

## WP43 OWNER'S MANUAL

This manual documents WP43, a free scientific software for portrait calculator hardware of SwissMicros. You can redistribute WP43 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 the GNU General Public License at 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. Stay informed by watching https://gitlab.com/rpncalculators/wp43 .

Copyright © 2015-2023 Walter Bonin, Borstr. 1d, 01445 Radebeul, Germany
All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without prior written permission of the author. For the time being, the locations highlighted cyan are open issues - information is missing there or needs further discussion and investigation to be determined. Any contributions in these matters are highly appreciated.

HP is a registered trade mark of Hewlett-Packard.
The photograph on p .18 was published by NASA in 1969. All paintings, the drawings on pp. 36 (top), 37 (top), 39, 87, 143f, 146, 165, 168-169, 224, 257, 265, 284-284, 330, and $334-338$, as well as the pictures on pp. 56,210 f, and 344 f were taken from various calculator manuals and advertisements published by HP between 1974 and 1989 (some were refined by me). The diagrams on p. 104 are based on material found in Wikipedia. The sources of the maps on pp. 98 and 298 are stated there - I found them by chance. All WP43 keyboard graphics as well as the other photographs, pictures, graphics, and diagrams printed in this manual were created by the author.

Internet addresses are specified as found and verified at 2021-07-21 (just the map on p. 98 is not online anymore). Please note internet addresses may change without notice at any time.

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

Ooh, and you may find some barbed remarks in the text, reflecting the author's impressions watching the world for several decades. No personal offense intended.

Printed in the USA

ISBN-13: 978-172950098-9
ISBN-10: 172950098-6

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 units was manufactured on SwissMicros' production line in 2022-10. WP43 could be purchased off the shelf - please turn to https://www.swissmicros.com/.
Alternatively, you can create a WP43 most easily by flashing a DM42 you own; this way costs next to nothing but you will need some key stickers and an overlay inevitably then. Please consult App. F of the WP43 Reference Manual for what and how to do.

The WP43 software package includes a free full size Simulator (see App. E of said manual), so you may check it out on your computer before buying any hardware. And later on, you may test your programs there as well before sending them to your WP43 calculator. Function sets are almost 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! ..... 9
Print Conventions and Common Abbreviations ..... 13
Section 1: Getting Started ..... 15
Problem Solving, Part 1: First Steps ..... 19
How to Enter Common Numbers (and How to Edit Them) ..... 25
How to Enter and Execute Commands ..... 26
Menus - Items à la carte ..... 27
How the Keyboard is Organized ..... 29
Problem Solving, Part 2: Elementary Stack Mechanics ..... 33
Looking Closer at the Automatic Stack ..... 40
Problem Solving, Part 3: Advanced Stack Mechanics ..... 44
Special RPN Tricks, \#1: Top Stack Register Repetition ..... 51
Special RPN Tricks, \#2: LASTx for Reusing Numbers ..... 52
Error Recovery: © , EXIT, and [ $\curvearrowleft$ ] ..... 53
Clearing and Resetting Your WP43 ..... 55
Addressing and Manipulating Objects in RAM ..... 56
Addressing Tables ..... 63
Indirect Addressing - Working with Pointers ..... 67
Store and Recall Arithmetic ..... 67
Section 2: Dealing with Various Objects and Data Types ..... 71
Some Display Basics ..... 71
Data Types Supported ..... 72
Processing Data Types ..... 74
Recognizing Calculator Settings and Status ..... 79
Getting Special Information: RBR, STATUS, VERS, etc. ..... 82
Localising Numeric Output ..... 83
Real Numbers: Changing the Display Format ..... 85
Reals: Squares and Cubes and Their Roots ..... 87
Reals: Percent Change ..... 91
Reals: Logarithms and Powers (a.k.a. Antilogs) ..... 92
Reals: Hyperbolic Functions ..... 101
Reals: Probabilities - Factorials, Combinations, Permutations, and Distributions ..... 102
Reals: 1D Statistics - Sampling Data, Calculating Means, Standard Deviations, and Confidence Limits ..... 105
Reals: 2D Statistics - Curve Fitting and Forecasting ..... 110
Reals: Chi-Square Statistic - Checking Expectations ..... 115
Real Numbers: Some Industrial Problems Solved ..... 117
Reals: Frequencies and Histograms ..... 126
Reals: Summary of Functions ..... 129
Rational Numbers (Fractions) ..... 136
Angles and Trigonometric Functions ..... 139
Mixed Calculations: Coordinate Transforms in 2D, Flight Directions, Courses over Ground, etc. ..... 142
Angles: Summary of Functions ..... 148
Integers: Input and Displaying ..... 149
Integers: Bitwise Operations on Short Integers ..... 154
Integers: Arithmetic Operations ..... 157
Integers: Overflow and Carry with Short Integers ..... 159
Integers: Summary of Functions ..... 162
Complex Numbers: Introduction ..... 164
Complex Numbers Used for 2D Vector Algebra ..... 168
Complex Numbers: Summary of Functions ..... 170
Vectors and Matrices: Introduction and Input ..... 173
Vectors \& Matrices: Displaying and Editing Larger Objects ..... 177
Vectors \& Matrices: Complex Stuff ..... 180
Vectors \& Matrices: Calculating ..... 181
Vectors \& Matrices: Solving Systems of Linear Equations ..... 186
Vectors \& Matrices: Eigenvalues and Eigenvectors ..... 188
Vectors \& Matrices: Dealing with Statistical Data ..... 192
Vectors \& Matrices: Summary of Functions ..... 193
Dates ..... 197
Times ..... 199
Proposal for Configuration Setting ..... 202
Alpha Input Mode: Introduction and Virtual Keyboard ..... 203
Alpha Input Mode: Entering Simple Text and More ..... 205
Combining Text Strings and Numeric Data ..... 207
Working with Alphanumeric Strings ..... 208
Section 3: Programming ..... 212
Recording a New Routine ..... 214
Labels ..... 218
Editing a Routine ..... 220
Running a Routine from the Keyboard (also for Debugging) ..... 222
Subroutines: Running a Routine from another Routine ..... 223
Automatic Binary Testing and Conditional Branching ..... 224
Loops and Counters ..... 228
Programmed User Interaction and Dialogues ..... 232
Solving Differential Equations ..... 236
The Programmable Menu (MENU) ..... 242
Basic Kinds of Program Steps ..... 243
Deleting Programs ..... 246
Flash Memory (FM) ..... 246
Input and Output of Data and Programs ..... 247
Local Data ..... 250
Section 4: Advanced Problem Solving ..... 252
Programmable Sums ..... 252
Programmable Products ..... 253
Solving Quadratic Equations ..... 254
Arbitrary Equations ..... 255
The Solver ..... 257
Interactively Solving Arbitrary Algebraic Equations ..... 257
Interactively Solving Expressions Stored in Programs ..... 262
Using the Solver in a Program ..... 264
Integration ..... 265
Numeric Integration of Arbitrary Algebraic Equations ..... 265
Interactively Integrating Expressions Stored in Programs ..... 268
Using the Integrator in a Program ..... 269
Differentiating ..... 271
Differentiating Arbitrary Algebraic Equations ..... 271
Interactively Differentiating Expressions Stored in Programs ..... 273
Computing Derivatives in a Program ..... 274
Nesting Advanced Operations ..... 274
Section 5: Two Browsers, two Applications, and two Special Menus ..... 277
The Browsers RBR and STATUS ..... 277
The Timer Application ..... 280
The Time Value of Money (TVM) ..... 282
The Catalog of Constants ..... 287
Unit Conversions ..... 292
Section 6: Creating Your Very Personal WP43 ..... 299
Assigning Your Favorite Functions ..... 300
Creating Your Own Menus ..... 304
Purging User-Defined Items ..... 306
Assigning Your Favorite Characters ..... 307
Launching Your Personal User Interface (User Mode) ..... 308
Appendix 1: Key Response Table ..... 311
Context Sensitive Keys ..... 322
Virtual Keyboards ..... 326
Appendix 2: Operator Precedence ..... 328
Appendix 3: Further Applications of TVM ..... 329
Ordinary Annuities (a.k.a. Payments in Arrears) ..... 331
Annuities Due (a.k.a. Payments in Advance) ..... 335
Appendix 4: Graphics ..... 339
Appendix 5: Power Supply ..... 343
Appendix 6: Time Line of Quoted Manuals ..... 344
Appendix 7: Release Notes ..... 346
Index ..... 348

