2 characters to gMATH , examples in <i>OM</i> and <i>ReM</i> , and a chapter to <i>App. I</i> . Got rid of 2 more colors in <i>OM</i> and 1 in <i>ReM</i> . Rearranged chapters in <i>OMS 2</i> . Refined <i>App. H</i> . Split <i>OMS 2</i> shifting <i>Sections 3ff</i> up. Inserted lots of links. Updated, purged, and corrected. ATTENTION: The manuals are ahead of the software!

APPENDIX N: INDEX OF NAMES, CALCULATORS, COMPANIES, ETC.

MS Word generated the following index automatically. Its sorting order neither complies to the *IOI* above nor follows a comprehensible logic overall (I changed it manually only where it was blatantly counterintuitive). And unfortunately *MS Word* doesn't produce a linked index for unknown reasons (what *AI* responds in this matter is simply off) – but **<ctrl> g** is your friend jumping to the relevant pages. This 15-page index may be helpful nevertheless, since

1. it shows all *items* provided, though in another order you may be more accustomed to (so you know which *names* you must not use when creating new *user items*), and
2. it covers the entire text above plus the [Background Facts and Considerations](#) below (but not the *Quick Reference Guide* since this is regarded as a separate short document).

Keytop *labels* are printed **white**, keyboard labels **blue** or **gold** according to their appearance on the calculator, *submenus* (as well as **EXP** and **TRI**) **black**. Remember that the *labels* **a b/c**, **#**, **d.ms**, **.d**, **h.ms**, **(α)**, **←**, **▲**, **▼**, **P/R**, and **(R/S)** don't represent *items* (→ 15).

What is not an *item* provided is printed in *italics* in this index. For *living persons*, only their prenames are listed here unless they have published under their full names, while historic persons or fictional names are listed fully.

Looking for a particular topic, check also the [Table of Contents](#)

-	106	R↓	80	
/	107	RCL	73	#B 116
+	106	STO	87	#DEC 116
+/-	106	TRI	92	% 112
×	106	↔	98	%+MG 113
√x	113	XEQ	96	%ile 112
e^x	40	1/x	20	%MRR 112
EXP	39, 143	←	15, 30, 151, 166, 170, 213, 240, B-1	%T 112
ln	54			

%Σ 113

$(-1)^x$ 106

$(x^2)^{-1}$ 105

?BATT 107

?BestF 107

?DISK 107

?HIDE 107

?ISM 108

?J/G 108

?KTYP 108

?LocR 108

?M.DIM 108

?MEM 108

?RANGE 109

?RM 109

?SDIGS 109

?SSIZE 109

?VOL 109

?WSIZE 109

$[M]^{-1}$ 106

$[M]^T$ 106

^MOD 107

$||$ 112

$|M|$ 112

$|x|$ 112

$\pm\infty?$ 106

xMOD 107

$\sqrt[3]{x}$ 20

$\sqrt[y]{x}$ 97

$\frac{1}{x}$ 114

$\frac{1}{x \rightarrow}$ 114, 151

$\int$ 113

$\int f$ 114, 143

$\int f dx$ 114, 142

↑Lim 111

→ 109

→BIN 109

→D.MS 109

→DATE 109

→DEC 109

→DEG 109

→GRAD 109

→H.MS 110

→HEX 110

→HR 110

→INT 110

→MIL 110

→MULπ 110

→OCT 110

→POL 110

→RAD 110

→REAL 111

→REC 111

↓Lim 111

↔ 111

°C→°F 20

°F→°C 20

 146

 # 116

 ADV 114

 CHAR 114

 DLAY 114

 MODE 114

 PROG 115

 r 115, 116

 REGS 115

 STK 115

 TAB 115

 USER 115

 WIDTH 115

 Σ 116

0.1s 20

10^x 20

1COMPL 20

2^x 20

2COMPL 20

∞ 113, 123, 199

$-\infty$ 106, 123, 199

a 20, 117

A 20

A: 24, 147

$a_{\oplus}$ 24, 117

a_0 20, 117

ABS 20

$ac \rightarrow m^2$ 21

$ac_{us} \rightarrow m^2$ 21

ACC 20

ACOS 20

ActUSB 20

ADD 21, 105
 ADM 21
 ADV 21, 142
 AGM 21
 AGRAPH 21
 Albert C. 340
 Albillo, Valentin 263, 316, 341
 ALL 21
 ALLENG 21f, 132
 ALP.IN 22, 132
 ALPHA 22, 131
 a_{Moon} 22, 117
 AND 22
 ANGLE 22
 ANGLES 22
 arccos 22
 arcosh 22
 arcsin 22
 arctan 23, 24
 artanh 23
 ASIN 23
 ASLIFT 23, 133
 ASR 23
 ASSESS 144
 ASSIGN (ASN) 23
 ATAN 24
 atm→Pa 24
 atom 125
 au→m 24
 AUTOFF 24, 132
 AUTXEQ 24, 132
 AVIEW 24
 Avogadro, Amedeo 120
 B 24
 Babylon 130
 BACK 24
 bacteria 126
 bar→Pa 25
 bbl→m³ 25
 BC? 25
 BEEP 25
 BeginP 25
 Bellatrix 128
 Bernoulli 44, 271, 274
 Bessel 50, 99, 329
 BestF 25
 Binom_Δ 26
 Binom⁻¹ 26
 Binom_Δ 26
 Binom_Δ 26, 146
 Binom_p 26
 BIT 26, 142
 B_n^{*} 26
 B_n 26
 Bohr 117, 122
 Boltzmann 119, 123
 Braun B-4
 Breit-Wigner 27, 276, 282
 BS? 26
 Btu→J 26
 Bürgel, Bruno 127
 BUZZ 26
 c 26, 117
 C 26
 c₁ 26, 118
 c₂ 26, 118
 C47 7
 cal→J 26
 CALC 26
 cancer 125
 CARRY 27, 130
 CASE 27
 Cassini 125
 Catalan 118
 CATALOG 27
 Cauch⁻¹ 27
 Cauch_Δ 27
 Cauch_Δ 27
 Cauch_Δ 28, 146
 CauchF 27
 Cauch_p 27
 Cauchy 27, 276, 282
 CB 28
 Čebyšev 91, 93, 324, 325
 CEIL 28
 CENTRL 28
 César R. 337
 CF 28
 CHARS 29
 chī→m 29

CLall 29
Classics 3, 255
 CLBKUP 29
 CLCVar 29
 CLFall 29
[CLK](#) 29, 143
 CLKEYS 29
 CLLCD 29
 CLMall 29
 CLMENU 29
CLMy... 29, 143
 CLMyM 29
 CLMyP 29
 CLMyα 29
 CLP 30
 CLPall 30
[CLR](#) 30, 143
 CLREGS 30
 CLSTK 30
 CLVall 30
 CLX 30
 CLΣ 30
 cn(u,m) 30, 327
 CNST 30
CODATA 116
 COMB 31
Commodore 333
Compton 122
 CONFIG 31, 136,
 137
 CONJ 31
[CONST](#) 31, 143
 CONVG? 31
 CORR 31, 72
 cos 32
 cosh 32
 COV 32
 cp_I→mℓ 32
 cp_{UK}→mℓ 32
 cp_{US}→mℓ 32
[CPX](#) 32, 143
 CPX? 32
 CPXRES 32, 129
[CPXS](#) 32
 CPXSLV 32
 cross 33
 ct→g 33
 ct→kg 33
 c_{Un}→m 33
curve fitting 344
 cwt→kg 33
 CX→RE 33
 D 33
 D.MAX 37
 D.MS 37
 D.MS→ 37
 D.MS→D 37
 D.MY 37
 D→D.MS 37
 D→J 37
 D→MIL 37
 D→R 37
Dante Alighieri
 271
dark matter 125
data types 189
 DATE 33
 DATE→ 33
DATES 33
David J. B-6
 DAY 34
 day→s 34
 dB→fr 34
 dB→pr 34
 DBL/ 34
 DBL× 34
 DBLR 34
 DEC 34
 DECIM. 34, 132
 DECOMP 34
 DEG 34
 DEG→ 35
del Rey,
 Fernando 316
 DelITM 35
 DelMall 35
 DelVall 35
 DENANY 35, 130
 DENFIX 35
 DENMAX 35
 DET 35
Dieter → *Rams*
[DISP](#) 35, 143
DM32 349
DM42 1, 3, 182,
 183, 186, 206,
 243, 244, 245,
 247, 251, 348,
 349, B-6, 427

<i>DM42n</i> 184, 349, B-7	ENG 38	<u>F&p:</u> 43, 147
DMCP 35, 107, 192, 245, 246, 247, 248, 254, B-4	ENORM 38	$F^{-1}(p)$ 42
DMY 35, 129	ENTER↑ → ENTER↑	$f'(x)$ 43
dn(u,m) 35, 327	ENTRY? 39	$f''(x)$ 43
dot 35	EQ.DEL 39	$F_{\Delta}(x)$ 42
DRAW 36	EQ.EDI 39, 143	$F_{\Delta}(x)$ 42
DROP ↓ 36	EQ.NEW 39	<u>F:</u> 43, 146
DROPy 36	<u>EQN</u> 39, 143	F_{α} 43, 118
DSE 36	<i>Eratosthenes</i> 335	F_{δ} 43, 118
DSL 36	erf 39	FACTOR 40, 335
DSP 36	erfc 39	<i>factorial</i> 55, 97, 335
DSTACK 36	<i>Eric R.</i> 239, 316	<i>Faraday</i> 118
DSZ 37	<i>Eric S.</i> 353, B-6	FASTFN 41, 131
DUMP 37	ERR 39	FB 41
 	<i>ET66</i> B-4	FBR 41
e 37, 118	<i>Euclid</i> 38, 93	FC? 41
e 40, 118	<i>Euler</i> 54, 101, 118, 121	FC?C 41
E(m) 40, 327	EVEN? 39	FC?F 41
E(φ ,m) 40	EXIT → EXIT	FC?S 41
<u>E:</u> 40, 147	EXITall 39	FCNS 41
e^x 40	<u>EXP</u> 39, 143	<i>Feigenbaum</i> 118
e^{x-1} 40	ExpF 39	FF 41
EDIT 38	Expon $_{\Delta}$ 40	FIB 41
EIGVAL 38	Expon $_{\Delta}$ 40	<i>Fibonacci</i> 41, 273
EIGVEC 38	Expon $^{-1}$ 40	FILL 41
<i>Eisberg</i> 303	<u>Expon:</u> 40, 146	FIN 41, 144
<u>Ellipt</u> 38, 149, B-3	Expon $_p$ 40	FIT 41, 144
END 38	EXPT 40	FIX 41
ENDP 38	 	FLAG 41, 144
ENDPMT 38	F 40, 118	FLOOR 41
	<u>f'</u> 43, 143	fm.→m 42
	<u>f''</u> 43, 143	

FP 42	GCD 44	ha→ac 46
$F_p(x)$ 42	g_d^{-1} 44	ha→m ² 46
FP? 42	g_d 44	<i>Heathkit</i> 261
fr→dB 42	g_e 44, 118	<i>helium</i> 125
FRACT 42, 129	Geom _△ 44	<i>Hermite</i> 46, 321, 322
FRCSRN 42, 129	Geom _△ 44	HIDE 46
<i>Free42</i> 247, 267, B-11	Geom ⁻¹ 44	HISTO 46
FS? 42	Geom: 44, 146	HISTOG 46, 147
FS?C 42	Geom _p 44	HISTOX 46
FS?F 42	<i>George</i> → <i>Orwell</i> 266	HISTOY 46
FS?S 42	$gl_{US} \rightarrow m^3$ 44	H _n 46
ft.→m 42	$gl_{UK} \rightarrow m^3$ 44	HNORM 46
ft _{US} →m 42	G _M 44	H _{np} 46
FV 42	GM _⊕ 118	<i>HP</i> 16, 17, 260, 263, 268, 282, 337, 4, 5, 22
fz _{US} →m ³ 42	GRAD 44	hp _{UK} →W 46
fz _{UK} →m ³ 42	GRAD→ 45	<i>HP-10C</i> 256
fz _{US} →mℓ 42	GRAMOD 45	<i>HP-12C</i> 345
G 43, 118	<i>Gregorian calendar</i> 51	<i>HP-15C</i> 17, 260, 337, 25
$g_{\oplus}$ 46, 118	<i>Gregorian year</i> 117	<i>HP-16C</i> 15, 24, 37, 50, 54, 58, 67, 76, 77, 78, 84, 93, 95, 116, 130, 192, 215, 221, 223, 239, 2
g→ct 45	GROW 45, 130	<i>HP-19BII</i> 268
g→oz 45	GTO 45	<i>HP-19C</i> 17, 37, 50, 263
g→trz 45	GTO. 45	<i>HP-20b</i> 4
G ₀ 43, 118	GTO.. 45	<i>HP-21</i> 265
<i>galaxy</i> 125	GTO↑ 45	
<i>Galton</i> 272	GTO↓ 45	
GAP 43	<i>Gudermann</i> 44, 328	
<i>Gauss</i> 44, 116, 279, 281, 284, 302, 344	h 46, 118	
GaussF 44	ħ 47, 118, 27	
G _C 44, 118	ha→ac _{US} 46	

<i>HP-21S</i> 42, 92, 105, 215, 223, 224, 239	Hyper _Δ 47	ISE 49
<i>HP-28C</i> 261	Hyper _Δ 47	ISG 49
<i>HP-28S</i> 22	Hyper ⁻¹ 47	ISM 49
<i>HP-29C</i> 17, 37, 50	Hyper: 47	ISZ 50
<i>HP-30b</i> 4	Hyper: 146	I _{xyz} 50
<i>HP-34C</i> 17, 210, 239, 337	Hyper _p 47	J 50
<i>HP-35</i> 260, 260, 265, 5	HypF 47	J- 50
<i>HP-35S</i> 35	I 47	J/G 51
<i>HP-41C</i> 17, 188, 25, 1	I- 50	J+ 50
<i>HP-41CX</i> 5	<i>i</i> 14, 50, 118, 347, 27	J→Btu 51
<i>HP-42S</i> 15, 21, 22, 25, 45, 48, 52, 62, 77, 78, 88, 98, 100, 131, 132, 215, 221, 226, 239, 336, 337, 6, 22, 25, 31, 33	i%/a 50	J→cal 51
<i>HP-48</i> 188, 32	I/O 50, 144	J→D 51
<i>HP-50g</i> 262, 25	I+ 50	J→Wh 51
<i>HP-55</i> 17	IDIV 47	<i>Jaco M.</i> B-7, 22
<i>HP-67</i> 17, 37, 50	IDIVR 47	<i>Jacobi</i> 102, 327
<i>HP-71</i> 67	IGN1ER 47, 132	<i>Jake S.</i> 353
<i>HP-9100A</i> 259	iHg→Pa 47	<i>Jean-François G.</i> 340
<i>HP-97</i> 17	Im 47	<i>Jeff R.</i> 188
hp _E →W 46	in.→m 49	jīn→kg 50
HPL0T 46, 147	in.→mm 49	<i>Jobs, Steve</i> B-4
hp _M →W 46	INC 47	<i>Josephson</i> 119
<i>Hubble</i> 125	INDEX 48	<i>Julian day</i> <i>number</i> 37, 51, 190, 197
<i>hydrogen</i> 125	INFO 48, 144	<i>Julian year</i> 158
	INPUT 48	<i>Jupiter</i> 128
	INT? 49	J _y (x) 50
	<i>Integrator</i> 337	
	INTING 48, 133	
	INTS 48, 144	
	INVRT 49	
	<i>ion</i> 125	
	IP 49	
		k 51, 119
		K 51
		K(m) 52, 327