## WELCOME!

Congratulations, you are holding your new WP43 in your hands now, a fast, compact, reliable, and solid problem solver like you never owned before - instant on, fully programmable, incorporating a state-of-the-art high-resolution display, user-customizable, connecting to your computer, and $R P N^{1}$ - a serious scientific instrument supporting you in your studies and professional activities. It readily provides advanced capabilities never before combined so conveniently in a true pocket-size calculator:

+ A Solver (root ${ }^{2}$ finder) that can solve for any variable in any equation.
+ An Integrator for computing definite integrals of arbitrary functions.
+ Numeric derivation, programmable sums and products.
+ A wealth of functions, operating on real and complex numbers, fractions, integers, dates, times, and text strings.
+ A comprehensive set of statistical operations, including probability distributions, data analysis functions, curve fitting, forecasting, and measuring system analysis.
+ Vector and matrix operations in real and complex domain, including a comfortable Matrix Editor, dot and cross products, eigenvalues, eigenvectors, and a solver for systems of linear equations.
+ Base conversions and integer arithmetic in fifteen bases from binary to hexadecimal; bit manipulations in words of up to 64 bits.
+ An easy-to-use menu system assigning up to eighteen operations to the top row of keys according to your actual needs.
+ Easy control by named system flags and variables.
+ An USB socket allowing for auxiliary external power supply as well as for program exchange with your computer; so you can edit, debug, and test routines there and return them verified.

[^0]${ }^{2}$ TNG: Root bedeutet hier Nullstelle.

+ Field proven keystroke programming incl. branching, loops, conditional tests, flags, subroutines, and program-specific local data.
+ Alphabetically searchable catalogs to find all items stored in memory be they provided by us or created and programmed by you.
+ A user interface you can customize: You can create and save various custom keyboard layouts on-board and recall them one by one as you need them. Dedicated keyboard overlays are supported.
+ Battery-fail-safe on-board backup for all your data and settings. State files you can store, recall, and exchange with other users.
+ A timer (or stopwatch) based on a real-time clock.
+ An infrared port for immediate recording of results, calculations, programs, and data using e.g. an HP 82240A/B Infrared Printer.

WP43 provides the amplest function set ever seen in an RPN calculator:

+ Almost 700 functions, including Euler's Beta and Riemann's Zeta, Lambert's W, Bessel functions of $1^{\text {st }}$ and $2^{\text {nd }}$ kind, Bernoulli and Fibonacci numbers, Jacobi elliptic functions and integrals, as well as Chebyshev, Hermite, Laguerre, and Legendre polynomials (no more need for table books or computer software in these matters).
+ 14 probability distributions: general normal (Gaussian), Student's $t$, chi-square, Fisher's F, Poisson, binomial, geometric, hypergeometric, Cauchy-Lorentz, exponential, logistic, Weibull, and more.
+ 10 curve fitting models featuring two or three parameters (linear, orthogonal, exponential, logarithmic, power, root, hyperbolic, parabolic, Cauchy and Gauss peak).
+ 53 fundamental physical constants as precise as used today by national standards institutes like NIST or PTB (following CODATA 2018); furthermore 24 important mathematical, astronomical, and surveying constants.
+ 126 unit conversions, mainly from old British Imperial to universal SI units and vice versa.
+ A complete set of financial functions and applications for the inevitable business matters.

Furthermore, your WP43 features lots of space for your data, results, programs, and ideas:

+ A high-resolution low-power dot-matrix display showing crisp results
and menus, allowing for natural matrix display as well as for a wide variety of mathematical symbols, Greek and international Latin letters.
+ Your choice of workspace comprising 4 or 8 stack registers and up to 107 global general purpose registers. Each register can take any object of arbitrary data type (be it a matrix, a vector, a text string, or any number of arbitrary kind).
+ Up to 1000 named variables. Also each such variable can take any object of arbitrary type.
+ 112 global user flags and 39 named system flags.
+ 32 local user flags and up to 100 local registers per routine, allowing e.g. for recursive programming.
+ Up to 10000 program steps per program, a total of some 20000 in RAM. Battery-fail-safe flash memory for saving your routines.

Your WP43 is a true pioneer: the very first entirely communitydesigned and -built RPN pocket calculator. All its hardware, firmware, and user interface were thoroughly thought through, discussed, designed and assembled, written and tested by us over and over again. WP43 is the result of a collaboration of two international teams: SwissMicros (https://www.swissmicros.com/) - i.e. Swiss Michael Steinmann and Emy Amstein, and Czech David Jedelsky - created the hardware ${ }^{3}$ and DMCP, while French Martin Lorang and Didier Lachieze, Mihail from Japan, South African Jaco Mostert, British Benjamin Titmus, Australian Paul Dale, and I designed the user interface and wrote the software. ${ }^{4}$
As our WP 34 S and WP 315 before, also WP43 is a hobbyist's project. A first draft was published in November 2012, and it was then discussed on https://www.hpmuseum.org/forum/forum-8.htm/ until 2016 and on

[^1]https://forum.swissmicros.com/ from 2017 on. ${ }^{5}$ Prototypes of SwissMicros' portrait DM42 hardware were publicly presented first on HHC2016 in Nashville (USA) and on the Allschwil Meeting 2016 (Switzerland). Martin and I presented the first version of the WP43 Simulator on the Allschwil Meeting 2018. The Allschwil Meeting 2022 saw the pilot run of WP43 calculators made. We thank the participants of said meetings and all members of the international community who contributed their ideas, put their votes, and lent their support at various phases throughout this project. We greatly appreciate your contributions!
We baptized our baby in honor of the HP-42S of 1988, the most powerful $R P N$ pocket calculator available before these activities. ${ }^{6}$ May it be a worthy and valiant successor of the HP-42S - though we would have preferred HP making it (the HP as we knew it controlled by Bill Hewlett and David Packard until the Eighties of last century). In any way, WP43 stands in the tradition of 50 years of RPN pocket computing.

We carefully checked all aspects of WP43 to the best of our abilities. Thus we hope it is free of severe bugs. This cannot be guaranteed, however, so we promise to continue improving WP43 whenever necessary. Should you discover any strange result, please report it to us (口SNAP will take a snapshot of your calculator screen). If it turns out being caused by an internal bug, we will correct the firmware and provide you with an update as soon as possible. As we did and do for WP 34S and WP 31S since 2011, we will continue maintaining short response times.

## Enjoy!

Walter

[^2]For us, a pocket calculator per definition is a device fitting comfortably in your shirt pocket. Marketing people see this feature more elastic - our shirt pockets are not elastic enough for that. Being at it, we generally recommend not to put your calculator in the back pocket of your jeans - beyond frazzling your pocket, it may break or multiply there.

## Print Conventions and Common Abbreviations

- Standard text font in this manual is Arial. Emphasis is added by underlining or bold printing. WP43 COMMANDS, MENUS, PREDEFINED VARIABLES and SYSTEM FLAGS (SET or CLEAR) are generally called by their (item) 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 $\boldsymbol{n}$ denote variables; bold regular characters constant sample values (e.g. specific labels, numbers, or symbols).
Specific terms, titles, trademarks, names, or abbreviations are printed in regular italics, hyperlinks in blue underlined italics. Each link will beam you to its target - since it cannot work in the printed copy, it generally refers to a page number, to the Table of Contents, or to 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. So e.g. register $\mathbf{Y}$ contains the value $\boldsymbol{y}$ and $\mathbf{R 4 5}$ contains $\boldsymbol{r} 45$. Overall stack contents are generally quoted in order $[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \ldots]$. We kept the term register for the space where a particular object is stored, although the actual size of such a register may vary widely following the size of the object stored therein.
- This KEY font (created by Luiz Vieira of Brazil) denotes calculator KEYS, incl. SOFTKEYS in general. For shifted operations like GTO or SF, the respective colors are used. Alphanumeric and numeric outputs (like $1.234 \times 10^{-56}$ or $7,089 \cdot 10^{12}$ ) are usually printed as seen on your calculator screen.
- Courier is used for file names and describing numeric formats.
- Regarding mathematical symbols and notation, we follow ISO 80000-2 striving to comply with standards agreed on internationally. Thus, we avoid :wherever possible. We chose decimal points, multiplication crosses, and division slashes unless ambiguous - you may choose decimal commas, multiplication dots, and division colons as well, of course.

All this holds unless stated otherwise locally.
The following abbreviations, sorted alphabetically, are used throughout WP43 manuals - find detailed information about the respective terms at the locations referred to or in the Index on pp. 348f, if applicable:

| $1 D, 2 D, 3 D$ | $=$ one-, two-, three-dimensional. |
| :--- | :--- |
| ADM | $=$ angular display mode. |
| AIM | $=$ alpha input mode. |
| App. | $=$ Appendix. |
| $D T$ | $=$ data type. |
| FM | $=$ flash memory. |
| f. | $=$ footnote. |
| GP | $=$ general purpose. |
| HP | $=$ Hewlett-Packard. |
| $I O I$ | $=$ Index of all Items provided in WP43 (see ReM below). |
| $O H$ | $=$ Owner's Handbook. |
| OHPG | $=$ Owner's Handbook and Programming Guide. |
| $O M$ | $=$ (WP43) Owner's Manual. |
| $O M S \ldots$ | $=$ OM Section ... |
| $P E M$ | $=$ program-entry mode. |
| $R A M$ | $=$ random access memory, allowing read and write operations. |
| $R e M$ | $=$ WP43 Reference Manual, comprising the IOI and more. |
| $R e M S . .$. | $=$ ReM Section ... |
| $R P N$ | $=$ Reverse Polish Notation (cf. f. 1 on p. 9). |
| $R U M$ | $=$ run mode. |
| $S I$ | $=$ système international d'unités, a coherent system of units of |
|  |  |
| $T A M$ | $=$ transient alpha mode. |
| $T N$ | $=$ translator's note (and TNG = TN for German readers). |

Further abbreviations are listed in the Index. A few more may be introduced and used locally.

Finally: WARNING indicates risk of severe errors. There are only three warnings in this manual. Resetting your calculator by actuating the RESET button on its rear will help in almost all cases - though it will erase all your data in RAM.

[^3]
## SECTION 1: GETTING STARTED

At its heart, your WP43 is an extremely powerful, versatile problemsolving tool. It allows you to solve even very elaborate mathematical and computational problems in either of two different ways:

- Manual problem solving: Using the calculator's RPN logic system, you can manually work step-by-step through the toughest problems while seeing intermediate answers after each and every step of the way. The advantages of RPN become particularly apparent when working with exploratory type problems where intermediate answers are an important part of the problem solving process.
- Programmed problem solving: Your WP43 can remember any sequence of keystrokes you entered, and it can then run it repeatedly as often as you need it. This simple programming paradigm is particularly useful in providing answers to repetitive problems that require different inputs. Advanced programs may also be written for solving more elaborate tasks, e.g. iterative computations containing automatic decisions and branching. Thousands of keystrokes can be recorded and exchanged with your computer or laptop.

SAFETY WARNING: Your WP43 is not designed to be used by children under 3 years.
This is not a toy. Do not use it before you can tell left from right, read, and do mental arithmetic reliably. it contains small parts which if swallowed are hazardous for health. Swallowing the battery can be fatal within 2 hours - seek medical advice immediately! ${ }^{8}$
Do not use your WP43 for any other purpose than specified above (e.g. neither as a hammer, nor as a lever, nor as a door stop, nor as a missile); else you may destroy it and/or other objects easily, even hurt animals or human beings including yourself. ${ }^{8}$
Your WP43 shall be operated in a clean, dry environment (relative humidity less than $35 \%$ ) at ambient temperatures from 5 to $40^{\circ} \mathrm{C}\left(41\right.$ to $\left.104{ }^{\circ} \mathrm{F}\right)$.
Do not drop your WP43 on solid ground - it may break. Do not leave it

[^4]lying in the sun; its dark surface may become hot, and hot surfaces may cause burns. Do not leave your WP43 lying in the cold; humidity may condense on its surface when a cold WP43 is put in a warmer space. ${ }^{8}$
Should your WP43 become wet, turn it off, remove the battery (see Disassembly below), and let your WP43 dry for sufficient time before turning it on again. Do not try to accelerate drying by blowing hot air (exceeding $60^{\circ} \mathrm{C}$ or $140^{\circ} \mathrm{F}$ ) into your WP43 or by putting it into a oven (neither conventional nor microwave) or the like - you may destroy it. ${ }^{8}$
Disassembly: Do not disassemble your WP43 unless you are qualified for such work and have the proper tools handy. You will need 1 small Phillips screwdriver and 2 hands for opening it. ${ }^{8}$
Inserting / Replacing the Battery: Your WP43 contains a battery. When it runs out of power, the symbol will appear top right on its screen. Then open your WP43 (cf. Disassembly above) and replace its battery by a fresh quality cell. Do not operate your WP43 without a good battery installed. Dispose of the old battery responsibly in the appropriate in-store containers or at your local disposal center; neither take it apart nor throw it in a bin nor in fire nor in your environment.
Disposal of the calculator: Your WP43 must not be disposed of along with household waste. Remove its battery (cf. Inserting / Replacing the Battery above) and dispose of your WP43 according to applicable laws and regulations at your electronics collection point.

If you know how to deal with a good old HP RPN scientific calculator, feel free to start using your WP43 right away. You will find it easy and intuitive to use. Browse this manual to learn about some fundamental design concepts putting your WP43 ahead of previous scientific pocket calculators.

On the other hand, if $R P N$ is new to you, we recommend going through OMS 1 and 2 thoroughly. This will enable you to easily solve problems manually, benefitting from this unique logic system implemented. Once learned, RPN forms a lifelong lasting, reliable basis of your work.

Most commands on your WP43 work as they did on its antecessors, in particular on WP 34S, WP 31S, and HP-42S. The OM and ReM are designed to supplement your prior knowledge, focussing on the new features of your WP43 and providing information about them. They include also some formulas and technical explanations; though they are
not intended to replace textbooks on mathematics, statistics, physics, engineering, economy, computer science, or programming.

The following text starts presenting the keyboard of your WP43 to you, so you learn where to find what you are looking for. It continues demonstrating basic calculation methods, the memory of your WP43, and addressing objects therein.

OMS 2 then covers the display and its indicators giving you feedback about what is going on. Furthermore, the various data types supported by your WP43 are introduced and demonstrated comprehensively.

Programming your WP43 for automatic problem solving (as shown in OMS 3) follows field proven concepts known from successful previous pocket calculators up to and including the HP-42S and WP 34S.

OMS 4 and 5 present advanced functionalities implemented. If you want to customize the user interface of your WP43 according to your very personal preferences you will find everything about it in OMS 6.

This $\mathbf{O M}$ is supplemented by the $\operatorname{ReM}$. A major part of the latter is taken by the IOI, i.e. the Index of all Items available, what they do, and how to call them. It also comprises full information about all menus, system flags, constants, and variables provided. The ReM closes with appendices covering special topics (like memory management, the WP43 Simulator for your computer, and advanced mathematical functions implemented). Find there also how to keep your WP43 up-to-date whenever new firmware revisions will be released.

Continue using both manuals for reference.
Before diving into the OM, here is something we ask you to keep in mind:

> Your WP43 is designed to support you solving analytical and calculatory problems. But it is just a tool: it can neither think for you nor check the sensibility of a problem you apply it to. Gather information, think before and while keying in and calculating, and check your results: these tasks will remain your responsibility always. We will not be liable for any of your results.


This iconic picture shows our Earth rising over the Moon, pictured by the crew of Apollo 8. Remember orbiting the Moon and even landing on it was entirely accomplished before a single scientific pocket calculator was launched on Earth (most results were achieved by careful slide rule calculations then).
Imagine how much more is within your reach using precise tools like your WP43.