Kahan, William

194, 337

KEY 51

KEY? 52

KEYG 51

KEYX 51

kg→cwt 52

kg→jīn 52

kg→lb. 52

kg→liǎ 52

kg→oz. 52

kg→s.t 52

kg→scw 52

kg→sto 52

kg→ton 52

kg→trz 52

K_J 52, 119

km→mi 52

km→nmi 52

Kolb, Walter 282

L 53

L.INTS 56

L.R. 57

l.y.→m 57

ℓ→gal_{us} 57

ℓ→gal_{uk} 57

ℓ→qt. 57

Laguerre 54, 323

Lambert 95, 331

Lanczos 123

Landé 118

LASTx 53

lb 53

lb.→kg 53

lbf→N 53

LBL 53

LBL? 53

LCM 53

LEAD.0 130

LEAP? 53

Legendre 70, 324

lft→Nm 53

lg 20, 53

LgNrm_Δ 54

LgNrm⁻¹ 54

LgNrm_Δ 54

LgNrm: 54, 146

LgNrm_p 54

lī→m 54

liǎ→kg 54

LinF 54

LJ 54

L_m 54

L_{mα} 54

ln 55

ln(1+x) 55

LNβ 54

LNΓ 55

LoadAll 55, 240,
248

LoadEq 55

LoadPr 55

LoadR 55

LoadSS 55

LoadV 56

LoadΣ 56

LocR 56

LogF 56

Logis_Δ 56

Logis_Δ 56

Logis⁻¹ 56

Logis: 56, 146

Logis_p 56

LOG_xy 56

LOOP 56, 145

Lorentz 27, 276,
282

LOWBAT 56, 131

l_{pl} 56, 119

Luiz V. 12

M.↑ 65

M.→ 65

M.↓ 65

M.← 65

M.C↻C 61

M.CMAX 61

M.CMIN 61

M.DELR 62

M.DIM 62

M.DY 62

M.EDI 62

M.EDIN 62

M.EDIT 62, 145

M.FIND 62

M.GET 62, 64

M.GOTO 63

M.GROW 63

M.INSR 63

M.LU 63	$m^2 \rightarrow ha$ 57	$mi \rightarrow km$ 59
M.NEW 64	$m^2 \rightarrow m\ddot{u}$ 57	$mi \rightarrow m$ 59
M.PUT 64	$m^3 \rightarrow fz_{UK}$ 57	<i>Michael S.</i> 351,
M.QR 64	$m^3 \rightarrow fz_{US}$ 57	353, 6
M.R&R 64, 226	$m^3 \rightarrow gl_{US}$ 57	<i>Michalski</i> 337
M.SIMQ 58, 64,	$m^3 \rightarrow gl_{UK}$ 57	MIL 59
141, 145	$m^3 \rightarrow bbl$ 57	MIL $\rightarrow$ 59
M.SQR? 64	MAD 58	MIL $\rightarrow$ D 59
M.WRAP 65	MANT 58	MIL $\rightarrow$ R 59
<u>m</u> : 65, 148	<i>Marcus v. C.</i> 353	min 59
$M_{\oplus}$ 66, 120	<i>Martin L.</i> 353	MIRROR 59
m_{μ} 61, 119	<i>Mascheroni</i> 121	$m\ell \rightarrow cp_{US}$ 59
$M_{\odot}$ 66, 119	MASKL 58	$m\ell \rightarrow cp_{UK}$ 59
$m \rightarrow au$ 65	MASKR 58	$m\ell \rightarrow cp^I$ 59
$m \rightarrow ch\ddot{i}$ 65	Mat.A 58, 137	$m\ell \rightarrow fz_{US}$ 59
$m \rightarrow c\ddot{u}n$ 65	Mat.B 58, 137	$m\ell \rightarrow fz_{UK}$ 59
$m \rightarrow f\ddot{e}n$ 65	Mat.X 58, 137	$m\ell \rightarrow pt_{UK}$ 60
$m \rightarrow ft_{US}$ 65	Mat_A 58, 137	$m\ell \rightarrow pt_{US}$ 60
$m \rightarrow ft.$ 65	Mat_B 58, 137	$m\ell \rightarrow tb_{UK}$ 60
$m \rightarrow ftm.$ 65	Mat_X 58, 137	$m\ell \rightarrow Tb_{US}$ 60
$m \rightarrow in.$ 65	<i>Mathworld</i> 326,	$m\ell \rightarrow tb^I$ 60
$m \rightarrow l.y.$ 65	331, 333f	$m\ell \rightarrow ts_{UK}$ 60
$m \rightarrow l\ddot{i}$ 65	MATR? 58	$m\ell \rightarrow ts_{US}$ 60
$m \rightarrow mi.$ 65	<u>MATRS</u> 58	$m\ell \rightarrow ts^I$ 60
$m \rightarrow nmi$ 65	<u>MATX</u> 58, 145	mm $\rightarrow$ in. 60
$m \rightarrow pc$ 65	max 58	mm $\rightarrow$ pt. 60
$m \rightarrow pt.$ 65	MDY 59, 129	mmHg $\rightarrow$ Pa 60
$m \rightarrow yd.$ 65	m_e 59, 119	M_{Moon} 119
$m \rightarrow y\ddot{i}n$ 65	MEDIAN 59	m_n 60, 119
$m \rightarrow zh\ddot{a}n$ 65	MENU 59	m_n/m_p 60, 119
$m^2 \rightarrow ac_{US}$ 57	<u>MENUS</u> 59	<u>MOD</u> 60
$m^2 \rightarrow ac$ 57	<i>Mercury</i> 128	<u>MODE</u> 60

MONTH 60	<i>Nelson, Richard</i>	<i>Orwell, George</i>
<i>Mosig</i> 337	B-22	266
m_p 60, 119	<i>neon</i> 125	<i>Osborne, Tom</i> B-5
m_p/m_e 60, 119	<i>Newton</i> 118, 121	<i>Ottosen, Richard</i>
m_{PL} 60, 119	NEXTP 67	353, B-6
<i>MSA</i> 28, 70, 85	<i>Nils Holgersson</i>	OVERFL 68, 130
MSG 60	150	oz.→g 68
m_u 61, 119	<i>NIST</i> 116, 162	oz.→kg 68
$m\ddot{u}\rightarrow m^2$ 61	Nm→lbft 67	<i>P.FN</i> 72, 146
$m_u c^2$ 61, 119	nmi.→km 67	<i>P.FN2</i> 72, 146
MULT× 61, 132	nmi.→m 67	<i>P:</i> 72, 148
MULπ 61	NOP 67	p_0 120
MULπ→ 61	NOR 67	Pa→atm 69
MVAR 61	Norml _Δ 67	Pa→bar 69
<i>MyMENU</i> 61, 145	Norml _Δ 67	Pa→iHg 69
<i>MyPFN</i> 61, 145	Norml ⁻¹ 67	Pa→mmH 69
<i>Myα</i> 61, 145	<i>Norml:</i> 67, 146	Pa→psi 69
$M\pi\rightarrow R$ 61	Norml _p 67	Pa→torr 69
	NOT 67	<i>PART</i> 145
N→lbft 68	<i>Novus</i> 334	<i>Paul D.</i> 353, 6
N_A 66, 120	NPER 67	PAUSE 68
NaN 66, 120, 131,	<i>nucleus</i> 125	pc→m 69
196, 199, 213,	NUM.IN 68, 132	PER/a 69
B-18	<i>Nuts</i> 3, 255	PERM 69
NaN? 66	NXTFIT 23, 68	PGMINT 69
NAND 66	$n\Sigma$ 68	PGMSLV 69
NBin _Δ 66	ODD? 68	<i>Pierre</i> 348
NBin _Δ 66	OFF 68	<i>Pioneers</i> 3, 255
NBin ⁻¹ 66	OR 68	PIXEL 69
<i>NBin:</i> 66, 146	OrthoF 68	<i>Planck</i> 118, 119,
NBin _p 66	Orthog 68, 149	121
nBINS 66		PLAY 69, 174
NEIGHB 66		

PLOT 70, 144
 PMT 70
 P_n 70
 POINT 70
 Poiss_Δ 70
 Poiss⁻¹ 70
 Poiss_Δ 70
Poiss: 70, 146
Poisson 70, 274, 276
 Poiss_p 70
 POLAR 70, 129
 PopLR 70
 PowerF 71
 pr→dB 71
prime sieve 335
 PRIME? 71
PRINT (🖨️) 146
PROB 71, 146
PROG 71
PROGS 71
 PROMPT 24, 71, 174
 PROPFR 71, 130
 psi→Pa 71
 pt.→m 71
 pt.→mm 71
 pt_{UK}→mℓ 71
 pt_{US}→mℓ 71
PTB 116
 PUTK 71
 PV 72
 Q_3-Q_1 72
 qt.→ℓ 72
 QUIET 72, 132
 R_∞ 80, 120
 R 72, 120
 $R_\oplus$ 80, 120
 $R_\odot$ 80, 120
 R† 80
 R→D 80
 R→MIL 80
 R→Mπ 80
 R↓ 80
R47 B-7
 RAD 72
 RAD→ 72
Rams, Dieter B-4, 6
 RAN# 73
random numbers 343
 RANGE 72
 RANI# 73
 RBR 73
 RCL / 74
 RCL+ 74
 RCL× 74
 RCLCFG 73
 RCLEL 74
 RCLIJ 74
 RCLMAX 74
 RCLMIN 74
 R-CLR 79
 RCLSTK 74, 88, 212, 335, B-31
 R-COPY 79
 RDP 74
 Re 75
 r_e 75, 120
 RE→CX 76
 Re→Im 76
 READP 75
 REAL? 75
 REALDF 75
REALS 75
 REDRAW 75
 REM 75, 168, 306, 307, 309, 312, 343
 RESET 75
Richard → *Nelson*
Riemann 102, 332
Rigel 128
 RJ 76
 R_K 76, 120
 RL 76
 RLC 76
 RM 77
 RMD 77
 R_{Moon} 77, 120
 RNORM 77
 RootF 78
 ROUND 78
 ROUNDI 78
RPL 16, 262, 264, 25, 31

RPN 17, 18, 37,
172, 257, 258,
261, 262, 264,
269, 335, 4, 22,
25, 31

RR 78

RRC 78

RSD 78

R-SORT 80

RSUM 78

R-SWAP 80

RTN 79

RTN.SKIP 79

RUNIO 79, 131

RUNTIM 79, 131

Rydberg 120

s 81

s(a) 89

S.INTS 89

s.t→kg 83

s→day 89

s→year 89

Sa 81, 120

Saturn 125

SAVE 29, 55, 81,
192

SAVEST 81

Sb 81, 120, 121

SB 81

SCI 81

scw→kg 81

SDL 82

SDR 82

Se² 82, 121

Se'² 82, 121

SEED 82

SETCHN 82

SETDAT 82

SETEUR 82

SETIND 82

SETJPN 82

SETSIG 82

SETTIM 82

SETUK 82

SETUSA 82

Sf⁻¹ 83, 121

SF 83

Shainin 301

SHOW 83

sign 49, 83, 169,
192, 194

SIGNMT 83

SIM_EQ 84

sin 84

sinc 84

Sinclair 334

sinh 84

SKIP 84

SL 84

SLVC 85

SLVQ 85, 266, 24

SM 13, 35, 107,
186, 188, 249,
252, 254, 348,
350, 6

s_m 85

s_{mi} 85

s_{mw} 86

sn(u,m) 86, 234,
327

SNAP 86

SOLVE 86

Solver 87, 143,
336

SOLVING 87, 133

SPCRES 87, 131

SPEC? 87

Spices 3, 17, 255

SQRT 87

SR 87

SSIZE8 87, 132

Stansifer, Paul
128

STAT 87, 147

STATS 87

STATUS 87

Štefan 123

Stephenson 125

Steve → *Jobs*

Steve S. 337, 349

Stings 3, 255

STK 87, 147

STO- 88

STO/ 88

STO+ 88

STO× 88

sto→kg 88

STOCFG 88

STOEL 88

STOIJ 88	TIME 91	V.ALL 94
STOMAX 88	<u>Time:</u> 91, 148	V: 94, 148
STOMIN 88	TIMER (🕒) 91	V ₄ 94
STOP 88	<u>TIMES</u> 91	<i>Valentin</i> → <i>Albillo</i>
STOSTK 88	T _n 91	VAR 93
STRI? 89	<i>Tom</i> → <i>Osborne</i>	VARMNU 93
<u>STRING</u> 89	ton→kg 91	<u>VARS</u> 93
<i>Student</i> 92, 276, 279	TONE 91	VERS 93
SUM 89	TOP? 91	<u>VIEW</u> 94
s _w 89	torr→Pa 91	V _m 94, 121
s _{xy} 89	t _p (x) 92	VMDISP 94, 133
SYS.FL 89	t _{PL} 91, 121	VOL 94
	T _{PL} 91, 121	VOL- 94
T 89	<u>TRACE</u> 92, 131	VOL? 109
t ⁻¹ (p) 92	TRANS 92	VOL+ 94
t _Δ (x) 92	TRI 147	<i>von Klitzing</i> 120
t _Δ (x) 92	troz→g 92	<i>Voyagers</i> 3, 17, 255, 265, B-5
T.RES 76, 91, 92	trz→kg 92	
t: 92, 146	ts _I →mℓ 92	
t ₂ 92	ts _{UK} →mℓ 92	
T ₀ 89, 121	ts _{US} →mℓ 92	
tan 90	<i>tumor</i> 125	
tanh 90	<u>TVM</u> 92, 144, 345	
<i>Taylor</i> 8	U→ 93, 147	
tb _I →mℓ 90	ULP 92	
tb _{UK} →mℓ 90	U _n 93	
Tb _{US} →mℓ 90	UNDO (🔄) 93	
TDISP 90	UNITV 93	
TDM24 90, 129	UNSIGN 93	
<u>TEST</u> 90, 147	<u>USB</u> 93, 131, 179, 251	
<i>Thomas K.</i> 342	<u>USER</u> 93, 115, 131	
TICKS 90		
		W ⁻¹ 95
		W→hp _{UK} 96
		W→hp _E 96
		W→hp _M 96
		<i>Walter</i> → <i>Kolb</i>
		WDAY 94
		Weibl ⁻¹ 95
		Weibl _Δ 95
		Weibl _Δ 95
		Weibl: 95, 146
		Weibl _p 95
		<i>Weibull</i> 95, 280
		Wh→J 95

WHO 95

William → *Kahan*

W_m 95

WolframAlpha 199

Woodstocks 3,
255

word size 192

world history 127

W_p 95

WP 31S 230, 22,
31, 32, 33

WP 34S 15, 17, 72,
95, 138, 190, 199,
207, 215, 225,
230, 260, 267,
337, 353, B-4,
10, 17, 22, 31

WP 43S B-4, 6

WRITEP 95

WSIZE 95

X 96

$\hat{x}$ 96

$\bar{x}$ 96

$\bar{x}_G$ 96

$\bar{x}_H$ 96

$\bar{x}_{RMS}$ 97

$\bar{x}_w$ 97

$x < ?$ 98

$x = ?$ 98

$x \neq ?$ 99

$x = -0?$ 98

$x = +0?$ 98

$x > ?$ 98

$x \approx ?$ 98

$x \leq ?$ 98

$x \geq ?$ 99

x! 97

X.FN 97, 148

$x:$ 97

$x > ?$ 99, 174, 222

$x \rightarrow \text{DATE}$ 98

$x \rightarrow \alpha$ 98

x 98

x^2 96

x^3 96

XEQ.SKIP 96

$x_{\max}$ 97

$x_{\min}$ 97

XNOR 97

XOR 97

X-ray 125

Y 99

$\hat{y}$ 99

Y.MD 99

y 99

y^x 99

$y_d \rightarrow m$ 99

YEAR 99

$year \rightarrow s$ 99

$y \rightarrow m$ 99

YMD 99, 129

$Y_y(x)$ 99

Z 99

$Z(\varphi, m)$ 327

z 100

Z_0 99, 121

$zh \rightarrow m$ 99

ZOOM 23, 100

α 100, 121

$\alpha \cdot (\bullet)$ 101, 151

A...Ω 101, 150

α.FN 101, 150

$\alpha \rightarrow x$ 101

αCAP 100, 131

αINTL (A) 100, 150

αLENG 100

αMATH (≈) 100,
150

αPOS 100

αRL 100

αRR 100

αSL 100

αSR 101

$\beta(x, y)$ 101, 175,
229

γ 101, 121

$\Gamma(x)$ 101

γ_{EM} 101, 121

γ_p 101, 121

Γ_p 101

Γ_q 101

γ_{xy} 101

Γ_{xy} 101

Δ% 102

Δx 102
 Δv_{cs} 102, 122

ϵ 102
 ϵ_0 102, 122
 ϵ_m 102
 ϵ_p 102

$\zeta(x)$ 102
 $Z(\varphi, m)$ 102

λ_c 103, 122
 λ_{cn} 103, 122
 λ_{cp} 103, 122

μ_0 103, 122
 μ_B 103, 122
 μ_e 103, 122
 μ_e/μ_B 103, 122
 μ_n 103, 122
 μ_p 103, 122
 μ_u 103, 122
 μ_ν 103, 123

π 103, 123
 $\Pi(n, m)$ 103, 327
 Π_n 103
 σ 103

Σ 103, 151
 Σ^- 105
 $\Sigma 1/x$ 104
 $\Sigma 1/x^2$ 104
 $\Sigma 1/y$ 104
 $\Sigma 1/y^2$ 104
 Σ^+ 105
 σ_B 103, 123
 $\Sigma \ln^2 x$ 104
 $\Sigma \ln^2 y$ 104
 $\Sigma \ln x$ 104
 $\Sigma \ln xy$ 104
 $\Sigma \ln y$ 104
 $\Sigma \ln y/x$ 104
 Σ_n 104
 σ_ω 104
 Σx 89, 104, B-11
 $\Sigma x/y$ 105
 Σx^2 104
 $\Sigma x^2/y$ 104
 $\Sigma x^2 \ln y$ 104
 $\Sigma x^2 y$ 104
 Σx^3 104
 Σx^4 104
 $\Sigma x \ln y$ 104
 Σxy 104
 Σy 89, 105, B-11