## Problem Solving, Part 1: First Steps

Start exploring your WP43 by turning it on: Press its bottom right key notice that ON is printed below that key. Turned on the very first time (or after a reset), your WP43 will show a display like this:


To turn it off again, press $\square$, then EXIT (note OFF printed below it). Since your WP43 features Continuous Memory, turning it off and on does not affect the information contained. For conserving battery power, your WP43 will even shut down automatically after ten minutes of inactivity turn it on again and you can resume your work right where you left off.

This works as on preceding calculators (e.g. WP 34S or HP-42S). Your WP43, however, looks cleaner than a WP 34S while more colorful than an HP-42S. This is due to your WP43 featuring two general prefixes $\square$ and and menus - offering you up to six functions per key.

Looking at an arbitrary one of its 41 black keys, white print is for its primary function. For additional functions, golden or blue labels are printed on the faceplate above 36 keys, and grey characters bottom right of 29 keys.

For calling a primary function, just press the corresponding key. For a golden function, press $\square$ first. For a blue function, press $\square$ first.

Take the key $\boldsymbol{X}$, for example. ${ }^{9}$ Pressing...

- $\boldsymbol{x}$ alone executes a multiplication.
- $\square+\boldsymbol{x}$ calls $x$ ! for the factorial (e.g. enter 6 (x! and you will get 720).
- $\square+\boldsymbol{x}$ opens (PROB), a menu (or collection) of functions related to probability ( $\boldsymbol{\Delta}$ or $\nabla$ will show more of it,_and EXIT will leave the menu). All labels printed underlined on the faceplate refer to menus.
- The grey $\mathbf{R}$ will become relevant with alphanumeric data.


Note that all the functions and labels printed on the keyboard are explained in App. 1 on pp. 311ff.

## Time for a first problem solving example:

Turn your WP43 on again if necessary. Press $\boldsymbol{\oplus}$. Your display will show 0 . in each of its four numeric rows then.

Now, let's assume you want to fence a little patch of land, ${ }^{10}$ rectangular, 40 yards long and 30 yards wide. ${ }^{11}$ You have already set the $1^{\text {st }}$ corner post

[^5](A), and also the $2^{\text {nd }}(\mathrm{B})$ in a distance of 40 yards from A . Where do you set the $3^{\text {rd }}$ and $4^{\text {th }}$ posts ( $C$ and $D$ ) to be sure that the fence will form a proper rectangle? Simply key in:
\[

$$
\begin{aligned}
& 0 . \\
& 0 . \\
& 0 .
\end{aligned}
$$
\]

Note the input cursor = just vanished and 30 is adjusted to the right, indicating this input is closed now; so the next number can be entered:


$$
40
$$

$$
\vartheta=
$$

$$
r=
$$

$$
\begin{array}{r}
0.0 \\
36.9^{\circ} \\
50.0
\end{array}
$$

All you need now is the number in the bottom row, a friend, and 80 yards of rope: Ask your friend to hold both ends of the rope firmly for you, take the loose loop and walk away as far as you can - when the loop is stretched, mark that position on the rope and return to your friend. Ask your friend to hold this marked point of the rope as well, fetch the two loose loops and walk again as far as possible - when both loops are stretched, mark the positions on the rope. Return again; hand over the two new points and walk away once more, now with four loose loops. After marking as before, your

[^6]Generally, we will print no more than one display row comprising just zero from here on for space reasons.

rope will show marks every 10 yards. ${ }^{13}$
Now, take this rope and nail its one end on post $A$ and the other end on $B$, fetch the loose loop and walk 5 marks away as calculated. When both sections of the rope are tightly stretched, stop and place post C there. You may set post $D$ the same way on the other side.

This solving method works for arbitrary rectangles, whatever distances may apply (you will appreciate a tapeline instead of a rope in the general case). As soon as you press $\rightarrow \mathrm{P}$, your WP43 calculates the diagonal automatically for you. You just provide the land, posts, rope, hammer and nails. And it will be up to you to set the posts!

Another example, dealing with greater areas: ${ }^{14}$

To calculate the surface area of a sphere, the formula $A=\pi d^{2}$ can be used, where $A$ is the surface area, $\pi$ is $3.1415 \ldots$, and $\boldsymbol{d}$ is the diameter of the sphere.

Ganymede, one of Jupiter's 79 moons, has a diameter of 5262 km . To use your WP43 to manually compute the area of Ganymede, you can press the following keys in order:
5) (2)6 5262
ENTER $\boldsymbol{x}$

diameter of Ganymede square of the diameter 3.1 the constant $\pi$ (rounded to 1 decimal as chosen above)
${ }^{13}$ Something alike, a knotted cord or rope, is known as one of the oldest surveying tools of mankind, used e.g. in ancient Egypt.
${ }^{14}$ Basically quoted from the HP-25 OH though updated following progress in astronomy only 12 moons of Jupiter were known in 1975 (not counting the black cuboidal monolith seen in Stanley Kubrick's masterpiece '2001 - a Space Odyssey' of 1968).
Reading some of HP's vintage calculator manuals may be real fun. Eric Rechlin collected them (and more literature in various languages about HP and 'nearly HP' pocket calculators from 1972 to 2023) and put them at https:///literature.hpcalc.org/ where you can browse and download them for free.

To calculate the area of a sphere using a program, you should first write the program, then you must record the program into the calculator, and finally you run the program to calculate the answer.

Writing the program: You have already written it! A program is nothing more than the sequence of keystrokes you would execute to solve the same problem manually.

Recording the program: To record the keystrokes of the program into the calculator, press the following keys in order:

P/R turns to program-entry mode. The display will change in this course - simply do not bother for the time being.

(LBL) T goes to where free program space begins in memory.


RTN
P/R
is the closing step of your program. Finally,
returns from program-entry mode to run mode.


Any straight program on your WP43 will consist of an opening (LBL) and a closing RTN, framing the keystrokes needed for solving the problem manually.

Running the program: Now all you have to do to calculate the area of any sphere is keying in the value for its diameter and press

## XEQ T (meaning 'execute program $\mathbf{T}$ ').

When you press XEQ T the sequence of keystrokes you recorded is automatically executed by the calculator, giving you the same answer you would have obtained manually:

For example, to calculate the surface area of Ganymede, press


With the program you have recorded, you can now calculate the respective surface area of any of Jupiter's moons - in fact, of any sphere - using its diameter. You have only to leave the calculator in run mode and key in the diameter of each sphere that you wish to compute, and then press XEQ T. For example, to compute the surface area of Jupiter's moon lo with a diameter of 3643 km :

| (3)64] | 3643 |  | 10's diameter |
| :---: | :---: | :---: | :---: |
| (XEQ $T$ |  | 41693486.7 | its surface area; |
| (3) 1 (2)(2) | 3122 |  | Europa's diameter |
| (XEQ $T$ |  | 30620739.2 | its surface area; |
| (4)8)(2) | 4821 |  | Callisto's diameter |
| (XEQ $T$ |  | 73017025.3 | its surface area |

Programming your WP43 is that easy! It remembers a series of keystrokes and then executes them automatically when you press XEQ ... ${ }^{15}$
Thus, there is no need memorizing any complex formula you keyed in once - your WP43 can remember it for you (and provides space for many, many more). And you can even define shortcuts to your favorite formulas by customizing the surface of your WP43 (see OMS 6).
The early portions of this manual show you how easy it is to manually use the power of the WP43; while in Section 3: Programming you will find a complete guide to WP43 calculator programming. Even if you have used other pocket calculators ..., you will want to take a good look at this manual. It explains the unique HP logic system that makes simple answers out of complex problems, and WP43 features that make

[^7]programming painless. When you see the simple power of your WP43, you'll become an apostle just as have some 20 million of RPN calculator owners before you.

## How to Enter Common Numbers (and How to Edit Them)

Numeric entry is as easy as typing. For 12.34567, for instance, press (1) 2 3 (4) 6 and you will see

### 12.34567

You may put in up to 43 digits, echoed immediately in the bottom numeric row on the screen (note the gap inserted automatically after each group of three digits for easier reading). As long as input is open, any digit mistyped may be erased by $\boldsymbol{\leftrightarrow}$ and can be replaced then (when no digit is left, your WP43 returns to the state as it was before this input).


For entering negative numbers like -7.8, key in $7+1 / 9)$ or
 keyed in. Only negative signs will be displayed.

For a huge figure like the age of the universe in years as we know it, just enter 1 B 8 E 9 which is echoed
in 'mantissa plus exponent' format. E stands for 'enter exponent of 10'. Note your WP43 allows for a naturally readable output instead of showing you cryptic machine formats like 13859.

During numeric input, your keystrokes are generally just echoed in the bottom numeric row. Input is closed and released for interpretation by a

## FILL DROP $\downarrow$ ENTER $\uparrow$

 command - e.g. by ENTERT. In the case introduced just above, ENTERT will change the display to the equivalent:13800000000.

Note the number moved to the right (cf. p. 20) and is duplicated.
Really tiny numbers such as a typical diameter of an atom (i.e. 0.0000000001 meter $)$ are entered in full analogy: $E+1 / 0$ will do here. Closed, this will display as

### 0.0000000001

with startup default settings. Note you did not have to enter $10 . E+1 / 0$ here: if no numeric input is heading $E$ then 1 is assumed for the mantissa. And pressing $+/ /$ after $E$ will flip the sign of the exponent - if you want to flip the sign of the mantissa, press $\dagger /$ before $E$ or after closing the entire input. Also note the number above can be displayed far more compact like $1 . \times 10^{-10}$ or even like $\quad 1,10^{-10}$, using other screen formats. ${ }^{16}$

The entire bottom display row may be removed by IDROP $\downarrow$. Subsequent input will then be put in this row pushing old content up. Closed content of the bottom row may be cleared at once by $\boldsymbol{\uplus}$, setting it to 0. - subsequent input will overwrite this zero then.

## How to Enter and Execute Commands

For calling the operation you wish executed, just enter the keystrokes required to access its label (cf. p. 19). Pending command input will be echoed at left end of the top numeric row on your screen until the command is completed. Therein, pending will be echoed by $\mathbf{f}$ or by ${ }^{9}$, if applicable; ${ }^{17}$ these symbols will be replaced by the name of the command accessed as soon as it can be decoded.

For many operations, ${ }^{\mathbf{f}}$ or ${ }^{\mathbf{g}}$ will be the only echo you will see during command entry since next keystroke may terminate command input already (as observed with $\square \square P$ above), call the command, execute it, and display its result,
 all done in a fraction of a second.

[^8]Some commands, however, require trailing parameters and will thus stay on the screen for longer. STO and RCL are of this kind, and there are many more (see pp. 63ff).
$\square, \square$, and menus in particular allow for easy access to a multiple of the 43 primary labels the keyboard of your WP43 could take.

## Menus - Items à la carte

Your WP43 features several hundred operations, far too many for showing all of them on the keyboard. Hence, most operations live in menus. Beyond commands, also applications, browsers, constants, digits, conversions, programs, submenus, symbols (a.k.a. characters), system flags, or variables defined may be stored in menus: we collectively call them menu items or simply items. Menus keep the keyboard relatively tidy.

Your WP43 shows 34 menus on its keyboard. Their labels are printed underlined for easy recognition. ${ }^{18}$ In alphabetic order, these labels are $\underline{A}$, ADV, BIT, CATALOG, CLK, CLR, CONST, CPX, DISP, EQN, EXP, FIN, FLAG, INFO, INTS, I/O, LOOP, MATX, MODE, PART, PROB,

Call any menu by simply accessing its label (cf. p. 19). This will open the corresponding menu and cause the lower third of the WP43 screen (called the menu section) displaying the respective menu view.

Whenever a menu is called the first time, its top view will appear in the menu section. Such a view may comprise up to 18 items:

- up to six assigned to the unshifted,
- up to six to the -shifted, and
- up to six to the $\square$-shifted top row of keys.

[^9]
## Example:

Press EXP and the menu EXP will open showing its functions as pictured overleaf. As long as this menu view is shown, simply press, e.g., ...

| SwissMicros |  | WP43 |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |
| $13800000000 .$ |  |  |  |  |  |
|  |  |  |  |  |  |
| 0.0000000001 |  |  |  |  |  |
| sinh | arsinh | cosh | arcosh | tanh | artanh |
| $\sqrt[3]{x}$ | $\sqrt[3]{y}$ |  | $\ln (1+x)$ | $e^{x}-1$ | $\sqrt{\left(1+x^{2}\right)}$ |
| $x^{3}$ | $y^{*}$ | $\log _{x} y$ | $\mathrm{lb} \times$ | $2^{\text {x }}$ | $\mathrm{x}^{2}$ |
| F1 | F2 | F3 | F4 | F5 | F6 |

- F2 for $\left[y^{\mathbf{x}}\right]$,
- $\square+\boldsymbol{F} \mathbf{2}$ for $(\sqrt[x]{y})$ (note the golden stripes enframing this display row),
-     + F4 for (arcosh), known as the inverse hyperbolic cosine (note the blue stripes).
- If you press $\square+$ F3), however, nothing will happen since no label is displayed there - no operation is linked to [] -shifted F3.

Saving space, we may print e.g. arcosh for a function in the -shifted row, $\sqrt[x]{\boldsymbol{y}}$ for a function in the -shifted row, and $\boldsymbol{y}^{\mathbf{x}}$ for a function in the unshifted row here.

A predefined menu may comprise more than just one view. Such a multiview menu will be indicated by a dashed line above the menu section on display. Whenever a multi-view menu shows up, $\Delta$ will advance to its next view and $\nabla$ will go back to the previous one, changing the labels displayed. Because multi-view menus are circular, also pressing $\boldsymbol{\Delta}$ (or (7) repeatedly will return to the $1^{\text {st }}$ view after all other stored views were displayed (thus, for a menu containing only two views, both $\boldsymbol{\Delta}$ and $\nabla$ will display the next menu view).

Note any menu view will remain on screen - granting direct access to its functions displayed - until you leave it (e.g. via $\nabla$ or $\Delta$, if applicable, via EXIT, or by calling another menu).

Catalogs are special menus: therein, items are sorted alphabetically. They may be also searched this way (see ReMS 2) - thus, there are various possibilities to call a cataloged item; we will hence refer to such
an item using the notation (item). Note picking an item from a catalog (e.g. from CONST) will execute an automatic EXIT (with one exception see OMS 2). We will use the same notation referring to an arbitrary item whose precise location in its menu is irrelevant at the moment.

Else we will use the background colors as explained on previous page from here on to indicate the access path to a specific menu function.

Note that - within a menu - labels of included (sub-) menus are displayed inverted without a 'menu underline' (cf. f. 18). Pressing EXIT in a submenu will return to its parent menu. Calling EXIT from a menu will return you to the menu you called before, if applicable.

## How the Keyboard is Organized

You might have noticed that the labels on your WP43 are grouped according to their purposes. Beyond the ten digits and the four basic arithmetic operations $\oplus, \square, \boldsymbol{x}$, and $\square,{ }^{19}$ five larger sets are provided:

[^10]Softkeys calling items from the menu view displayed above.

Modes, data types, and 'common' transcendental functions. SIN, COS, TAN, and their inverses are collected in TRI.


Functions for programming and calling programs: E.g. XEQ for calling and executing a program, R/S for running or stopping it.

Find the basic functionalities of all primary keys of your WP43 explained overleaf, top left to bottom right on the keyboard. See the sources mentioned (or the $I O I$ ) for details of these functions. There may be further functionalities of these keys not printed here depending on particular types of operands; they will be covered in OMS 2.