Σy^2 105
 $\Sigma y \ln x$ 105

Φ 105, 123
 Φ_0 105, 123

χ^2 : 146
 $\chi^2_{\Delta}(x)$ 105
 $\chi^2_{\Delta}(x)$ 105
 χ^2 : 105
 $\chi^2_p(x)$ 105

$\psi(u, m)$ 106, 327

ω 106, 123, 178

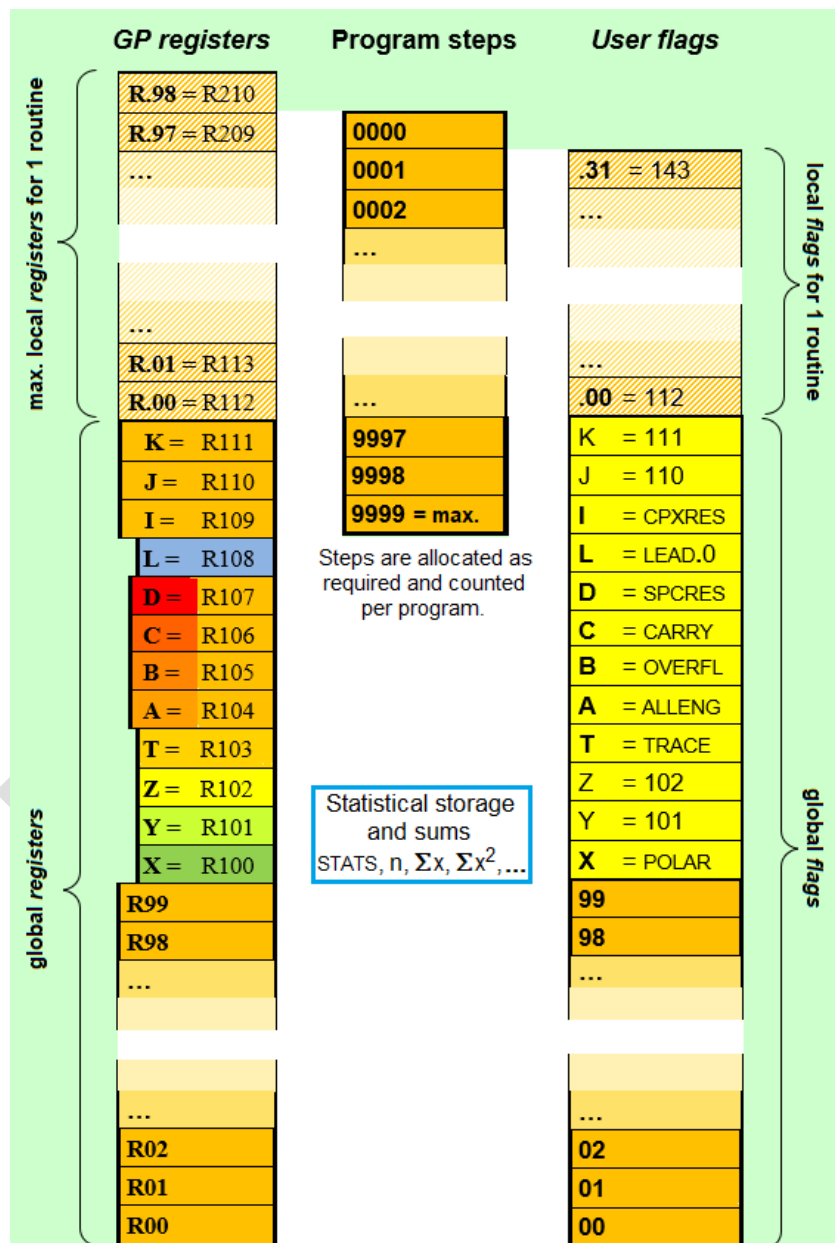
CC 15, 23, 33,
76, 169, B-2

ENTER↑ 15, 165f,
168-72, 210, 240,
243f, 246, 248
260f, 269, B-1,
4f, 22, 33, 38f

EXIT 15, 23, 38f,
51, 67, 87, 113,
135, 138, 140,
166, 171, 213,
243f, 246, 248,
254, 272, 336,
354, B-1, 33

WP43 QUICK REFERENCE GUIDE

ADDRESS SPACE



The stack is your workspace for calculations. Choose a *stack* of 4 (**X**, **Y**, **Z**, and **T**) or 8 *registers* (**X**, **Y**, **Z**, **T**, **A**, **B**, **C**, and **D**; startup default). Last *x* is saved in **L**. Each *stack register* (e.g., **Y**) may contain any type of data (*y*).

General purpose (GP) registers: There are 100 numbered global *GP registers* (**R00** ... **R99**). Furthermore, there are **I**, **J**, and **K** serving special purposes in matrix handling (→ Q-8), probability distributions (→ Q-10), and programming but may be used globally otherwise. Also **A** ... **D** may be *GP registers* unless being part of the *stack*. Each *GP register* may contain any type of data.

[STO] *nn* stores a copy of *x* in **Rnn**,

[RCL] *nn* recalls a copy of *rnn* into **X**, and

[x↔] *nn* swaps *x* and *rnn*.

Variables are named storage locations. Each may contain any type of data. E.g., for storing *x* in a new *variable* named **XYZ**, type **[STO]** **[α]** **XYZ** **[ENTER]**.

Names of variables shall be unique and ≤7 *characters* long. User-created *names* (also for *menus* and *global labels*) shall start with a letter and must not contain +, -, ×, /, :, ^, (,), =, !, ;, |, a comma, or a space.

Once created, *names* (also of commands) are case sensitive but level insensitive, i.e. **B_n*** accesses **B_n***, too (easing input in equations or searches).

Flags: There are 112 global *user flags* and 43 named *system flags* (→ Q-16).

A program consists of ≥ 3 steps: **LBL** with a *global label*, ≥1 action steps, and **END**. Each program may contain subroutines (containing ≥ 3 steps each: **LBL**, ≥1 action steps, and **RTN**). Subroutines may be nested. Each (sub-) routine may allocate 32 local user flags and ≤ 99 *local registers*. Steps are counted per program (→ Q-1 and Q-5).

Available memory: **[INFO]** **?MEM** and **[FLAG]** **STATUS** show the amount of free memory. Use **CLP** for removing programs or **DELITM** for deleting *variables* to free memory no longer needed. Also clearing large *register* contents helps.

DATA TYPES

Long integers: Any pure numeric input (without **[.]**, **[E]**, **[CC]**, or **[#]**) is taken as a *long integer* of base 10. Such numbers may be up to 1000 digits long.

Short integers: Any pure numeric input followed by **[#]** and a *base* (2 ... 16) is read as a *short integer* of the base specified. **[D]** and **[H]** are shortcuts for bases 10 and 16. A *short integer* may occupy ≤ 64 bits (→ Q-14).

Real numbers: Any number entered using one **[.]** and/or **[E]** is taken as a *real number*. Each may carry 34 digits precision.

Complex numbers: A *complex number* consists of two *real numbers* combined:

- 1) its *real* and imaginary part like $1.23 - i \times 4.56$ in rectangular notation (CF POLAR (or CF X) and press **1.23** **CC** **4.56** **+/-** **(ENTER†)**) or
- 2) its magnitude and phase like $7.89 \angle 120^\circ$ (other ADMs may be chosen) in polar notation (SF POLAR (or SF X) and press **7.89** **CC** **120** **(ENTER†)**).

Angles: Any *real* input followed by **d.ms** is interpreted as an *angle* in *sexagesimal degrees*. *Angles* may be entered and displayed as well in *decimal degrees*, *radians*, *multiples of π* , *mils*, or *gradians (gon)*. Select the appropriate *angular display mode (ADM)* via **MODE** ($\rightarrow$ overleaf).

Times: Any *real* input followed by **h.ms** is taken as a sexagesimal *time*. It will be displayed like 23:45:43.210 9 with as many *hours* as necessary and as many decimals of *seconds* as chosen.

Dates: Any *real* input followed by **.d** is read as a *date* in the format chosen (yyyy.mmdd for YMD or dd.mmyyyy for DMY or mm.ddyyyy for MDY). It will be displayed like 2021-12-02 or 02.12.2021 or 12/02/2021, respectively. A time of day may be appended, and *times* may be added to *dates*.

Vectors and matrices $\rightarrow$ Q-8.

Text strings: **α** opens *alpha input mode (AIM)*. Data entered in AIM become a *text string* when closed (unless they are function parameters). Latin letters A...Z are on the keyboard, more are in **A**. Greek letters are accessed via **□** plus the corresponding Latin letter ($\rightarrow$ calculator backside). Turn to lower case by **▼** and to upper by **▲** for all letters.

□ plus one of the keys **+**, **-**, **×**, **/**, **+/-**, **e^x**, **√x**, **□**, and **0 ... 9** will enter the respective *character*. Further symbols are in **□-** and **□□**.

□ + **R↓** makes the subsequent *character* entered a subscript, and

□ + **E** makes it a superscript, if applicable.

Equations are entered in **EQN**. They may be labelled, follow algebraic notation, and may contain =, (,), ^, and a choice of monadic and dyadic functions. Such equations may be solved automatically for any *variable*, integrated, and differentiated ($\rightarrow$ Q-12).

USING MENUS

A *menu* defines the functionalities of the keys **F1** – **F6** by displaying up to three *labels* above each such key. To access a *label* in the lowest row, press the corresponding key. To access a *label* in 2nd or 3rd row, press **□** or **□** before. A dashed line on top of the current *menu view* indicates a *multi-view menu*; then **▲** or **▼** will show additional *views*. *Menu names* are printed underlined in running text.

Each *menu view* stays displayed until **EXIT**ed or another *menu (view)* is called.

MODES

MODE

DMCP						
	GRAD	RM		SETSIG	DENMAX	
SF	DEG	RAD	MIL	MUL π	CF	

- SF n , CF n** Sets (or clears) the flag specified.
- DEG** Selects decimal *degrees*
- RAD** Selects *radians*
- MUL π** Selects *multiples of π*
- MIL** Selects *mils*
- GRAD** Selects *grades*, a.k.a. *gon*, } as angular display mode.
- RM n** Sets rounding mode.
- SETSIG n** Sets calculator precision (1 ... 34 significant digits, 0 = max.).
- DENMAX n** Sets the maximum denominator for calculating with fractions.
- DMCP** Exits WP43 to the *DMCP menu* for file transfer or firmware update.

DISPLAY FORMATS

DISP

GAP	HIDE	?HIDE	RANGE	?RANGE	DSTACK	
CHINA	EUROPE	INDIA	JAPAN	UK	USA	
SDL	SDR			RDP	RSD	
FIX	SCI	ENG	ALL	ROUNDI	ROUND	

- FIX n** Fixed point display of n decimals.
- SCI n , ENG n** Scientific (or engineering) format, SCI with n decimals.
- ALL n** Displays all digits as far as possible with max. n decimal zeros.
- ROUND** Rounds a *time*, *real*, or *complex x* to the current display format.
- ROUNDI** Rounds x 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) in x 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.
- HIDE n** Set to hide all results $< 10^{-n}$, i.e. display them as 'zero'.
- RANGE n** Sets the maximum exponent to be displayed for *real numbers*.
- ?HIDE, ?RANGE** Returns the current HIDE or RANGE setting.
- DSTACK n** Sets the number of *stack registers* to be displayed (1 ... 4).

PROGRAMMING

Program Entry and Navigation

	toggles <i>program entry mode</i> .
	moves the <i>program pointer</i> to free program space.
	moves it to step number <i>nnnn</i> within the <i>current program</i> .
	moves it to top of <i>current program</i> or to previous program.
	moves it to top of next program.
	moves it to previous step (you can use  or  unless
	moves it to next step a <i>multi-view menu</i> is displayed).
	removes the <i>current program step</i> entirely.
	exits <i>program entry mode</i> unless a <i>menu</i> is displayed.

Labels

A program *label* is a tag used to identify a program or a section within a program. Each program must begin with a *global label* (→ Q-2).

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

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

Local Registers

LocR *n* allocates *n* local registers (≤99). 32 local flags come with them. Local data are valid in *current (sub-) routine* only.

Tests (Do if True, Skip if False)

When a binary test step is executed and the result is **true** then the step immediately following will be executed ("do if true"); if the test result is **false**, the step following the test step will be skipped.

Looping

ISE, ISG, ISZ, DSE, DSL, and DSZ (in **LOOP**) control looping. **ISE, ISG, DSE, and DSL** access a *variable or register* containing a *loop control number* in the format *ccccc.fffii* with *ccccc* = current counter value, *fff* = final value, and *ii* = increment (or decrement) step 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 (often a branch to the top of the loop); compare the backside of your **WP43**. The example in the box counts from 1 to 52 by 3 (executing the loop 18 times) and then beeps.

```
...  
1.05203  
STO 'Count'  
LBL 01  
... { loop }  
ISG 'Count'  
GTO 01  
BEEP  
...
```

Using a Variable Menu

A *menu of variables* may be displayed by the *Solver* or *Integrator* (→ Q-12f) or by **VARMNU** within a program. Each *label* in this *menu* represents a *variable*. While this *menu* is displayed, you can...

... store a value into a *variable*: Key in the value, then press the resp. *softkey*.

... recall the contents of a *variable*: Press **RCL**, then the respective *softkey*.

... view the contents of a *variable* without recalling: Press **VIEW**, then the *softkey*.

... select a *variable*: Press the respective *softkey* without typing 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*.

EXECUTING FUNCTIONS AND PROGRAMS

Every function or program can be executed via **XEQ** **α** *name* **ENTER↑** where **name** is the function *name* or the program *label*. Case matters. If **name** is not unique, the *global label* closest to the permanent end (**END.**) has precedence. If **name** is a *local label*, WP43 searches for it in the current program only.

Smart program menu: **XEQ** **PROG** displays all programs (actually: all *global labels*) defined at execution time. Specify the required program by pressing the corresponding *softkey*.

Single stepping: To execute the current program step, press **⇓** (or **▼** if no *multi-view menu* is displayed). Press **⇑** (or **▲**) for browsing backwards.

Run/Stop: Press **R/S** to run the current program starting with the current step or to stop the running program after the current step is executed.

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

Specifying Function Parameters

Numeric parameters: Functions accepting trailing numeric parameters will prompt you with an underscore for each digit expected. Just enter the digits; terminate shorter input using **ENTER↑**. Alphanumeric parameters are accepted by many functions as well.

Registers and stack: Rather than keying in an actual parameter, you can specify the *variable* or *register containing* the parameter. Any function accepting a *GP register* will accept a *stack register* as well. Press the corresponding *softkey* for **X ... T** or one of the keys **A ... D**, if applicable.

If the wanted parameter is a *variable* already existing, pick it from **VAR**. If a destination doesn't exist yet, type it (e.g., for creating a *variable* **ABC** type **(STO) (α) ABC (ENTER↑)**).

Indirect addressing: Instead of keying in an actual *variable/register* address, you can specify the *variable* or *register* containing its address – just press the *softkey* →. E.g., to display the contents of the *variable* or *register* specified in **R12**, key in **(VIEW) → 12**. This works with lettered and *stack registers* as well.

For every command **XY** requiring one trailing parameter, **XY → X** will let **XY** take its parameter from **X** instead.

CLEARING AND DELETING

	RESET		CLKEYS	DelITM	CLMENU	
(CLR)	CLall	CLBKUP	DelVall		DelMall	CLLCD
	CLREGS	CLPall	CLVall	CLFall	CLMall	CLSTK
	CLΣ	CLP	CLCVAR	CF	CLMy...	CLX

CLΣ	Clears all statistical data (also accessible via <u>FIT</u> and <u>STAT</u>).
CLP	Clears (deletes) the <i>current program</i> .
CLCVAR	Clears (resets) all <i>variables</i> used in the <i>current program</i> .
CF <i>n</i>	Clears <i>flag n</i> .
CLMy...	<i>Submenu</i> to clear (reset) <u>MyMENU</u> , <u>MyPFN</u> , or <u>Myα</u> .
CLX	Clears <i>stack register X</i> .
CLREGS	Clears all <i>registers</i> (except the <i>stack</i> and statistical data).
CLPall	Clears (deletes) all programs in <i>RAM</i> .
CLVall, CLMall	Clears all user <i>variables</i> or <i>menus</i> .
CLFall	Clears all numbered user <i>flags</i> .
CLSTK	Clears the entire <i>stack</i> (i.e. resets all its <i>registers</i> to 0.).
CLall	Clears almost everything but the modes set.
CLBKUP	Clears (deletes) the backup file.
DelVall, DelMall	Delete all user <i>variables</i> or <i>menus</i> .
CLLCD	Clears the <i>LCD</i> above and to the right of pixel (<i>x</i> ; <i>y</i>).
RESET	Resets your WP43 to <i>startup default configuration</i> (<i>SDC</i>).
CLKEYS	Clears (deletes) all key assignments.
DelITM	Deletes a user-created <i>item</i> , → overleaf.
CLMENU	Clears the <i>programmable menu</i> .

Operations clearing programs or deleting other *items* will free space in memory.

DelITM	VARS	...	☐	Deletes the user <i>variable</i> selected.	} Use the applicable <i>softkey</i>
DelITM	MENUS	...	☐	Deletes the user <i>menu</i> selected.	
DelITM	PROGS	...	☐	Deletes the user program selected.	

MATRIX OPERATIONS

A $m \times n$ matrix is a rectangular array featuring m rows and n columns of *real* or *complex numbers* (a.k.a. *matrix elements*). A vector is a $1 \times n$ or $m \times 1$ matrix.

	CMIN	CMAx	M.QR	FIND	EIGVAL	EIGVEC	
	RNORM	RSUM	M.LU	?DIM	C↔C	R↔R	
	I+	I-	STOIJ	RCLIJ	J-	J+	
MATX	ENORM	V _∠	STOEL	RCLEL	PUTM	GETM	
	dot	cross	UNITV	DIM	INDEX	EDITN	
	NEW	[M] ⁻¹	M	[M] ^T	SIM EQ	EDIT	

NEW	Creates a new matrix in X (→ Q-9).
[M] ⁻¹	Inverts the matrix x .
M	Returns the determinant of the matrix x .
[M] ^T	Transposes the matrix x .
SIM EQ n	Solves a system of n simultaneous linear equations (→ Q-9).
EDIT	Edits the matrix x (→ Q-9).
dot	Returns the dot product of the matrices (or vectors) y and x .
cross	Returns the cross product of the (2D or 3D) vectors y and x .
UNITV	Returns the unit vector of x .
DIM	Creates a new named matrix (→ Q-9).
INDEX, EDITN	Indexes (or edits) a named matrix (→ Q-9).
ENORM	Returns the <i>Euclidean</i> norm of the matrix x .
V _∠	Computes the <i>angle</i> between two vectors y and x .
STOEL	Stores x into the current element a_{ij} of the indexed matrix (<i>IM</i>).
RCLEL	Recalls the current element a_{ij} of the <i>IM</i> into X .
PUTM	Puts the matrix x into the <i>IM</i> , starting with target element a_{ij} .
GETM	Gets a $IP(y) \times IP(x)$ sub-matrix out of the <i>IM</i> into X , starting with a_{ij} .
I+, I-, J+, J-	Increments or decrements the row or column pointer for the <i>IM</i> .
RCLIJ	Recalls the values of the row and column pointers into Y and X .
STOIJ	Sets the row pointer i to $IP(y)$ and the column pointer j to $IP(x)$.
R↔R, C↔C	Swaps row (column) $IP(x)$ and row (column) $IP(y)$ of the <i>IM</i> .
CMIN, CMAx	Returns the smallest (greatest) element in a column of the <i>IM</i> .
FIND	Searches the specified value in the <i>IM</i> .

RNORM, RSUM	Returns the row norm (or row sum) of the matrix x .
M.LU, M.QR	Computes the LU (or QR) decomposition of the matrix x .
EIGVAL, EIGVEC	Computes the eigenvalues (or eigenvectors) of the matrix x .
?DIM	Returns the dimensions of the matrix x (rows in Y , columns in X).

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

- **MATX NEW** for a matrix in X or
- **MATX DIM** r for a matrix in the register or variable specified or
- **MATX DIM** α **name** **ENTER** for a matrix in the variable specified. If the target variable already exists, DIM will re-dimension it.

To edit the matrix in X , use **MATX EDIT** to start the Matrix Editor (**ME**).

To edit a named matrix, use **MATX EDITN VAR** and press the respect. *softkey*.

When a matrix is being edited it is called *indexed* (to index a matrix without editing, use INDEX). Whenever there is an indexed matrix (**IM**), two pointers (in **I** and **J**) indicate the row and column of the current element ($\rightarrow$ Q-8).

If i and j are pointing to the last (bottom right) element in the **IM** and you press **→** then ...

- the pointers wrap around to the first element of the **IM** (if in *wrap mode* which is set automatically whenever you enter or exit the **ME**) or ...
- the **IM** grows by one row and the pointers move to the first element in the new row (if in *grow mode* – set **GROW** via the **ME menu**; reset by **WRAP**).

Matrix Arithmetic

+, **-**, **×**, and **/** work for matrices like they do for numbers if the matrix dimensions match. Instead, advanced functions (like **e^x**, **ln**, **10^x**, **lg**, **1/x**, **√x**, **x²**, trigonometric and hyperbolic functions) operate on all matrix elements individually ($\rightarrow$ OMS 3 and the IOI).

Whenever a matrix is involved in a mathematical operation with a *complex* object, the result will be a *complex matrix*.

Solving a system of linear equations $[A] \vec{X} = \vec{B}$

1. Enter **MATX SIM EQ** n with n being the number of unknowns. Your WP43 will create or re-dimension the matrix variables **Mat.A** and **Mat.B**.
2. Press **Mat A**; fill the matrix; eventually press **EXIT**.
3. Press **Mat B**; fill the vector; eventually press **EXIT**.
4. Press **Mat X** to compute and return the solution vector **Mat.X**.

PROBABILITY

	RAN#	SEED	RANI#		$\ln \Gamma$	$\Gamma(x)$	
(PROB)		NBin:	Geom:	Hyper:	Binom:	Poiss:	
	LgNrm:	Cauch:		Expon:	Logis:	Weibl:	
	Norml:	t:	C_{yx}	P_{yx}	F:	χ^2 :	

Submenu e.g.

Norml:	Norml _p		Norml _Δ	Norml _Δ		Norml ⁻¹
--------	--------------------	--	--------------------	--------------------	--	---------------------

Norml_p denotes the *probability density function*, **Norml_Δ** the *cumulated distribution function*, **Norml_Δ** the *error probability*, and **Norml⁻¹** the *quantile function* (this naming scheme holds also for the other *distributions*, except χ^2).

C_{yx} , **P_{yx}** Returns the number of possible combinations (or permutations a.k.a. arrangements) of x objects taken out of a set of y objects.

RAN# Returns a pseudo random *real* $\in [0; 1)$.

SEED Stores a seed $\in [0; 1)$ for **RAN#**. **SEED 0** takes it from the clock.

RANI# Returns a pseudo random *integer* $\in [IP(x); IP(y)]$.

$\Gamma(x)$, **$\ln \Gamma$** Returns the value of the *gamma function* (or its natural logarithm).

5 discrete and 9 continuous distributions ($d.$) are supported:

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 (= **dof₁**), $j =$ **dof₂**)

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 Bin. d.* ($i = p_0$, $j = n$)

Norml: *Normal d.* ($i = \mu$, $j = \sigma$. *Standard Gaussian d.*: $i = 0$, $j = 1$)

Poiss: *Poisson d.* ($i = n \times p_0 = \lambda =$ Poisson parameter)

t: *Student's t d.* ($i =$ **dof**)

Weibl: *Weibull d.* ($i = b =$ shape, $j = T =$ characteristic lifetime)

χ^2 : *Chi-square d.* ($i =$ **dof**)

Store the required parameters in **I**, **J**, and **K** as specified above; the remaining parameter must be given in **X** before calling the respective function – note *quantile functions* require a probability input ($0. \leq x \leq 1.$).

STATISTICS

Statistical data are collected in the $n \times 2$ *real* matrix STATS, separate from user-generated matrices.

Clear STATS by **STAT** **CLΣ** (**FIT** **CLΣ**, **CLR** **CLΣ**) before a new statistical analysis.

Accumulate the data:

- for each individual data value: **x-value** **STAT** **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), put this matrix in **X** and then press **Σ+**

To undo input errors or remove erroneous data,

- use  or **STAT** **Σ-** to delete the last data point entered from STATS or...
- edit STATS using the *Matrix Editor* (→ Q-8).

Data Analysis

STAT			$\bar{x}_H$				
			$\bar{x}_{RMS}$	x_{min}	x_{max}	Δx	
			MEDIAN	MAD	Q_3-Q_1	%ile	
		CLΣ	$\bar{x}_G$	ϵ	ϵ_p	ϵ_m	FIT
		Σ-	$\bar{x}_w$	s_w	σ_w	s_{mw}	HISTOG
		Σ+	$\bar{x}$	s	σ	s_m	SUM

SUM Recalls Σy and Σx . – Note all following operate on $\{x\}$ and $\{y\}$.

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

$\bar{x}_w$, s_w , σ_w , s_{mw} Same for weighted data (→ above).

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

HISTOG Submenu to prepare and plot histograms (→ *OMS* 2).

FIT Pointer to the *menu* overleaf.

MEDIAN, **MAD** Median and median absolute deviation.

%ile, Q_3-Q_1 Percentile, distance between 75%- and 25%-percentiles.

x_{min} , x_{max} , Δx Min. and max. values in the data sets and their difference.

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

FIT	GaussF	CauchF		BestF		
	ParabF	HypF	RootF			
	LinF	ExpF	LogF	PowerF		OrthoF
	CLΣ	ASSESS				PLOT
	Σ-		s _{xy}	cov		HISTOG
	Σ+	L.R.	s(a)	r	$\hat{x}$	$\hat{y}$

L.R. Computes the parameters a_0 and a_1 (and a_2 , if applicable,) of the regression for the fit model (FM) selected (→ below).

s(a) Returns the standard errors of a_0 and a_1 for the green FMs.

r Returns the correlation coefficient between {x} and {y}.

$\hat{x}$, $\hat{y}$ Returns the forecast for x (or y) according to the FM selected.

s_{xy}, cov Returns the sample (or population) covariance.

ASSESS Plots the data points together with the FM curve(s) selected:

CauchF Cauchy peak FM: $y = 1/[a_0(x + a_1)^2 + a_2]$.

ExpF Exponential FM: $\ln(y) = \ln(a_0) + a_1x$ or $y = a_0e^{a_1x}$.

GaussF Gauss peak FM: $y = a_0 e^{\frac{(x-a_1)^2}{a_2}}$.

HypF Hyperbolic FM: $y = 1/(a_0 + a_1x)$.

LinF Linear FM: $y = a_0 + a_1x$.

LogF Logarithmic FM: $y = a_0 + a_1\ln(x)$.

ParabF Parabolic FM: $y = a_0 + a_1x + a_2x^2$.

PowerF Power FM: $\ln(y) = \ln(a_0) + a_1\ln(x)$ or $y = a_0x^{a_1}$.

RootF Root FM: $y = a_0a_1^{1/x}$.

BestF n Picks the FM returning the best r^2 of all FMs allowed ($n = FM$ code).

OrthoF Works like LinF but assumes equal errors in x and y. Note OrthoF is not checked by BestF but used in PLOT.

PLOT Plots the measured data points to analyze and qualify a measuring system. With ≥ 30 points, its standard error s_{mi} can be determined (→ OMS 2).

ADVANCED OPERATIONS

EQN is for interactive editing, storing, solving, integrating, and differentiating,

ADV for programmed summing, multiplying, solving, integrating, differentiating.

Interactive Operations on Equations

EQN	...					
	DELETE					
	NEW	EDIT	f''	f'	∫f	Solver

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 all existing equations, press **▲** or **▼**.

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 therein (maybe more).

Using Advanced Operations in Programs

(ADV)

PGMSLV	SLVC	f''(x)		PGMINT	
SOLVE	SLVQ	f'(x)	Π_n	Σ_n	∫fdx

SLVQ (or **SLVC**) solves the quadratic (or cubic) equation $ax^2 + bx + c = 0$ (or $ax^3 + bx^2 + cx + d = 0$) with all its parameters on the input *stack* [(d,) c, b, a, ...]. **SLVQ** will either return two *real* or two *conjugate complex* roots; **SLVC** will return three roots and the *discriminant* (→ IOI).

Π_n label calculates the product ...
Σ_n label calculates the sum ...

} ... of the terms given by the routine specified, using the loop control number in **X** (→ Q-5).

SOLVE var solves for an unknown *variable* in an expression $f(x_1, x_2, ...) = 0$ when values for all the other *variables* are given. The expression shall be written as a program (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 $f(x_1, x_2, ...) = 0$ is fulfilled. Recall the *variables* of the expression as they are required and calculate **f**.
- **AB** must logically end with **RTN**.

Then write a 2nd program (called **CD**, for example) calling the *Solver*. At the position where you need the expression solved, press **(ADV)**, then...

1. Press **PGMSLV** and specify **AB**.
2. Store a value into each known *variable*, e.g., using **ST0**. 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 (→ IOI).

f'(x) (or **f''(x)**) computes the 1st (or 2nd) derivative of **f(x)** at location **x**. A program (called **EF**, for example) shall evaluate **f(x)**; it must begin with a *global label* and take care of all *variables* used.

Then write a 2nd program calling the derivation (call it **GH**, for example).

1. Store a value into each *variable* to 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 respective derivative will be returned in **X**.

∫fd var computes a definite integral. The integrand $f(x)$ shall be written as a program (called **IJ**, for example); it must begin with a *global label*, recall all integration constants used, and evaluate $f(x)$.

Then write a 2nd program (call it **KL**, for example) calling the *Integrator*. At the position where you need the integral, press **ADV**, then...

1. Press **PGMINT** and specify **IJ**.
2. Press **∫fdx**. A *submenu* will open.
3. Store a value into each *variable* that shall remain constant under integration, e.g., using **ST0**.
4. Store the lower (**↓Lim**) and upper limit (**↑Lim**) of integration and the **ACC**uracy factor.
5. Press **∫** and specify the *variable of integration*.

When running **KL** later on, it will return the integral in **X** and its uncertainty in **Y**.

OPERATIONS ON SHORT INTEGERS

	→BIN	→OCT			→DEC	→HEX	
	1COMPL	2COMPL	UNSIGN	SIGNMT		WSIZE	
INTS	DBL /	DBLR	DBL ×	^MOD	ceil	GCD	
	IDIV	RMD	MOD	×MOD	floor	LCM	
	A	B	C	D	E	F	

A ... F Digits for input of *short integers* of bases >10.

IDIV Integer divide – works also for *reals* (**R**) and *long integers* (**Z**).

RMD (**R**, **Z**) Remainder.

MOD (**R**, **Z**) Modulo.

×MOD (**R**, **Z**) Returns $(z \cdot y) \bmod x$.

^MOD (**R**, **Z**) Returns $(z^y) \bmod x$.

floor, **ceil** (**R**) Returns the greatest *long integer* $\leq x$ (or the smallest $\geq x$).

LCM, **GCD** (**Z**) Returns the least common multiple (or the greatest common divisor) of x and y .

DBL /, **DBLR**, **DBL ×** Double *word* length commands for division, remainder, and multiplication (→ *IOI*).

1COMPL, 2COMPL Sets 1's (or 2's) complement mode.
 UNSIGN Sets unsigned mode.
 SIGNMT Sets sign-and-mantissa mode.
 WSIZE *n* Sets the *word* size to 1 ... 64 *bits*.
 →BIN, →OCT, →DEC, →HEX Convert the display of *x* directly.

	1COMPL	2COMPL	UNSIGN	SIGNMT		WSIZE	
	LJ				RJ	ASR	
	SL	RL	RLC	RRC	RR	SR	
	A	B	C	D	E	F	
BIT	SB	BS?	#B	FB	BC?	CB	
	NAND	NOR	XNOR	MIRROR			
	AND	OR	XOR	NOT	MASKL	MASKR	

AND, OR, XOR, NAND, NOR, XNOR Boole's binary operators.
 NOT Flips every bit in *x*.
 MASKL, MASKR *n* Creates a mask of *n* bits on the left (or right) side.
 MIRROR Reflects all bits.
 SB, FB, CB *n* Sets, flips, or clears bit *#n* in *x* (counting starts with #0).
 BS?, BC? *n* Checks if bit *#n* in *x* is set (or clear).
 #B Returns the total number of bits set in *x*.
 SL, SR *n* Shifts *x* to the left (or right) by *n* places.
 ASR *n* Arithmetically shifts *x* to the right by *n* places, i.e. divides *x* by 2^n .
 RL, RR *n* Rotates *x* to the left (or right) by *n* places.
 RLC, RRC *n* Rotates *x* to the left (or right) by *n* places through CARRY.
 LJ, RJ Adjusts the bits set in *x* to the left (or right).

OPERATIONS ON TEXT STRINGS

Use ⊕ to connect strings. Then *x* will be appended to *y*; with numeric data in *X* or *Y*, their current display format is taken into account.

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

x→α *d* Converts the code *x* to the corresponding *character* and appends it to the string in *d*.
 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*.

$\alpha \rightarrow x$ s	Removes the 1 st <i>character</i> from s and pushes its code on the <i>stack</i> .
αLENG s	Pushes the length of the string in s on the <i>stack</i> .
αPOS s	Returns the position where substring x begins in the string s .
FBR	Shows all <i>characters</i> defined in both fonts.

USER WRITEABLE SYSTEM FLAGS

The *startup default* state of each *flag* is indicated (**FLAG** means 'FLAG clear'). **STATUS** shows all *flags* currently set, some *flags* appear also in the *status bar*.