Note the lowest numeric row displayed is called $\mathbf{X}$, with its content $\boldsymbol{x}$. The row above is called $\mathbf{Y}$ and contains $\boldsymbol{y}$ (see next chapter for the reasons).
$1 / \times x$ Inverts $\boldsymbol{x}$ (see p. 33 for an example).
EXP Opens a menu containing exponential functions like $\boldsymbol{y}^{\boldsymbol{x}}, 2^{\boldsymbol{x}}$, $x^{2}, x^{3}$, roots, logarithms, and hyperbolics (cf. its contents on p. 28).
TRI Opens a menu containing trigonometric and hyperbolic functions and their inverses (cf. its contents shown on previous page, see also pp. 101f and 139ff).
(In Returns the natural logarithm of $\boldsymbol{x}$ (see pp. 33 and 92ff).
$\boldsymbol{e}^{\boldsymbol{x}}$ Raises $e$ to the power of $\boldsymbol{x}$ (see pp. 33 and 92ff).
$\sqrt{\boldsymbol{x}}$ Returns the square root of $\boldsymbol{x}$ (see pp. 33 and 87 ff ).
STO Stores (copies) $\boldsymbol{x}$ in the destination specified (see pp. 56ff).
RCL Recalls (copies) an object from the source specified into $\mathbf{X}$ (see pp. 43 and 56ff).

R】 Rolls the stack contents one level down (see p. 42).
(CC Composes, cuts, or converts complex numbers (see pp. 164ff).

Prefix to reach golden labels. Press $\square$ twice to cancel it.
Prefix to reach blue labels. Press $\square$ twice to cancel it.
ENTERT Context sensitive key, cf. p. 25 and see next chapters.
$x^{2} \geqslant y$ Swaps the contents of $X$ and $Y$ (see p. 42).
+// If pressed during input of a mantissa or exponent, changes its sign (cf. p. 25). Else multiplies $\boldsymbol{x}$ times -1 .
(E Allows entering an exponent of ten for convenient input of very big or very small numbers (cf. p. 25).
4 Context sensitive key, cf. p. 25 and see next chapters.
(1) ${ }^{19}$ If there is an open question like Are you sure?, enters N for 'no'. Else divides $\boldsymbol{y}$ by $\boldsymbol{x}$.
(7), 8, (9, Enter the respective digit.
(4), 5), 6,
(1), 2),

0

XEQ If there is an open question like Are you sure?, confirms it; else - if in PEM - inserts a call to the subroutine with the label specified;
else (i.e. in RUM) calls the routine with the label specified and starts executing it (see pp. 212ff).

## $\boldsymbol{x}$ Multiplies $\boldsymbol{y}$ times $\boldsymbol{x}$.

$\triangle$ Context sensitive key, cf. p. 28 and see next chapters.

## - Subtracts $\boldsymbol{x}$ from $\boldsymbol{y}$.

(3) If there is an open question like are you sure?, enters $Y$ for 'yes'. Else enters the digit 3.
( Context sensitive key, cf. p. 28 and see next chapters.

## $\pm$ Adds $\boldsymbol{x}$ to $\boldsymbol{y}$.

$\square$ Enters a decimal radix mark in numeric input (cf. p. 25).
If pressed twice in numeric input, allows for entering a fraction (see pp. 72 f and 136ff).

In register or flag addressing, $\square$ heads a local address (see pp. 60ff).
R/S Context sensitive key, see OMS 3.
EXIT/ON Context sensitive key, cf. pp. 28 f and see next chapters.
All the context sensitive keys are comprehensively explained on pp. 323ff.
Before demonstrating the operation of your WP43 in detail, let's return to our introductory problem solving examples for four general remarks:

1. We presume you have graduated from an US High School at minimum, passed Abitur, Matura, or an equivalent graduation. So we will not explain basic mathematical rules and concepts within the OM or ReM. Please turn to respective textbooks.
2. As long as you stay with one coherent set of units while calculating reasonably you will get meaningful results within this set. Thus, it were waste of time and energy to enter any units here and pull them through your calculations. Should you need to convert special inputs
into $S /$ units or require results expressed in particular units, however, $U \rightarrow$ and (x $\Rightarrow$ will help (see pp. 292ff). ${ }^{20}$
3. Although you entered just integers for both edges of your little patch of land in the example on pp. 20f, your WP43 calculated the diagonal using real numbers (see next page). This also allows for decimals in input and output. Alternatively, you may enter fractions such as e.g. $61 / 4$ if this carries a benefit for you. ${ }^{21}$
4. In five decades of scientific computing, a wealth of sample applications has been created and published by various authors more and better than we can ever invent. We did not intend to copy all of them; instead, we recommend the media mentioned in the footnote below. ${ }^{22}$ All computations described there for any scientific calculator can be done on your WP43 - notably faster and often in a more elegant way. Nevertheless, we included some 170 new and vintage examples in this OM to support you learning your new tool.

## Problem Solving, Part 2: Elementary Stack Mechanics

Most commands of your WP43 are mathematical operations or functions taking and returning real numbers (shortly reals) - like 3.14159265359
${ }^{20}$ A quick and simple unit analysis (a.k.a. dimension check) is strongly recommended before starting a calculation of a formula you may have derived yourself or during exploratory calculations (see below).

Working with different sets of units carries significant risks - see this expensive experiment: https://www.latimes.com/archives/la-xpm-1999-oct-01-mn-17288-story.html.
The big advantage of $S I$ is that it is the largest coherent set of units available on this planet (look up ReMS 2 if necessary).
${ }^{21}$ Your WP43 features many data types - we will introduce them to you in OMS 2.
${ }^{22}$ Almost all user guides, application handbooks, and manuals printed for HP calculators in two decades, beginning with the HP-9100A of 1968 (HP's $1^{\text {st }}$ desktop calculator, without any IC but with a cathode ray tube built in for display) are still available at low cost (together with almost complete information about all those HP calculators built until 1990) in a package of over 14 gigabytes on media distributed by David Hicks' online Museum of HP Calculators (see http://www.hpmuseum.org/cd/cddesc.htm).
Compare with Eric Rechlin's free online offer (see https://literature.hpcalc.org) covering pocket calculators between 1972 and 2021 as mentioned on p. 22.
or 0.125 or $-5.67 \times 10^{-8}$. Note that integers like 3 or 12345678 or -12321 , as well as fractions like $4 / 5$ or $137 / 7$ are mere subsets of reals.

Depending on the command you choose, it may operate on one, two, or three numbers at once to generate a result. In spite of the hundreds of functions available, you will find your WP43 functions simple to operate by using a single, all-encompassing rule:

When you press a function key, your WP43 will execute the operation assigned to it immediately. ${ }^{23}$

One-number (monadic) functions: Many functions provided operate on one number only. Nine of them are on the keyboard: the reciprocal $1 / 1 / x$, the logarithms In and (19), the exponentials $\boldsymbol{e}^{x}$ and $10^{x}$, the square root $\sqrt{\boldsymbol{x}}$, furthermore $|x|$ (returning a positive number always), $+\infty$ (multiplying a closed number times -1 ), and the factorial

Examples to be executed consecutively:


Each monadic function replaces $x$ by the respective function result $f(x)$ (= $x$ ! in the last example). Everything else will stay as it was. - Please check the $I O I$ for the many monadic functions provided.

[^11]Two-number (dyadic) functions: Some of the most popular mathematical functions operate on two numbers and return one. Think of the four basic arithmetic operators + and,$- \times$ and $/$.

## Example: <br> Assume owning an account of 1234.00 US\$ and withdrawing 56.70 US\$ from it. What will remain?

Solving such a problem easily may work like this:
On a piece of paper: $\quad \rightarrow$

Write down the $1^{\text {st }}$
number: 1234.00
Start a new line and write down
the $2^{\text {nd }}$ number:
56.70

Draw a line below.
Subtract: 1177.30
$\rightarrow$
On your WP43:
Key in the $1^{\text {st }}$
number: 12 3 1234.
Close $1^{\text {st }}$ input: ENTER $\uparrow$
Key in the $2^{\text {nd }}$
number:
5 $6 \longdiv { 7 }$
56.7

This is the essence of $R P N$ :
Provide the necessary operands, then execute the requested operation by pressing the corresponding function key.

HP itself explained RPN using the following compact picture. And a major advantage of RPN compared to all other calculator operating systems we know is it sticks to this basic rule - always. ${ }^{24}$

[^12]As paper holds your operands while you are calculating manually, space holding your operands is required on your WP43 as well.