ALLENG (A)	ALL overflows to ENG , else to SCI
ALPHA	WP43 in AIM
ALP.IN	Alphanumeric input received, else NUM.IN
AUTOFF	Turns WP43 off after 10 minutes inactivity, else stays on
AUTXEQ	Executes current program automatically when WP43 is turned on
CARRY (C)	For <i>short integer</i> manipulation
GPXRES (I)	Allows <i>complex</i> results for <i>real</i> input
DECIM.	Displays decimal points, else commas
DENANY	Allows for any denominator up to D.MAX
DENFIX	Fixed denominator = D.MAX , else factors of D.MAX are allowed
FASTFN	Faster function computation
FRACT	Set for fraction display, clear for decimal display
FRGSRN	Verbose fraction display showing <i>register names</i>
IGN1ER	Ignores one error
LEAD-0 (L)	Displays leading zeros for <i>short integers</i>
MULT×	Displays multiplication crosses, else points
NUM.IN	Numeric input received, else ALP.IN
OVERFL (B)	For <i>short integer</i> manipulation
POLAR (X)	Polar notation for <i>complex numbers</i> , else rectangular
PROPRF	For FRACT set: displays proper fractions, else improper
QUIET	Disables the beeper
SPGRES (D)	Allows for special <i>real</i> results, i.e. $\pm\infty$ and NaN
SSIZE8	Sets <i>stack</i> size to 8 <i>registers</i> , else 4
TDM24	Sets <i>time</i> display to 24 hours, else 12
TRACE (T)	Print line is in tracing mode
USER	User mode

Copyright © 2017 - 2026 Walter Bonin, Radebeul, Germany

BACKGROUND FACTS AND CONSIDERATIONS

πάντα ῥεῖ

(φησὶν Ἡράκλειτον τὸν Ἐφέσιον φάναι)

This section is for recording and explaining some of the boundary conditions considered and settings chosen for your *WP43* in the course of this project. It is not necessary for operating your *WP43* but may foster understanding, and also a bit of the product philosophy may be found here.

WP43 is a *Gesamtkunstwerk*. Creating it, our primary goals were and are

1. Coherence,
2. Internationality (using international symbols instead of national abbreviations wherever possible (e.g., $\frac{\pi}{L}$ instead of $\frac{C}{HS}$), complying to *ISO* standards, etc. Alas, didn't succeed with $\frac{L}{}$ instead of $\frac{\pi}{\div}$),
3. Reliability, and
4. Precision and accuracy, in this order.

If in doubt, the *WP43* is programmed to be rather strict than lenient. Mathematics is a science allowing for much freedom: you may define a lot – but once you've specified something, you shall stick to this definition. And at the bottom line, a calculator is nothing more (nor less) than applied mathematics.

With this project developed in public for over a decade, some of our ideas diffused elsewhere. E.g., named *summation registers* (actually launched with the *WP 34S* already) are now found in *Plus42* as well. And *C47* and *R47* share at least 90% of their items with *WP43* which is expected due to their history.

Note there are no update marks in the following text since I presume you will read entirely what you are interested in.

Accessing Items

This is what a calculator is made for. But its hardware offers just 43 keys – for over 700 functions plus 79 constants. Subtract six for the *menu* keys. Thus, space for *primary functions* is quickly occupied – the respective functions are mostly set:

Obvious primary functionalities are the digits $\mathbf{0}$... $\mathbf{9}$, $\mathbf{.}$, $\mathbf{E}$, $\frac{\pi}{L}$, $\leftarrow$, $\mathbf{ENTER}$, $\mathbf{x} \mathbf{y}$, $\mathbf{+}$, $\mathbf{-}$, $\mathbf{\times}$, $\mathbf{/}$, $\mathbf{STO}$, $\mathbf{RCL}$, $\mathbf{\text{yellow}}$ and $\mathbf{\text{blue}}$, taking 24 keys. $\mathbf{\blacktriangle}$, $\mathbf{\blacktriangledown}$, and $\mathbf{EXIT}$ are required for menu handling, $\mathbf{XEQ}$ and $\mathbf{R/S}$ for programming. Hence, only 8 locations are left. Once you want to deal with

complex numbers seriously, you will need a primary **CC**.

Other functions may be debated: **R↓**, **1/x**, **y^x**, **x²**, **√x**, **e^x**, **ln**, **10^x**, **Ig**, **sin**, **cos**, and **tan** are the most popular – 12 candidates for 7 keys remaining. Either **x²** or **√x** shall be primary; and we can ditch **10^x** and **Ig** when we have **e^x** and **ln** primary. So the 6 functions **R↓**, **1/x**, **y^x**, **sin**, **cos**, and **tan** compete for the remaining 4 keys. We chose the first two and put the latter four (and their inverses) under two *primary menus*:

- **EXP** lets you access **y^x**, **log_xy**, **2^x**, **lb x**, **x²**, and **x³**, and
- **TRI** allows for calling **sin**, **cos**, **tan**, **arcsin**, **arccos**, and **arctan**

each by two keystrokes maximum, no more than needed if they were shifted labels on the keyboard.

10^x and **Ig** shall become secondary functions. But is it better to have them as shifted keyboard functions or as unshifted *softkeys*? No definite answer can be given here since it depends on the time **EXP** will stay open. We made **10^x** and **Ig** both -shifted keyboard functions since they may be missed if not seen immediately. (For similar reasons, **x≥**, **SF**, and **CF** are present on the keyboard although they are unshifted *softkeys* as well – there is no pressing need to remove them from the keyboard until today.)

WP43 features 35 *menus* on its keyboard. Of these, **CATALOG**, **CONST**, and the three alpha *character menus* follow special rules. Each of the other 30 *menus* offers 6 unshifted locations for *its* most popular *items*. Selecting these can be easy (like in **ADV**, **LOOP**, **STK**, or **TRI**) or more difficult (or even arbitrary) (like in **INFO**, **P.FN**, or **X.FN**). Though changing these is ‘only software’.

Summarizing the labels on the keyboard and unshifted *softkeys* (→ 142ff) we get all the following functions accessible by up to three keystrokes maximum (those of them asking for trailing parameters are printed on **light green**):

Keystroke(s)	Menu	Functions					
1	None	0 ... 9	.	E			
		EXIT	ENTER↑	x ² y	R↓	+/-	+
		-	x	/	STO	RCL	XEQ
		R/S	CC	1/x	ln	e ^x	√x
1 + 1	<u>EXP</u>	x ³	y ^x	log _x y	lb x	2 ^x	x ²
	<u>TRI</u>	sin	arcsin	cos	arccos	tan	arctan
2	None	a b/c	#	d.ms	.d	h.ms	α
		Ig	10 ^x	ASN	SAVE	RBR	VIEW
		R↑	x	↯	SNAP	USER	FILL

Keystroke(s)	Menu	Functions					
2		DROP↓	$\times \rightarrow$	$\Delta\%$	DSP	$\hookrightarrow$	
		MOD	GTO	LBL	RTN	$\times!$	$\odot$
		$R \leftarrow$	$\rightarrow P$	$\equiv \Delta$	$\equiv \nabla$	π	SHOW
		P/R	OFF	SF	CF	USB	EDIT
2 + 1	ADV	SOLVE	SLVQ	$f'(x)$	Π_n	Σ_n	$\int f dx$
	BIT	AND	OR	XOR	NOT	MASKL	MASKR
	CLK	DATE	$\rightarrow$ DATE	DATE $\rightarrow$	WDAY	TIME	$x \rightarrow$ DATE
	CLR	CLΣ	CLP	CLCVAR	CF	CLMy...	CLX
	CPX	dot	cross	UNITV	Re	conj	Re $\rightarrow$ Im
	DISP	FIX	SCI	ENG	ALL	ROUNDI	ROUND
	EQN	NEW	EDIT	f''	f'	$\int f$	Solver
	FIN	%	%MRR	%T	%Σ	%+MG	TVM
	FIT	Σ+	L.R.	s(a)	r	$\hat{x}$	$\hat{y}$
	FLAG	SF	FS?	FF	STATUS	FC?	CF
	INFO	?SSIZE	?MEM	?RM	?ISM	?WSIZE	?KTY
	INTS	A	B	C	D	E	F
	I/O	LoadAll	LoadPr	LoadEq	DUMP	WRITEP	SAVEST
	LOOP	DSE	DSZ	DSL	ISE	ISZ	ISG
	MATX	NEW	$[M]^{-1}$	$ M $	$[M]^T$	SIM EQ	EDIT
	MODE	SF	DEG	RAD	MIL	MULπ	CF
	PART	IP	FP	MANT	EXPT	sign	DECOMP
	PRINT	Σx	Σr	$\Sigma \Sigma$	ADV	LCD	PROG
	PROB	Norml:	t:	C_{yx}	P_{yx}	F:	χ^2 :
	P.FN	INPUT	MVAR	VarMNU	END	PAUSE	P.FN2
	STAT	Σ+	$\bar{x}$	s	σ	s_m	SUM
	STK	$\times \rightarrow$	$y \rightarrow$	$z \rightarrow$	$t \rightarrow$	$\rightarrow$	DROPy
	TEST	$x < ?$	$x \leq ?$	$x = ?$	$x \neq ?$	$x \geq ?$	$x > ?$
	U $\rightarrow$	E:	P:	Time:	F&p:	m:	x:
	X.FN	AGM	B_n	B_n^*	erf	Ellipt	Orthog

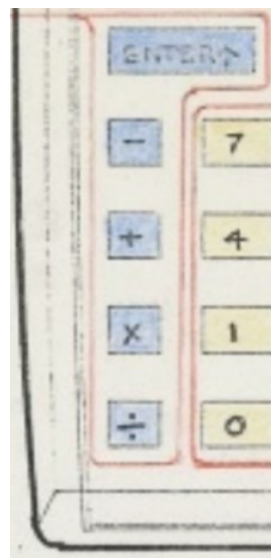
Keystroke(s)	Menu	Functions					
2 + 1	<u>α.FN</u>	$x \rightarrow \alpha$	αRL	αRR	αSL	αSR	$\alpha \rightarrow x$
	Σ	n	Σx	Σx^2	Σxy	Σy^2	Σy
	<u>4→</u>	→DEG	→RAD	→MULπ	→MIL	→GRAD	→D.MS
1 + 2	<u>EXP</u>	$\sqrt[3]{x}$	$\sqrt[y]{x}$		$\ln(1+x)$	$e^x - 1$	$\sqrt{1+x^2}$
		sinh	arsinh	cosh	arcosh	tanh	artanh
	<u>TRI</u>	sinc	sinc				atan2
		sinh	arsinh	cosh	arcosh	tanh	artanh

All the first 180 *menu functions* listed above can be accessed by a single keystroke if, when, and as long as their respective *menu view* is open. Thus, repeating any unshifted *softkey* as a shifted keyboard *label* is of limited value. It is done for **three functions** only, for reasons mentioned above.

After drawing many fictional calculators just for fun, I gained my real-world laying experience with the **WP 34S**. Based on this and expecting a forthcoming better display than the very limited LCD of the **HP-20b/30b**, I conducted an innocent little poll about the community's preferences for placing the four basic arithmetic operators on a hypothetical *RPN* portrait pocket calculator (→ <https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234505>) in 2012-11 – instead of resurrecting an old device or copying an existing layout, we wanted to create a really new, optimized pocket calculator based on our design experience, and we believed getting our product made by an *OEM* on a contract basis as soon as we were ready.

Said poll returned a majority of votes for placing the arithmetic keys beneath **ENTER↑** for ergonomic reasons – like they were on *HP's Classics* (→ the clip of an **HP-35** design sketch here, copied from <https://calc.fjk.ch/der-legendaere-hp-35/>).

The picture at the bottom of next page shows my last (internal) layout before the poll, taking the colors of **Dieter Rams' Braun ET66** (of 1987, the calculator which inspired **Steve Jobs'** designs) and wishfully assuming the good old slanted *HP* keys of the 70's. In return to said poll, I posted a first draft of the **WP 43S** incl. its innovative *menu* concept featuring three rows of softkeys displayed in the *menu section* (a user interface concept I had even outlined in an article for the *HPCC*



Datafile V27 N3 in May 2008 already) late in November 2012 (→ <https://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=234685>).

The sequence and location of the four basic arithmetic operators on the keyboard may be subject to vivid discussions even after more than a decade. Thus, let's take a short look at history:

All the early *HP pocket* calculators (from the world's very first scientific pocket calculator, the *HP-35* of 1972, up to the *HP-41CX* of 1983) put them beneath **ENTER↑** in the order **−**, **+**, **×**, and **÷** (top down). Though actually no one knows the reason for this vintage arithmetic operator sequence anymore [we would have to ask *Tom Osborne* and hope he will remember still after more than five decades, → http://www.hp9825.com/html/osborne-s_story.html :



The *HP-35* project was just plain fun. We knew it would work (...) so we spent a lot of time deciding whether the arithmetic keys should be on the right or the left, and whether the “+” key should follow the convention of adding machines and be placed in the lower right corner, or whether it should be located where it is most convenient (for whatever reason).¹

With the landscape *Voyagers* in '81, the arithmetic keys *had* to move from the left to the right of the numeric keys. In that course, *HP* abandoned their proprietary operator sequence for pocket calculators (re-) turning to the conventional adding machine scheme with $+$, $-$, $\times$, and $\div$ bottom up (featured on all their desktops anyway).

HP kept this scheme for all their subsequent calculators until today.

Three weeks after said first concept post of 2012-11, [WP 43S](#) got its iconic ‘Ukrainian’ display marks (rendering clumsy display pointers or screen shading obsolete; as [Dieter Rams](#) stated: “Good design is as little design as possible”). Thereafter, just the six brown keytops in second and third row had to be changed until today. Everything else was patient refinement and careful tuning – shifted functions wandered around (→ B-1ff); and, alas, slanted keys were not available anymore.

Our calculator layout even survived an hardware switch – we were still waiting for [Eric Smith’s](#) and [Richard Ottosen’s](#) so-called *Nordic Reptiles*²³² (→ *US HHC* meetings until 2015) when [Michael](#) mailed me the concept of his and [David’s](#) *DM42* in 2016-03. Development of the *DM42* easily overtook the *WP 43S* – it was launched as a resurrection of the *HP-42S* late in spring 2017 while [Paul](#) and I were looking for actual support (beyond friendly words) and willing software engineers still in vain. Despite its popularity in the (reading) community, lack of qualified coding staff was the lasting bottleneck in our project since 2012. It was overcome not earlier than in 2020, when more than only one programmer joined us (in 2025, we fell back to the pre-2020 state, → 353).

After long and tense negotiations with *SM* (they didn’t want to manufacture just batches of hardware for us), a *WP43* pilot series was produced under control of *SM* in China in the summer of ‘22. It was distributed and sold at the *Allschwil Meeting* ‘22 and in few weeks following. Thereafter, we just had to fill the last free slots on shifted **7** and **8** (→ [App. K](#)).

Just in case: Anyone wanting to modify the *WP43* layout for whatever reason must tell me which function (s)he wants to replace where, and where to this displaced function shall go instead. All primary functions are set – modifying secondary functions is possible since it will (just) cost an overlay. Note all functions must be placed again – no other modification will be even considered.



²³² Kudos to [Eric](#) and [Richard](#) for finding the display we all use now.

On the other hand, the C43/47 team has pursued a layout featuring just a single shift key for years²³⁴, primarily to keep the keytops of DM42 unchanged – so you can get a C47 by just putting an overlay on a DM42, hardware wise. The necessary 2nd shift is reached by pressing  twice there. They confirmed they will continue this way.

Additionally, they unveiled a dual-shift-key layout in 2023-12, called *R47* (→ overleaf). Therein, 36 keys match your *WP43* (8 of them are just mirrored), and only **7 keys** plus (of course) the bezel differ significantly. A pilot batch has been manufactured by *SM* in 2025, and first *R47* calculators are being sold. Note, however, that an *R47* can't be repurposed into a *WP43* for the time being since it is electronically equivalent to a *DM42n* (→ 184).

Angles

This innocent looking *DT* caused many and long discussions. A separate *DT* for *angles* was planned from the beginning. It was removed in v0.9 since its

²³³ This picture is quoted from <https://arxiv.org/pdf/2603.21852>.

²³⁴ *Jaco's C43* forked from *WP 43S* in 2019. After the *WP43* pilots, the majority of *WP43* developers turned to *C43* in 2022-11 since *SM* stated they will not produce any more *WP43* calculators. It was a decision between design and production. *SM's* reasons are a longer story, mutual misunderstandings included.

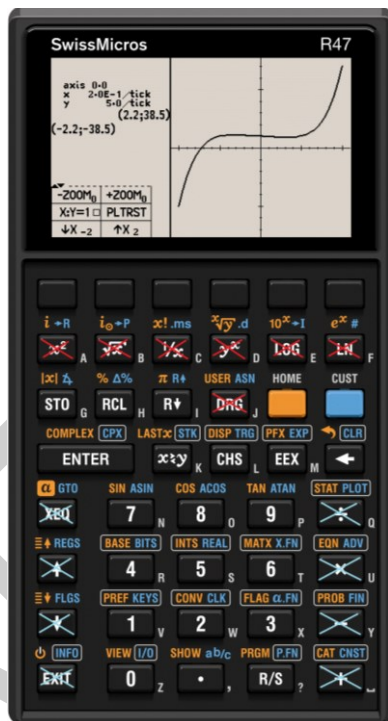
scope is quite limited and the opinion rose that '*angles* work like *reals*'. It turned out, however, that D.MS data would need special treatment in calculations then, so DT 4 returned with v0.10 to keep algebraic operations simple for the user and allow for standard operations instead of dedicated commands like D.MS+, D.MS-, etc.

Actually, *angles* are displayed in various units or '*display modes*' (ADM: *decimal* and *sexagesimal degrees*, *radians*, *multiples of π* , *gon* or *gradians*, and *mils*). They were represented internally by a fixed number of intervals per turn. *Rads*, however, did not fit into this paradigm since π is irrational. So, internally, *angles* are specified as tagged *reals* now ($\rightarrow$ [App. B](#)).

Generally, the trigonometric functions operate on *angles* within $[-180^\circ, 180^\circ]$ (or equivalents) only. Thus, any numeric input beyond this interval shall be ...

1. converted from *long integer*, if applicable, to a **39-digit real**, then
2. reduced modulo **360°** (or **2π** or **2** or **6400°** or **400^g**, according to ADM), then
3. decremented by **180°** (or **π** or **1** or **3200°** or **200^g**),
4. converted into *rads*, if necessary,²³⁵ before calling the respective function.

These modulo calculations are easy for all ADMs except for *rad*, where **2π** must be stored in very high precision (since a full turn can't be represented exactly in *rad* but with finite precision only) to avoid rounding errors in modulo **2π** reduction (step 2) for the angular input range allowed.



²³⁵ ... since the standard *Taylor* series for trigonometric functions are written for *rads*. Although they can be easily rewritten for *multiples of π* nobody did this apparently so far. As far as I am aware of, WP43 was the first calculator worldwide offering *multiples of π* , originally introduced by me in 2016-10 to get rid of the *rad* problem sketched above. Though *rads* are popular in mathematics and among mathematicians, although actually *multiples of π* are used frequently in expressions there instead.

For a more fundamental approach, feel free to look up <https://www.tauday.com/tau-manifesto> of 2010-06-28 (but I don't remember knowing that in 2016) and <https://www.tauday.com/state-of-the-tau> (of 2025).

Example: $\cos(6\,381\,956\,970\,095\,103 \times 2^{797}) = ?$ in RAD

Solution: 797 [2^x] 6 381 956 970 095 103 [x] returns the *long integer*

5 319 372 648 326 541 416 707 296 656 673 541 083 813 475 031 793
 921 822 105 998 164 685 326 343 987 747 477 646 239 125 204 069
 843 392 466 931 105 720 371 047 561 653 378 447 496 736 288 905
 533 500 277 726 150 903 890 962 697 774 418 679 535 123 008 556
 835 980 236 851 047 840 822 029 788 166 318 932 319 835 828 816
 270 258 618 761 216 exactly (i.e. some 5.3×10^{255})

WolframAlpha tells us that this integer value modulo 2π should return

1.570 796 326 794 896 619 700 038 284 065 214 203 210 ...

[i.e. $\pi/2$ (= 1.570 796 326 794 896 619 231 321 691 639 751 442 098 ...)]
 plus some 4.687×10^{-19} . The cosine of this reduced *angle* is

-4.687 165 924 254 627 611 122 582 801 963 884 398 777 914 ... $\times 10^{-19}$
 as expected from the *Taylor series*.

Your *WP43* returns a different result – instead of the precise 256-digit *integer*, it takes 5.319 372 648 326 541 416 707 296 656 673 541 083 81 $\times 10^{255}$ (the 39-digit real), reduces this modulo a **1071**-digit representation of 2π to 3.175 035 381 806 309 195 338 342 479 504 994 764 49, and thus gets -0.999 440 844 081 712 927 910 734 972 739 862 0 for its cosine.²³⁶

Note, however, that angular input $\geq 10^{34}$ completely loses significance in modulo reduction since no such number can support a defined fraction of a full circle anymore (this holds for any *ADM*). Hence, very large *angles* make no sense in real-life problems (hence we throw a warning there, $\rightarrow$ [App. C](#)). But if you need them for any reason whatsoever, avoid specifying them in *rads* at least.

Examples: Assume an *angle* of $(360 \times 10^{16} + 45)^\circ$ or equivalents in the other *ADMs*. Then the sine of this *angle* returns in *ADM* ...

$\angle^\circ$: $\sin(360 \times 10^{16} + 45)^\circ =$ 0.7071067811 8654752440 0844362104 849

$\angle^r$: $\sin(2\pi \times 10^{16} + \pi/4)^r =$ 0.7071067811 8654752631 0312520267 0949

$\angle\pi$: $\sin(2 \times 10^{16} + 1/4)\pi =$ 0.7071067811 8654752440 0844362104 849

$\angle^-$: $\sin(6400 \times 10^{16} + 800)^- =$ 0.7071067811 8654752440 0844362104 849

²³⁶ Note that using [d] for converting said *long integer* input leads to a different result since it will provide only 34 digits instead of 39.

And *WolframAlpha* returns for the cosine of above 39-digit *real number* (precisely for $N(\cos[\text{Mod}(531\,937\,264\,832\,654\,141\,670\,729\,665\,667\,354\,108\,381 \times 10^{217}, \text{floor}(2\pi, 10^{-1070})]), 34)$, $\text{WorkingPrecision} \rightarrow 1100$) the same as your *WP43* (actually it does this for any precision of 2π exceeding 288 decimals).

$$\text{✖}^g: \sin(400 \times 10^{16} + 50)^g = 0.7071067811\ 8654752440\ 0844362104\ 849$$

$$\text{Expected result is } \frac{1}{2}\sqrt{2} = 0.7071067811\ 8654752440\ 0844362104\ 849$$

Assume, on the other hand, an *angle* of $(360 \times 10^{16} - 30)^\circ$ or equivalents in the other *ADMs*. Then the sine of this *angle* returns in *ADM* ...

$$\text{✖}^o: \sin(360 \times 10^{16} - 30)^\circ = -0.5$$

$$\text{✖}^r: \sin(2\pi \times 10^{16} - \pi/6)^r = -0.5000000000\ 0000000\ 398\ 9637143416\ 523$$

$$\text{✖}^\pi: \sin(2 \times 10^{16} - 1/6)\pi = -0.5000000000\ 0000000\ 906\ 8996821171\ 0892$$

$$\text{✖}^-: \sin(64 \times 10^{18} - 1600/3)^- = -0.4999999999\ 9999999\ 716\ 5938493384\ 0346$$

$$\text{✖}^g: \sin(4 \times 10^{18} - 100/3)^g = -0.4999999999\ 9999999\ 546\ 5501589414\ 4554$$

due to the limited precision of the fractions involved – after modulo reduction an accuracy of ‘only’ 18 digits remains for the reduced *angle*. Above results for ✖^r and ✖^g match those returned by C47.²³⁷

Within ± 500 full rotations, trigonometric function results in different *ADMs* match each other and expectations to some 10^{-31} – this should suffice for all real-life problems. Above examples are of purely academic value only.

On the other hand, *WP 34S* has demonstrated that 451 digits for 2π suffice to warrant 16-digit accuracy of function results within the range of *single precision reals* there, though neglecting the significance limit mentioned above.²³⁸ In full analogy, *WP43* uses 1071 digits of 2π to warrant 34-digit accuracy of respective function results for 34-digit input within $\pm 10^{999}$. Thus, **RANGE 999** is the maximum sensible setting if you decline confining trigonometric functions to sensible arguments.

Calculation Internals

For $y > 0$, general powers in $\mathbb{R}$ and $\mathbb{C}$ are calculated 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.

For odd (‘integer’) roots of $y < 0$, $\sqrt[x]{y} = -e^{\frac{\ln(-y)}{x}}$; here, ‘integer’ includes *DT 1* and *10* and *reals* with zero fractional part. Some special results in *App. B* can be deduced from these calculation paths.

Note that *Free42* calculates even powers differently (as a series of multiplications using repeated squaring), never employing more than 34 digits ($\rightarrow$ also p. B-23).

²³⁷ Compare the *HP-15C Advanced Functions Handbook*, pp. 184 – 186, obviously written by professor *William Kahan*. $\rightarrow$ also *Precision and Accuracy* below.

²³⁸ $\rightarrow$ <https://forum.swissmicros.com/viewtopic.php?f=2&t=350#p4349>

Character Sets

FBR displays all the *characters* provided as designed and implemented for the [WP 43S](#) very early in this project, sorted according to their hexadecimal codes (most of them following *Unicode*). Some *characters* had to be ‘invented’ in either font since I found no *Unicode* equivalents in the respective range, e.g., the *USB* symbol, distribution indices, [Euler’s *e*](#) and complex *i* for sake of unambiguity.

The so-called ‘**numeric**’ font uses a matrix of up to 16 × 32 px (variable width, fixed height). Therein, the punctuation space (2008₁₆, 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 *WP43*. It is also employed for echoing numeric input unless too long. It could be used for echoing command input as well – screen space suffices.

Six blank *characters* are provided here 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 CONSTants. Optionally, narrower digits (2470₁₆ ff) can be used in *complex numbers*, matrices, or [short integers](#) of small base where space may be scarce (→ B-17ff).

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 end.

Characters with codes < 0020₁₆ are for control purposes; e.g., 4, 10₁₀, or 27₁₀ may be useful for printer control (e.g., of an [HP-82240 A/B](#)).

Many symbols are 8 px wide as digits, helping where a constant symbol spacing is wanted. Many super- and subscripts are also provided, allowing for displaying all the *items* featured on *WP43*. Arbitrary numeric indices or exponents are possible as well.

Eight blank *characters* are featured, listed here with their addresses and widths → Using them, any spacing is viable.

	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

Find below some 900 *characters* I have designed for the **WP43_NumericFont** and the **WP43_StandardFont** with their codes (note 102 / 149 *UTF* code points were **abused** below (plus 18 being **unused** so far) – I didn't find any equivalent *characters* in the range covered):

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0020		!	/	#	/	%	/	'	(	)	*	+	,	-	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	/	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0060	/	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0070	p	q	r	s	t	u	v	w	x	y	z				~	/
...																
00b0	°	±				μ		·								
00d0								×								
0170								×								
0230			ȳ	ȳ												
0370								̄	̂							̄
...																
0390	/	A	B	Γ	Δ	E	Z	H	Θ	I	K	Λ	M	N	Ξ	O
03a0	Π	P		Σ	T	Υ	Φ	X	Ψ	Ω						
03b0		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
03c0	π	ρ		σ	τ	υ	φ	χ	ψ	ω						
...																
2000																
2010									'	'						
...																
2080	0	1	2	3	4	5	6	7	8	9	+	-				*
2090															∞	∞
...																

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
2100			©												h	ħ
2140								e	i							
2160	0	1	2	3	4	5	6	7	8	9	+	-				
21c0	→				≥											
2210											√	∛	∜		∞	
2220		¼										∫				
2240									≈							
2260	≠				≤	≥										
2290					⊕	α	δ	μ		⊙						
2390										☰						
23a0																
23e0								10								
2420								☐						UK	US	
2460	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2470				0	1	2	3	4	5	6	7	8	9			
2480								f	g	h						
2490				r						x			a	b	c	d
24a0	e				i	j	k	l	m	n	o	p	q		s	
24b0	u	v	w	x	y	z										
24c0										T						
24d0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
24e0	Q	R	S	T	U	V	W	X	Y	Z						

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0020		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0060		a	B	c	d	e	f	g	h	i	j	k	l	m	n	o
0070	p	q	R	s	t	u	v	W	x	y	z	{		}	~	
...																
00a0		i	¢	£		¥		§		©		«	¬			
00b0	°	±	²	³		µ		·		¹		»	¼	½		¿
00c0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
00d0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý		ß
00e0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
00f0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý		ÿ
0100	Ā	ā	Ă	ă	Ą	ą	Ć	ć					Č	č	Ď	ď
0110	Đ	đ	Ē	ē	Ĕ	ė	É	Ê	Ė	ę	Ė	ė			Ğ	ğ
0120								ħ			Ī	ī	Ĭ	ĭ	Į	į
0130	İ	ı								Ł	ł			Ł	ł	
0140		Ł	ł	Ń	ń			Ń	ń				Ō	ō	Ŏ	ö
0150			Œ	œ	Ř	ř			Ř	ř	Š	š			Ş	ş
0160	Š	š	Ţ	ţ	Ť	ť			Ů	ů	Ū	ū	Ů	ů	Ů	ů
0170			Ů	ů	Ű	ű	Ŷ	ŷ	Ÿ	Ž	ž	Ž	ž	Ž	ž	
...																
0230			ȳ	ȳ												
...																
0370									̄	̄						̄
0380																
0390	Ĭ	A	B	Γ	Δ	E	Z	H	Θ	I	K	Λ	M	N	Ξ	O
03a0	Π	P		Σ	T	Υ	Φ	Χ	Ψ	Ω	İ	Ÿ	ά	έ	ή	ί
03b0	Û	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
03c0	π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	ϊ	ϋ	ό	ύ	ώ	
...																
2000																
2010									‘	’	,	‘	“	”	”	”
2020							...	1								
...																
2070	0		-1	E	4	5	6	7	8	9	+	-				
2080	0	1	2	3	4	5	6	7	8	9	+	-				*
2090														∞	∞	
20a0													€			
...																
2100			©												h	ħ
2110			ℓ		N					Q			R			
2120				Z												
...																
2140								e	i							
...																
2160	0	1	2	3	4	5	6	7	8	9	+	-				
...																
2190	←	↑	→	↓		↕										
...																
21b0						ς										
21c0	→			↘				Δ		▽						
...																
21e0							↑		↓							
...																
2200	∇	∫	∂	∃	∅	∆	∇	ε	€		∂	∅		.	π	
2210				±				°	•	√	³	¼	α	∞	ℓ	
2220	∠	4	℥		†		‡	λ	ν	η	υ	∫	∫	∫	∫	∫
2230	∫					:										
2240				≈				≈								
2250				≡				≡	≡							
2260	≠	≡		≡	≡						≡	≡				
...																
2290						⊕	α	δ	μ	⊙	⊙	⊙				

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
22a0					T	L										
22b0												y	ā	ȳ		
2300									f	l	L	J				
2310											⌚	⌚				
2390											📄					
23a0		f	l	L	J											
23e0									10							
23f0		⌚														
2420				⌚				⌚			🔋	⬆️	U	UK	US	⌚
2430	⬆️	⬆️	⬆️	⬆️	⬆️											
2460	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
...																
2480			a	b	c	d	e	f	g	h	i	j	k	l	m	n
2490	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
24a0	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
24b0	u	v	w	x	y	z	A	B	C	D	E	F	G	H	I	J
24c0	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
24d0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
24e0	Q	R	S	T	U	V	W	X	Y	Z						
...																
2710				✓												

This latter font allows for correctly spelling the languages of >3.5×10⁹ people using either Greek or Latin alphabets:

Afrikaans, aymara, Bahasa Indonesia, Bahasa Melayu / Malaysia, Basa Jawa, Basa Sunda, Bhāsa Bali, bosanski, català, Cebuano (Bisayan), čeština, Cymraeg, dansk, deutsch, eesti, Ελληνικά, English, español, euskara, Filipino, français, Gaeilge, galego, Hiligaynon, hrvatski, Ilokano, italiano, Kiswahili, kreyòl (ayisien), kurdî, lietuvių, magyar, Malagasy, Nāhuatl (Mexicatlatolli), nederlands, nihongo (rōmaji), norsk, O'zbek tili, polski, portu-

guês, quechua (runasimi), română, shqip, slovensčina, slovensky, srpski, suomi, svenska, Tagalog, tatarça, Türkçe, Türkmen dili, Vlaams, walon, Waray, and zhōngwén (hànyǔ pīnyīn).²³⁹

This made the *WP43* the most versatile multilingual calculator available worldwide. See the *OM* for examples where and how these characters can be used.²⁴⁰

Complex Notation, Storage, and Calculations Close to Infinity

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, every notation will do.

For reasons of tradition, rectangular notation is seen most frequently. We used it for storing *complex numbers* in the *WP 34S*. Hence, it is used for the *WP43* as well; but care must be taken at *complex infinities*.