The stack does this job, and this is its raison d'être: It is a pile of connected registers, work space for your calculations. Bottom up, these stack registers are traditionally called $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, and $\mathbf{T}$, optionally followed by $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and $\mathbf{D}$ on your WP43. ${ }^{25}$ Your input is entered in $\mathbf{X}$ always, and at least $\boldsymbol{x}$ is always displayed in RUM - $\boldsymbol{y}, \boldsymbol{z}$, and $\boldsymbol{t}$ may be, so you can see the actual contents of the bottom four stack registers at a glance.


Stack register name

| D | $d$ |
| :--- | :--- |
| $\mathbf{C}$ | $c$ |
| B | $b$ |
| A | $a$ |
| T | $t$ |
| Z | $z$ |
| Y | $y$ |
| X | $x$ |

ENTERT separates two input numbers by closing $x$ and copying it into Y. The contents of the upper stack registers are lifted out of the way before. I.e. for a 4-level stack, $\boldsymbol{z}$ is copied into $\mathbf{T}$, then $\boldsymbol{y}$ into $\mathbf{Z}$, then $\boldsymbol{x}$ into $\mathbf{Y}$. Thereafter, $\mathbf{X}$ can take new input without loss of information.

This is the classical functionality of ENTERT from the HP-35 of 1972 until the HP-42S ceased production in 1995. EENTERT affects all stack

[^13]Press
registers, and the previous content of the top register gets lost. ${ }^{26}$ It is often said ENTERT 'pushes $\boldsymbol{x}$ on the stack' (although it pushes $\boldsymbol{x}$ under the stack in all pictures we know).

Let's look at our account example again, putting it in a stack diagram: ${ }^{27}$

| Y |  | $\longrightarrow$ | 1234 |  | 1234 | $\xrightarrow{-\Delta} \stackrel{ }{1177.3}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| X | 1234 |  | 1234 | 56.7 |  |  |  |
| Input | 1234 | ENTER |  | 56.7 |  | $\square$ |  |

After entering the $2^{\text {nd }}$ number ( $56.7=\boldsymbol{x}$ ), pressing - subtracts this from the $1^{\text {st }}$ number $(\mathbf{1 2 3 4}=\boldsymbol{y})$ and puts $f(x, y)=y-x=1177.3$ into $X$. This simple method applies to almost all dyadic functions on your WP43:

> Put the operands on the stack, then call the operation $f(x, \ldots)$, and its result will be displayed.

## Example:

$$
\frac{(12.3-45.6)(78.9+1.2)}{(3.4-5.6)^{7}}=?
$$

This is as a combination of 6 dyadic functions: two subtractions, an addition, a multiplication, an exponentiation, and a division. Starting top left in the formula, this is what will happen on the stack of your WP43while solving it:

[^14]27 At beginning, some arbitrary data may be present in the upper stack registers $\mathbf{Y}, \mathbf{Z}$, etc., remaining from earlier work. These data are irrelevant for the current calculation, so we left them aside here (actually, they are pushed out of the way - compare the picture above). In further stack diagrams we will omit entirely all stack contents irrelevant for the calculation under investigation, for sake of clarity. And we will generally use plain bold black text denoting numeric input from here on for the same reasons.
${ }^{28}$ This eliminates the need for an $\Xi$ on the keyboard, also beyond monadic operations.


You will have recognized that this $1^{\text {st }}$ parenthesis in the numerator was solved exactly as demonstrated in our little account example above. Proceed to the $2^{\text {nd }}$ parenthesis now:


It is solved like the $1^{\text {st }}$. But in the $1^{\text {st }}$ step of this sequence, the prior result (of $1^{\text {st }}$ parenthesis) is lifted automatically to $\mathbf{Y}$ to avoid overwriting it with the next number keyed in. This move is called automatic stack lift (ASL).

Actually, such an ASL (originally called automatic ENTER) works as if ENTERT was pressed before 7 here. ASL is standard on RPN calculators since 1972, reducing the keystrokes necessary, and will not be indicated from here on. ${ }^{29}$ Remember you need ENTERT just for separating two consecutive numbers in input - cf. p. 36. In consequence, we need neither (1) nor (1) and can solve problems with a minimum of keystrokes.

After having solved the $2^{\text {nd }}$ parenthesis by pressing $\boldsymbol{\oplus}$, the results of both upper parentheses were on the stack in $\mathbf{X}$ and $\mathbf{Y}$ - well prepared for the multiplication to complete the numerator. Thus, we just did it.

29 Due to ASL, there is also no need for clearing your WP43 before starting a new calculation - old data are just lifted out of the way when new input is entered. An ASL affects all stack registers (as a standard ENTERT does) and the previous content of the top register gets lost again. Of the hundreds of commands provided on your WP43, there are only ten disabling ASL: ENTER, CLX, $\Sigma+, \Sigma-$, and six matrix editing commands. Some reasoning for the first four:

- After ENTERT, you generally want to key in the next input number.
- CLX (called by CLX or $\boldsymbol{\Psi}$, the latter only for closed $\boldsymbol{x}$ ) is for clearing $\mathbf{X}$ to make room for a corrected value instead of the one deleted (and we do not want a useless extra zero on the stack).
- $\Sigma+$ and $\Sigma$ - are designed for entry of statistical data. Please see the chapters about statistical functions in OMS 2.

Now we start calculating the denominator - once again the intermediate result is lifted automatically in the $1^{\text {st }}$ step:

|  | -2 667.33 | -2 667.33 |  | -2 667.33 |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| -2 667.33 | 3.4 | 3.4 | -2 667.33 | -2.2 | -2 667.33 |  |
| 3.4 |  | 5.6 | -2.2 | 7. | -249.43... | 10.693... |
| 3.4 | ENTERT | 5.6 | $\square 7$ | 7 | EXP $\mathrm{y}^{\mathrm{x}}$ |  |

Note the ASL when entering 7 affects two intermediate results now. Thus, everything is well prepared for the exponentiation in the penultimate step and the final division of the numerator $\boldsymbol{y}$ by the denominator $\boldsymbol{x}$. Voilà!

Following this example, you have observed the five most popular dyadic functions in action: $\oplus, \boxed{\square}, \boldsymbol{x}, \square$, and ${\sqrt{y^{x}} \text {. Your WP43 provides many }}^{\square}$ more dyadic functions (see the $I O I$ ).

As you have watched several times now, the contents of the stack registers drop whenever a dyadic function is executed. Like ASL

| Press | Contents | Location |
| :---: | :---: | :---: |
| $+$ |  |  | above, also this stack drop affects all stack registers.

In the example pictured here, $x$ and $y$ are combined resulting in $f(x, y)=\boldsymbol{y}+\boldsymbol{x}$ put into $\mathbf{X}$; then $z$ drops to $\mathbf{Y}$, and $\boldsymbol{t}$ to $\mathbf{Z}$. In a 4-level stack, nothing is available above $\mathbf{T}$ for dropping, so the top register content is repeated (see also p. 43).

There are also some three-number (triadic) functions featured (e.g. $\rightarrow$ DATE and \%MRR). Executing such a function replaces $x$ by $f(x, y, z)$; then $\boldsymbol{t}$ drops into $\mathbf{Y}$ and so on, and the content of the top stack register is repeated twice (see p. 43 for an example). All triadic functions provided by your WP43 are found in menus.

And some functions operate on one number but return two (like DECOMP) or three (like DATE $\rightarrow$ ). Other operations do not consume any stack input
at all but may just return one, two, or three objects (like RCL, SUM, or L.R.). Then these returned objects will be pushed on the stack, taking one register each (see p. 43).

Further ways of function parameter handling will be shown in OMS 2.

## Looking Closer at the Automatic Stack

For understanding the genius of $R P N$, let's look a bit closer at the functions operating on the stack. In addition to the monadic, dyadic, and triadic functions explained above, there are some dedicated stack and register commands:

ENTER $\uparrow$, $x^{2} \geqslant \mathrm{y}, \mathrm{R} \downarrow, \mathrm{R} \uparrow$, STO, (RCL, $x^{2}$, and VIEW are known from previous calculators, some of them for decades. You find them - together with the new labels (RBR), (FILL), DROP $\downarrow$, and STK - all within this small area of the keyboard.