Since infinities are taken as numeric data being part of the *real number* range ($\rightarrow 196$), also *complex infinities* are part of the *complex number* plane the *WP43* operates on. In rectangular notation, eight '*complex infinities*' are possible only, listed here aside to their equivalents in polar notation (note the phase is counted counterclockwise, starting with $\varphi = 0$ at the positive *real* axis). $\rightarrow$

Re(z)	Im(z)	r(z)	$\varphi(z)$	
$-\infty$	$-\infty$	∞	-135°	$-3\pi/4$
0	$-\infty$		-90°	$-\pi/2$
∞	$-\infty$		-45°	$-\pi/4$
∞	0		0°	0
∞	∞		45°	$\pi/4$
0	∞		90°	$\pi/2$
$-\infty$	∞		135°	$3\pi/4$
$-\infty$	0		180°	π

Calculating with *complex infinities*, every finite number 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 just the eight cases listed here as long as you build your *complex* calculations on *Cartesian* notation – while with polar notation, an infinite number of *complex infinities* would have to be treated (but $\rightarrow 202$, f. 148).

Though polar notation is advantageous for *complex* powers and roots:

²³⁹ $\rightarrow$ https://de.wikipedia.org/wiki/Liste_lateinischer_Alphabete#Erweiterungen.

²⁴⁰ Some *characters* are not required for any *item* provided nor show up in any menu so far. They may be used in future.

The n^{th} root of a *complex number* $(r; \varphi)$ in polar notation shall return $(\sqrt[n]{r}; \varphi/n)$. Hence, e.g., $\sqrt[3]{(\infty; 180^\circ)} = (\infty; 60^\circ)$, corresponding to the *Cartesian* point $\infty \times (1 + i\sqrt{3})$ — but the calculator will display this point as $\infty + i \times \infty$ (converted following the calculation rules mentioned above) though mathematically wrong. And $\sqrt[6]{(\infty; 180^\circ)} = (\infty; 30^\circ)$, corresponding to the *Cartesian* point $\infty \times (\sqrt{3} + i)$, will return $\infty + i \times \infty$ as well.

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

Thus, the odds are high that roots and powers of *complex numbers* near the edge of the *complex* plane will return incorrect data, in particular if entered in polar notation (and immediately translated internally to *Cartesian*). These errors seem to be inevitable.

Display Segmentation

The *LCD* specifications were defined very early in this project. They were refined in 2017 when we learned of the final display specs ($\rightarrow$ [App. A](#), p. B-6). Now you find top down ...

1. 20 px for the *status bar*,
2. 4 blank pixel rows for separation,
3. 147 px for either ...
 - a. the contents of ≤ 4 *stack registers* or ≤ 5 rows of matrix elements, or
 - b. 7 program steps in *PEM*, or
 - c. ≤ 7 rows of numeric output of **SHOW**, and
4. ≤ 69 px for the menu section.

2017-05-08 23:49 C.L. 4π.					
-12 345 678 91					
-9.234 567 89					
-5.678 901 23					
010 1100 0111					
ABCDEF	B				
B					
C					

Reasons for these figures are:

- Each regular alphanumeric row (e.g., for labels, status, a text string, or a program step) requires 20 px vertically ($\rightarrow$ B-11f). There shall be at least one row of blank pixels separating it from the next row of data.
- Hence, each *menu* row takes (counting bottom-up) $1 + 20 + 1 + \blacksquare = 23$ px

(the first pixel is for separation from the black frame, the last for its upper frame line). Thus, three *menu* rows require 69 px. In $\text{U} \rightarrow$, extra-high labels may appear: they will require $1 + 20 + 1 + 20 + 1 + \boxed{1} = 44$ px for double height or even $1 + 20 + 1 + 20 + 1 + 20 + 1 + \boxed{1} = 65$ 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 remain free on the screen so far.
- Each regular numeric output requires 32 px vertically ($\rightarrow$ B-11) plus 4 px separating it from the next such output row. Thus, for 4 rows we need $4 \times 32 + 3 \times 4 = 140$ px. Since there are 4 px above them already (separating it from the *status bar*), we put the remaining 7 px below this 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.
- The same applies to **SHOW** where pure numeric output may require more than one row ($\rightarrow$ 83).

In the *menu section*, we also have a horizontal structure for the six *menu* keys. 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 (≤ 7 character) *labels* wider than 64 px in *menu views* may be not fully displayed, so labels deviating only in their very last *characters* may become visually indistinguishable. Such ambiguities should be avoided.

If gaps are chosen between groups of digits (**GAP n** with $n \geq 2$), they are filled by a punctuation space each.

Echo and Fallback

Most key presses are echoed and fall back to **NOP** after 2 s. *Softkeys* shall not since *WYSIWYG* applies here always. Furthermore, some functionalities

(wherever they are assigned to) are never echoed (and hence can't fall back) since

- 1) they are dedicated to numeric input or correcting it ($\boxed{0}$ – $\boxed{9}$, $\boxed{.}$, $\boxed{E}$, $\boxed{+/-}$, and $\boxed{\leftarrow}$) or
- 2) it has a special long-press functionality ($\boxed{EXIT}$ turns to **EXITALL** after 2 s) or
- 3) reverting or exiting them is a no-brainer ($\boxed{\blacktriangle}$, $\boxed{\blacktriangledown}$, $\boxed{P/R}$, and all *menu* calls) requiring just one keystroke.

With the presence of **UNDO** ($\rightarrow$ B-32), fallback to **NOP** becomes debatable overall; there is a benefit remaining in *user mode* as long as an overlay matching the actual assignments is not available – and **UNDO** is shifted in *SDC*.

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* (a *variable name* ends when either the next operator or a **:** or a **)** is entered) as well as after **=** and behind each operator like **+**, **-**, *****, **/**, and **!** (except **^**); it is also inserted after **:**. There is no implicit multiplication.

Functions like roots, absolute values, or trigs shall be written using parentheses or bars, e.g.,

- pressing $\boxed{\sqrt{}}$ shall display $\sqrt{}$,
- pressing $\boxed{e^{}}$ shall display $e^{}$, and
- pressing $\boxed{| |}$ shall display $| |$;
- cosine shall be either picked from CATALOG'FCNS or entered **cos()** – both will be echoed as **cos()**;

then specify the argument. The same applies to dyadic (like **COMB**) and triadic operations in analogy – their arguments shall be separated by colons inserted from the *Equation Editor menu*.

Terminating the *Equation Editor*, numeric exponents are automatically converted from, e.g., **xy^23** to **xy²³**. For easier handling, this will be reverted when editing such an equation again.

Execution Times

The time a particular command takes to be executed depends on the command and its argument or parameter. For trigonometric functions, for example, execution times may easily vary by a factor of two.

Just for your information, here are measurement results for some monadic functions operating on $x = \pi/6.3 \approx 0.498\,665\,5$ on a WP43 v0.27 pilot powered through USB:

erfc	67 ms
$\Gamma(x)$	45 ms
$\ln\Gamma$	37 ms
arcosh	22.2 ms
erf	21.8 ms
sinc	12.4 ms in 4^r
E(m), K(m)	10.8 ms
$(-1)^x$	6.6 ms in 4^r
sincπ	6.1 ms in 4^r
tanh, arccos	5.6 ms
arsinh	4.9 ms
e^x, sinh, cosh	4.8 ms
arcsin, arctan	4.6 ms
2^x	4.5 ms
10^x, artanh	3.9 ms
tan	3.8 ms in 4^r
ln, lg, lb	3.2 ms
$\sqrt[3]{x}$	2.4 ms
sin, cos	2.3 ms in 4^r ²⁴¹
$\sqrt{(1+x^2)}$	1.8 ms
$\sqrt{x}$	1.3 ms
x^3	0.7 ms
$1/x$, x^2, FP, IP, ceil, floor, ROUNDI, x	0.4 ms – 0.5 ms

²⁴¹ For the *angle* used here, computing the sine in DEG took 0.1 ms less.

Functionality of ENTER↑

ENTER↑ on the *WP43* works as it does on the *HP-42S* – our reference – and on our *WP 34S* and *WP 31S*. This is “*Classic RPN*”. Some *HP RPN* pocket calculators of last years featured slight variants of this functionality.²⁴²

Keystrokes

On calculators, a keystroke is a keystroke is a keystroke in my opinion. Strictly no aberrations like double strokes (watch Parkinson!), long presses, or alike – or do you know of any *HP* pocket calculator featuring such a user interface?

(I have to admit we have one exception: **EXIT** – but else:) As long as I'm responsible for *WP43* design, such stuff will go only over my dead body. Period.

MSA

Actually, this is the very reason why I wanted this calculator. Working for the Swabian corporation mentioned in f. 115, we did a lot of such analyses for investigating and qualifying instruments for production and quality control, using a little homegrown *MS Excel* sheet supporting this method by automatic evaluation. It could even be used for dispassionately clearing a μm complaint of a big US east-cost computer corporation – turned out they didn't know their instrumentation sufficiently. Made some Swabians smile.

At the bottom line, this method proved being easy enough for the shop floor also in our international affiliates and very beneficial for our company. Thus, I voted for implementing it in your *WP43*. Thanks to *Jaco* for making this possible.

Names of User Defined Items

Such a *name* may consist of max. 7 *characters* containing at least 1 letter, and it shall begin with a letter. For user-created *labels*, *menus*, and *variables*, +, -, ×, /, :, ^, (,), =, !, ;, |, comma, and space must not be used. Examples of legal user-defined *names* are **where?**, **all.in**, and **Go_17** (→ also p. 346).

Predefined commands, constants, *menus*, *variables*, and *system flags* do not suffer from these restrictions.

²⁴² Learn more about those variants in an article (4 pages) by *Richard Nelson*: <http://h20331.www2.hp.com/hpsub/downloads/S07%20HP%20RPN%20Evolves%20V5b.pdf>.

Number Range

A number range up to 10^{99} is sufficient for almost all real-world problems – else common scientific calculators would generally feature a larger range ($\rightarrow 126$). So we can conclude that the *real* range supported by your WP43 ($\rightarrow 193$) suffices by far for solving what has to be solved. Saving display space for sensible results gave reason for **RANGE**.

For sake of coherence, maximum numbers within different *DTs* should match. I.e. the maximum absolute value allowed for a *long integer* should be approximately equal the respective values for a *real* and the parts of a *complex number*. Hence the startup default *real* range of $\pm 10^{999}$. In real life, e.g., huge *angles* are meaningless ($\rightarrow$ B-7ff).

Note the number range determines the precision required for calculating accurately ($\rightarrow$ B-10).

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 featured by the calculator. Already a simple fraction like $1/3$ (stored as a *real number* following the representation as explained on pp. 193ff) deviates from the truth by more than 3×10^{-35} . This looks very small though such errors will accumulate during longer calculations ($\rightarrow$ 302f).

In the check on p. 265 using primes, results $\neq 0$. occur more frequently between 700 and 1000, while you won't find a single one between 1000 and 5300!

In the majority of cases we calculate with 39 internal digits and round the result to 34 digits. Then ...

- [illegible]

In real-world problems, usually the least accurate of all input parameters determines the accuracy of the result. In the standard test mentioned on p. 266 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 step a trigonometric function is calculated there).²⁴³

Internally, e.g., the **WP 34S** computes with 39 digits and rounds the results to 34 or 16 digits, respectively. Consequently, **WP43** also works with 39 digits internally most times and rounds results to 34 digits. Each of the 79 constants in **CONST** and the 62 conversion factors feature 51 digits internally (rounded from 55 specified throughout `generateConstants.c`). This applies also to most of the ca. 200 auxiliary constants stored (like $\frac{1}{3}, \frac{2}{3}, \ln 2, \ln 10, \frac{1}{2} \ln(2\pi), \pi, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \frac{3\pi}{2}, \frac{180}{\pi}, \frac{200}{\pi}, \frac{3200}{\pi}, \frac{1}{2}\sqrt{2}, \frac{1}{2}\sqrt{3}, e, \gamma$,²⁴⁴ but also trivial values like 0.5 with over 50 zeros). **SLVQ** calculates with 72 digits. The statistical summation registers are 75 digits wide (used also for the initial steps of variance calculation).

Real-world problems are usually many orders of magnitude less precisely specified than the internal precision of your **WP43**. Compare also the precisions in the set of physical and astronomical constants provided (→ 116ff). To get an idea what an 'order of magnitude' means in real world, look there as well. Also → 124, f. 80.

Generally for (in-) accuracy indications, relative deviations $\delta_r = \left| \frac{v_T - v_C}{v_C} \right|$ are standard with v_T being the true and v_C the computed value (cf. **ACC** in **OMS 5**, inherited from the **HP-34C**).

For comparing a set of generally very small δ_r , one can work with $-lg(\delta_r)$ instead to get positive values of reasonable size.

Prefixes

Prefixes **f** and **g** passed without any discussion for more than six years until 2019-06. Alternatively, prefixes **g** and **f** could have been chosen but their typography leaves less freedom for placing the corresponding **golden** and **blue** labels. Dropping the letters **f** and **g** – resulting in **g** and **f** – made b/w copies of the manuals less useful (which is on purpose).

²⁴³ If you are generally interested in accuracy of numerical calculations, reading the appendix of the **HP-15C Advanced Functions Handbook** is recommended (pp. 172ff, written by professor **William Kahan**).

²⁴⁴ The last is the **Euler-Mascheroni** constant, called γ_{EM} in **CONST**. The **Lanczos** coefficients are stored with 97-digit precision, 2π with 1100 digits (at position 312) for the modulo reduction of → **Angles**.

Random Numbers

RAN# and **RANI#** use a *PCG* random generator (→ <http://www.pcg-random.org>).
Look up p. 343 as well.

RPN vs. RPL or Vice Versa

HP moved from *RPN* to *RPL* in 1986 for their high-end scientific pocket calculators – the big ones exceeding pocket size. There were good reasons for doing that move. *RPL* is a very coherent operating system – actually it is so coherent that it hurts (in my opinion). It carries the postfix principle to the extreme.

RPL may be next to perfect (for programming, in particular) if you use it every day – else it is annoying, however. Since almost all of our users use their *WP43* casually nowadays, the annoying part would become overwhelming very soon. Thus, we stick to classic *RPN* – quick and dirty, but far easier to resume after pausing for days, weeks, or even months.

It took me until 2025 to realize that we had silently taken over one of the best properties of *RPL*: object orientation.²⁴⁵ By introducing *data types* and letting functions operate the same way on objects of various *DTs*, we eased computing significantly compared to, e.g., *HP-41C* or *HP-15C*. I didn't notice that earlier since our reference, the *HP-42S*, used *DTs* already though to less extent. Actually, one should discern *RPN* before *RPL* from *RPN* thereafter.

Presently, there are projects called *DB48X* and *NewRPL* pursuing an extended *RPL* implementation on a new platform (since *HP* abandoned *RPL* ceasing production of the *HP-50g* in 2015). There is, however, no calculator hardware readily available to host these systems as far as I am aware of.

Sorting in Detail

There is no international standard for sorting *characters*; we had to invent our own. Thus, sorting of *items* (i.e. commands, *variables*, *text strings*, *labels*, *system flags*, etc.) on *WP43* 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. The *IOI* is sorted according to this algorithm (hopefully), *CATALOG* is definitely.

Sorting is specified for the small font below. It holds also for the large font as far

²⁴⁵ Thanks to *c3d* for pointing me to this.

as *characters* are applicable there. The 4-digit number following each *character* in the table is its hexadecimal *Unicode*. 163 codes printed on blue or green were abused (→ B-11ff). *Characters* printed on bright yellow or orange are neither accessible through the keyboard nor contained in any alpha *menu*; the latter ones are designed but not used at all by WP43 so far.

0020	2003	2004	...	2008	200a	2423
0 0030	. 220e	° 00b0	0 2160	0 2080		
1 0031	1 2027	¼ 00bc	½ 00bd	1 2161	1 2081	1 2460
2 0032	2 2162	2 2082	2 2461			
3 0033	3 2163	3 2083	3 2462	√ 221b		
4 0034	4 2164	4 2084	4 2463			
5 0035	5 2165	5 2085	5 2464			
6 0036	6 2166	6 2086	6 2465			
7 0037	7 2167	7 2087	7 2466			
8 0038	8 2168	8 2088	8 2467			
9 0039	9 2169	9 2089	9 2468			
10 23e8	10 2469					
11 246a						
12 246b						
13 246c						
14 246d						
15 246e						
16 246f						
A 0041	a 0061	À 24b6	à 2482	À 24d0	à 249c	
	À 00c0	à 00e0	Á 00c1	á 00e1	Ã 00c2	ã 00e2
	Ã 00c3	ã 00e3	Ä 00c4	ä 00e4	Å 00c5	å 00e5
	Æ 00c6	æ 00e6	Ä 0100	ä 0101	Å 0102	å 0103
					Ä 0104	ä 0105
B 0042	b 0062	ß 24b7	b 2483	ß 24d1	b 249d	

C 0043	c 0063	ċ 24b8	ċ 2484	ċ 24d2	ċ 249e	
	Ç 00c7	ç 00e7	Ć 0106	ć 0107	Č 010c	č 010d
					Ĉ 2102	ĉ 2201
D 0044	d 0064	Ḑ 24b9	Ḑ 2485	Ḑ 24d3	Ḑ 249f	
	Ð 00d0	ð 00f0	Ǧ 010e	ǧ 010f	Đ 0110	đ 0111
E 0045	e 0065	Ē 24ba	ē 2486	Ē 24d4	ē 24a0	Ê 00c8
	è 00e8	É 00c9	é 00e9	Ê 00ca	ë 00ea	Ë 00cb
	ě 00eb	Ě 0112	ě 0113	Ě 0114	ě 0115	Ě 0116
	è 0117	Ė 0118	ė 0119	Ě 011a	ě 011b	E 2073
	e 2147					
F 0046	f 0066	F 24bb	f 2486	F 24d5	f 24a1	
G 0047	g 0067	G 24bc	f 2487	G 24d6	g 24a2	
					Ğ 011e	ğ 011f
H 0048	h 0068	h 210e	H 24bd	h 2488	H 24d7	h 24a3
					Ħ 0127	ħ 210f
I 0049	i 0069	I 24be	i 2489	I 24d8	i 24a4	
	İ 00cc	î 00ec	Ī 00cd	î 00ed	Ī 00ce	î 00ee
	ï 00cf	ï 00ef	Ī 012a	ī 012b	Ī 012c	ī 012d
	İ 012e	î 012f	Ī 0130	ı 0131	i 2148	
J 004a	j 006a	J 24bf	j 248a	J 24d9	j 24a5	
K 004b	k 006b	K 24c0	k 248b	K 24da	k 24a6	
L 004c	l 006c	L 24c1	l 248c	L 24db	l 24a7	ł 2113
	Ł 0139	ĺ 013a	Ĺ 013d	ľ 013e	Ł 0141	ł 0142
M 004d	m 006d	M 24c2	m 248d	M 24dc	m 24a8	
N 004e	n 006e	N 24c3	n 248e	N 24dd	n 24a9	Ñ 00d1
	ñ 00f1	Ñ 0143	ń 0144	Ñ 0147	ñ 0148	N 2115
O 004f	o 006f	o 24c4	o 248f	o 00a9	o 24de	o 24aa
	ò 00d2	ò 00f2	Ó 00d3	ó 00f3	Õ 00d4	ô 00f4
	õ 00d5	õ 00f5	Ö 00d6	ö 00f6	Ø 00d8	ø 00f8
	ō 014c	ō 014d	Ǫ 014e	ǫ 014f	Œ 0152	œ 0153

P 0050	p 0070	P 24c5	P 2490	p 24df	p 24ab	
Q 0051	q 0071	Q 24c6	q 2491	Q 24e0	q 24ac	Q 211a
R 0052	r 0072	R 24c7	r 2492	R 24e1	r 24ad	
		Ř 0154	ř 0155	Ř 0158	ř 0159	R 211d
S 0053	s 0073	S 24c8	s 2493	s 24e2	s 24ae	Š 015a
	ś 015b	Ş 015e	ş 015f	Š 0160	š 0161	ß 00df
T 0054	t 0074	T 24c9	T 22a4	t 2494	T 24e3	t 24af
			Ț 0162	ț 0163	Ț 0164	ţ 0165
U 0055	u 0075	U 24ca	u 2495	u 24e4	u 24b0	
	Ù 00d9	ù 00f9	Ú 00da	ú 00fa	Û 00db	û 00fb
	Ü 00dc	ü 00fc	Ů 0168	ů 0169	Ů 016a	ů 016b
	Ů 016c	ů 016d	Ů 016e	ů 016f	Ț 0172	ț 0173
V 0056	v 0076	V 24cb	v 2496	v 24e5	v 24b1	
W 0057	w 0077	W 24cc	w 2497	W 24e6	w 24b2	
					Ẃ 0174	ẃ 0175
X 0058	x 0078	X 24cd	x 2498	x 24e7	x 24b3	
			Ẃ 0379	ẃ 0378	Ẅ 037f	Ẃ 221c
Y 0059	y 0079	Y 24ce	y 2499	y 24e8	y 24b4	
	Ý 00dd	ý 00fd	Ÿ 0176	ÿ 0177	Ÿ 0178	ÿ 00ff
					Ț 0233	ț 0232
Z 005a	z 007a	Z 24cf	z 249a	z 24e9	z 24b5	Ž 0179
	ž 017a	Ž 017b	ž 017c	Ž 017d	ž 017e	Z 2124
A 0391	α 03b1	α 2296	á 03ac			
B 0392	β 03b2					
Γ 0393	γ 03b3					
Δ 0394	δ 03b4	δ 2297				
E 0395	ε 03b5	É 03ad				
Z 0396	ζ 03b6					
H 0397	η 03b7	ή 03ae				
Θ 0398	θ 03b8					

l 0399	ł 03b9	í 03af	ï 03aa	ï 03ca	ĩ 0390	
K 039a	κ 03ba					
Λ 039b	λ 03bb					
M 039c	μ 03bc	μ 00b5	μ 2298			
N 039d	ν 03bd					
Ξ 039e	ξ 03be					
Ο 039f	ο 03bf	ό 03cc				
Π 03a0	π 220f	π 03c0				
P 03a1	ρ 03c1					
Σ 03a3	σ 03c3	ς 03c2				
T 03a4	τ 03c4					
Υ 03a5	υ 03c5	ύ 03cd	ÿ 03ab	ü 03cb	ű 03b0	
Φ 03a6	φ 03c6					
Χ 03a7	χ 03c7					
Ψ 03a8	ψ 03c8					
Ω 03a9	ω 03c9	ώ 03ce				
(0028	) 0029					
[005b	⌈ 23a1	⌊ 23a2	⌋ 23a3			
			⌋ 005d	⌋ 23a4	⌋ 23a5	⌋ 23a6
{ 007b	} 007d					
↵ 2430	↵ 2431	↵ 2432	↵ 2433			
+ 002b	+ 208a	+ 216a	± 00b1			
- 002d	- 208b	- 2072	- 216b	± 2213		
× 00d7	• 00b7	• 2219	• 2218	* 002a	* 208f	
/ 002f	\ 005c					
^ 005e						
,	002c					
.	002e	... 2026				
! 0021	¡ 00a1					
? 003f	¿ 00bf					
:	003a	: 2236	÷ 00f7			
;	003b					

' 0027	‘ 2018	’ 2019	, 201a	‘ 201b	246	
" 0022	“ 201c	” 201d	„ 201e	” 201f	« 00ab	» 00bb
© 0040						
_ 005f	⌘ 2427					
~ 007e						
→ 2192	➔ 21c0					
← 2190	➔ 21b6					
↑ 2191	⬆ 21e7	⬆ 21c8	⬆ 242b			
↓ 2193	⬇ 21e9	⬇ 21ca				
↗ 21c4						
↕ 2195						
≡ 2263						
¬ 00ac						
⌘ 2227	⌘ 2228	⌘ 22bb	⌘ 22bc	⌘ 22bd		
& 0026						
007c	2223	2224	2225	2226		
« 226a	< 003c	≤ 2264	≡ 2261	:= 2254	= 003d	≈ 2243
	≈ 2248	≡ 2258	≡ 2259	≠ 2260	≧ 2265	> 003e
						» 226b
% 0025	\$ 0024	€ 20ac	¢ 00a2	£ 00a3	¥ 00a5	₹ 00a7
✓ 221a	α 221d					
∞ 221e	∞ 209e	∞ 209f				
∫ 222b	∫ 222c	∫ 222d	∫ 222e	∫ 222f	∫ 2230	
⊙ 2299	⊙ 229a					
⊕ 2295						
⌊ 221f	⌊ 22a5					
∠ 2220	∠ 2221	∠ 2222				
[2308	] 2309					
[230a	] 230b					

²⁴⁶ Please 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.

	2399		231b		231a		23f1		242a		242c		242f
											2434		2298
#	0023												
UK	242d	us	242e										
V	2200	ð	2202	Э	2203	ђ	2204	ø	2205	Δ	2206	▽	2207
		€	2208	¢	2209	Э	220b	ђ	220c	и	2229	U	222a
✓	2713												

Stack Lift Disabling Functions

For sake of backward compatibility, we decided to keep these 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+, Σ-, M.EDI, M.EDIN, M.←, M.↑, M.→, and M.↓ **disable** ASL, **all other functions enable** it. But compare INPUT on p. 48.

Stack Size

In 2013, *stack* size was discussed for this project. An *RPL*-like ‘infinite’ *stack* would allow for saving (pushing) everything thereon before calling a (sub-) routine, and popping it after RTN; though it makes traditional R↓, R↑, and top level repetition obsolete (and FILL as well). In this context I suggested two new commands CLOSES and OPENS: the first 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; OPENS for opening this section thereafter.

At the bottom line, however, the 8LS turned out being sufficient for solving every real-world mathematical, scientific, or engineering problem (→ [OMS 1](#) and field experience with [WP 34S](#) and [WP 31S](#) since 2011). Thus, we decided to stick to classic *RPN* as implemented on the [WP 34S](#) and [WP 31S](#): it simply covers everything needed most easily. For special requirements, STQSTK and RCLSTK are provided – feel free to use them when they help your endeavors.

Stack overflow is another topic reappearing in discussions every once and a while. Principally, it can happen in a traditional 4LS (→ [OMS 1](#)) – and according to *Murphy’s law* such an overflow will occur in a longer calculation when you won’t expect it, of course. All examples I know circumvent that problem by using a GP register for critical cases, but this presumes you watch the stack continuously.

At the bottom line, **setting SSIZE8 is easier** – from then on you can *calculate sans soucis*.™

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;

Eventually, we refrained from implementing such commands since we had doubts about the sensibility of mixing keystroke and structured programming paradigms. – Recently, this topic is discussed again (after over 10 years of absolute silence), now for the *R47*.

UNDO

In 2013, UNDO was planned for *WP43* as it works in *HP-48*, recalling just the *stack* as it was before executing last command. The *WP 31S* of 2014, on the other hand, features an UNDO recalling the entire calculator state as it was before. Until 2020, we assumed that such a complete UNDO was viable in *WP43* as well, but the necessary overhead turned out forbidding.

Though since

- any *user flag* altered inadvertently can be reverted easily and
- any wide-reaching clear command (**CLREGS**, **CLFALL**, etc.) asks for confirmation before executing,

we implemented **UNDO** recalling the *stack* plus the statistics storage (i.e. **STATS**) and all *system flags* as they were before executing the last command, for a better user experience (note this specification also allows for undoing **Σ-**).

Being at it, undoing the previous program step while going back in program memory (via **BST**) is not viable. It would require saving also the previous content of the *registers* and *variables* for each operation affecting them. Think of undoing a delete or clear operation, for instance, and you get an idea.

It turned out that undoing **ENTER↑** and **EXIT** can be ambiguous. The **WP 31S** could not serve as model here since the functionality of **EXIT** there deviates from the one on **HP-42S** and **WP43**. Thus, we specified these functionalities for your **WP43** in the way that **(ENTER↑)**  equals **EXIT** and **(EXIT)**  equals **NOP** (but with *x* being closed):

1_	1 1	1 ABC_	1 ABC ABC	1 ABC	1 ABC 123_
1	(ENTER↑)	 ABC	(ENTER↑)		123
1_	1 1	1 2_	1 2 2	1 2	1 2 123_
1	(ENTER↑)	2	(ENTER↑)		123
1_	1 1	1 2_	1 2 2	1 2	1 2 123_
1	(EXIT)	2	(EXIT)		123

What Is Open Still

All *items* except the print commands are implemented and work within their domains as far as I am aware. So we achieved a serious scientific calculator.

Some issues, sorted according to their priority, need further care:

1. Power consumption: The code shall be optimized to minimize battery drain while keeping execution times low.
2. The *eigenvector* code should be optimized along the lines of **C47**.
3. There is a glitch in sine returning less than 34 digits near odd multiples of π .
4. There are some unexpected results very close to the maximum *long integer*.
5. Transfer of recent **C47** improvements if worthwhile.

6. Option: The print commands may be implemented.

7. Option: Input of *rich text* files as proposed on pp. 346ff.

If you know of any further issues, please bring them to our attention.

DRAFT

FOR YOUR CONVENIENCE

Please find here the display frame for your WP43 as we designed it (printed to scale) and below its virtual keyboard in *alpha input mode (AIM)* plus the branching helper. Overleaf, the original WP43 keyboard is printed to scale.

If you bought your WP43 from SwissMicros off the shelf, keys and bezel are printed properly (except shifted 7 and -shifted 8) which we populated after the pilot run), and you find the b/w print shown bottom right here laser-engraved on its backside for your reference (almost – note we added a few positions since 2022-10, → 351, and the website is missing).

If you are not among the happy few owners of a WP43 pilot but flashed a DM42 instead, take the keyboard print overleaf, cut it out, stretch it over the DM42 keyboard and fix it, e.g., with some scotch tape. This is the fastest, easiest and least expensive way to operate WP43 on a converted DM42. And it adds protection of the keyboard (compare the vintage HP-41C keyboard covers).

Alternatively, you can get an adhesive overlay and 17 key stickers properly printed, cut, and punched from Jeff (→ 246).

WP 43S

RPN SCIENTIFIC

RESET

A Aα

A

Bβ

B

[Γ γ

C

] Δ δ

D

e^ E ε

E

√ Φ φ

F

ASNI Γ γ

CH X x

↓ I i

↑ H η

SNAP

USER

G

H

I

J

f

g

↔ K k

± Λ λ

⇧ M μ

↩ CLR

ENTER↑

K

L

M

←

/ N ν

7 Ω ω

8 Π π

9 O o

QU

N

O

P

Q

:

× P ρ

4 Σ σ

5 Τ τ

6 Θ θ

R

S

T

U

▲

- ≈

1 Ψ ψ

2 Ξ ξ

3 Υ υ

V

W

X

Y

▼

+ Z ζ

0

⏎

÷

!

CATALOG

Z

?

,

_

EXIT

www.wp43s.com

R = ccccc.ffff

CCCCC +ii ≤fff

ISG

CCCCC ±ii >fff

DSE

CCCCC -ii ≤fff

ISE

CCCCC +ii <fff

DSL

CCCCC ±ii ≥fff

CCCCC -ii <fff

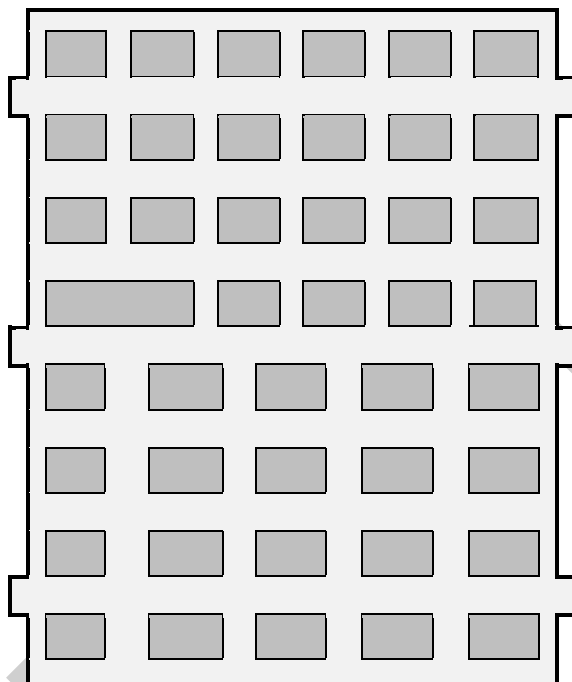
Or look in <https://gitlab.com/rpncalculators/wp43/-/tree/master/res/artwork> for the file WP43_layout.svg; open it using, e.g., GIMP or INKSCAPE, and hide all its layers except "Bezel blue", "Bezel gold", "Bezel grey", "Overlay", and "Background". Then print to scale on suitable material, cut, and apply all yourself (→ 186f).



Furthermore, here is an explanatory picture for specific *short integer* operations taken from the back of an *HP-16C* (with the letter **C** denoting CARRY and **G** standing for OVERFL).

	C	G	
	x	x	
	--	x	
	x	x	RMD≠0 → C
	x	--	RMD≠0 → C
CHS ABS	--	x	
DBLX	--	o	y · x → (X&Y)
DBL÷	x	o	(Y&z) ÷ x → X; RMD≠0 → C
SL	x	--	
SR	x	--	
ASR	x	--	
RL	x	--	
RR	x	--	
RLC	x	--	
RRC	x	--	

From the *USA*, here is a simple template for a quick and easy overlay (almost to scale, → point 3 below):



1. Print it on standard copy paper (80 g/m²). Fill in labels if, where, and as required.
2. Laminate with 80 µm stock (resulting in a thickness of about 0.25 mm).
3. Cut out the perimeter. Note the protruding tabs (a.k.a. tongues) as drawn here are too wide and too long (compare previous page); cut smaller. For the voids, a standard 6 mm square punch might be helpful (→ 186) since punches fitting our needs exactly will hardly be available off shelf.