## ASNSAVE RBRVIEW R^CPX <br>  <br> FILL DROP $\downarrow$ ENTER $\uparrow$ <br> $x \geq y_{k}$

Your WP43 features even more stack and register commands: CLREGS, CLSTK, STOIJ and RCLIJ, STOS and RCLS, STOCFG and RCLCFG, DROPy, y*, z STK.

And your WP43 allows for expanding the traditional 4-level stack of 1972 to 8 levels (like WP 34S and WP 31S do since 2011).

Just set the respective system flag: SF SYS.FL (SSIZE8).
If you ever want to return to the old 4-level stack, enter CF SYS.FL (SSIZE8) to clear this flag.

In consequence, the fate of certain stack contents will depend on the particular operation executed as well as on the stack size at execution time (current stack size is indicated in the status bar at the top of your WP43 screen).

Operations on the 4-level stack work as known from vintage HP RPN calculators since the HP-45. On the optional 8-level stack of your WP43, everything works in full analogy - just more registers are available for intermediate results. Advantages will become evident in examples following, no extra efforts are requisite.

Find on the next two pages what ENTERt, (FILL), DROP $\downarrow$, DROPy, $x \geqslant y$, $R \downarrow,(R \uparrow$, and further representative functions do in detail on stacks of either size. Then you will also know why the labels ENTERT and $\mathbb{R} \uparrow$ show arrows pointing up while $\mathbb{R} \downarrow$ and $\triangle D R O P \downarrow$ point down.

Manually clearing the entire stack can be done by 0 (FILL most easily, regardless of stack size. Nevertheless, a dedicated command CLSTK is provided in CLR for backward compatibility, program space economy, and speed of execution (see p. 55).
$x \geqslant y$ swaps the contents of stack registers $\mathbf{X}$ and $\mathbf{Y}$. Depending on the problems you solve and the way you proceed, you may find that $\boldsymbol{x}$ and $\boldsymbol{y}$ should be swapped sometimes before executing e.g. $-\boxed{\square}, \square$, or $y^{x}$.

Note the previous content of the top stack register is lost when ENTERT or RCL is executed. Functions like ss or DATE $\rightarrow$ will even cost the contents of two stack registers. Set SSIZE8 for mitigating the effects of such losses.

RCL (L) as shown on p. 43 represents the vintage command LASTx (cf. the picture on p .39 ). You will learn about its benefits on p .52.


|  | T | $t=4$. | 3. | 1.1 | 4. | 4. | 4. | 1.1 | 3. |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Z | $z=3$. | 2. | 1.1 | 4. | 4. | 3. | 4. | 2. |
|  | Y | $y=2$. | 1.1 | 1.1 | 3. | 3. | 1.1 | 3. | 1.1 |
|  | X | $x=1.1$ | 1.1 | 1.1 | 2. | 1.1 | 2. | 2. | 4. |

The stack rotation commands $\mathbb{R \downarrow}$ and $\mathbb{R} \uparrow$ might come handy for reviewing stack registers else unseen (the little picture shows a traditional diagram for $\mathbf{R} \downarrow$ in a 4-level stack). ${ }^{30}$ Alternatively, use the
 register browser (RBR displaying ten registers at once without disturbing the stack at all (see pp. 277ff).

[^15]|  | 0 | Stack contents after executing ... |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\underset{\underset{\sim}{0}}{\underset{\sim}{\theta}}$ | $\underbrace{\infty}_{\sim}$ | 춫 <br> $\stackrel{\omega}{\omega}$ | $\oplus_{\underset{\sim}{\infty}}$ |  |  |
| D | $d=8$. | 7. | 6. | 8. | 8. | 8. | 6. |
| C | $c=7$. | 6. | 5. | 7. | 8. | 8. | 5. |
| B | $b=6$. | 5. | 4. | 6. | 7. | 8. | 4. |
| A | $\boldsymbol{a}=5$. | 4. | 3. | 5. | 6. | 7. | 3. |
| T | $t=4$. | 3. | 2. | 4. | 5. | 6. | 2. |
| Z | $z=3$. | 2. | 1.1 | 3. | 4. | 5. | 11 |
| Y | $y=2$. | 1.1 | $S_{y}$ | 2. | 3. | 4. | 12 |
| X | $x=1.1$ | last $x$ | $S_{x}$ | 1.21 | 3.1 | 0003-02-01 | 30 |


| T | $t=4$. | 3. | 2. | 4. | 4. | 4. | 2. |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Z | $z=3$. | 2. | 1.1 | 3. | 4. | 4. | 11 |
| Y | $y=2$. | 1.1 | $s_{y}$ | 2. | 3. | 4. | 12 |
| X | $x=1.1$ | last $x$ | $\boldsymbol{S}_{\boldsymbol{x}}$ | 1.21 | 3.1 | 0003-02-01 | 30 |

See the $I O I$ for further information about the commands mentioned above.
${ }^{31}$ RCL (L) represents an arbitrary function pushing one object on the stack.
${ }^{32}$ (s) represents an arbitrary function pushing two objects on the stack.
${ }^{33}$ x $\mathbf{x}^{2}$ represents an arbitrary monadic function. .
${ }^{34} \mp$ represents an arbitrary dyadic function.
${ }^{35}$ Assume $\rightarrow$ DATE being called in startup default date mode (i.e. YMD). $\rightarrow$ DATE represents an arbitrary triadic function here.
${ }^{36}$ Assume 11.123 or 11-12-30 in $\mathbf{X}$ initially and startup default mode here, cf. OMS 2.

## Problem Solving, Part 3: Advanced Stack Mechanics

Using the stack, RPN eliminates the need for (1) (1) keys as well as for $\Xi$. See the following example: it shows a more elaborate formula than above. Find below a way for solving it, step by step, and the corresponding stack diagrams.
$2+\sqrt{\frac{1+\left|\left(\frac{30}{7}-7.6 \times 0.8\right)^{4}-\left(\sqrt{5.1}-\frac{6}{5}\right)^{2}\right|^{0.3}}{\left\{\sin \left[\pi\left(\frac{7}{4}-\frac{5}{6}\right)\right]+1.7(6.5+5.9)^{3 / 7}\right\}^{2}-3.5}}=?$
Enter (MODE RAD and start calculating at the red 7:

| Z |  |  |  |  |  | 1.75 | 1.75 |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Y |  | 7 | 7 |  | 1.75 | 5 | 5 | 1.75 |  |
| X 7 |  | 74. |  | 1.755 |  | 56 |  | 0.83... | 0.91... |
|  | 7 | 14 |  | 5 |  | NTT 6 |  | 1 | $\square$ |


|  |  |  |  |  |  |  | 0.25... | 0.25... |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | 0.25... | 0.25... |  | 0.25... | 12.4 | 12.4 |
| 0.91... |  | 0.25... | 6.5 | 6.5 | 0.25... | 12.4 | 3 | 3 |
| 3.14... | 2.87... | $0.25 . . .6 .5$ | 6.55 .9 |  | 12.43 |  | 37 |  |
| (17) | 区 |  | ENTT | 5.9 | $\pm$ | 3 | ENTT |  |


| 0.25 | 0.25 | 0.25. | 0.25. | 0.25 | 0.25 | 0.25. | 0.2 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 0.25... | 0.25 | 0.25... | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| 12.4 | 0.25... | 2.94... | 0.25... | 0.25 | 0.25. | 27.6... | 0.25. |
| 0.42... | 2.94...1.7 |  | 5.00... | 5.25... | 27.6... 3.5 |  | 24.1... |
|  | $\mathrm{y}^{\mathrm{x}}$ |  |  |  |  |  | $\square$ |

This was the solution of the denominator. Let's continue with calculating the numerator now, basically following the same procedure, i.e. calculating from inside out (as you would do with pencil and paper):

| 0. | 0.25. | 0.25 | 0.25 | 0.25 | 24.1... | 24.1... |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 0.25. | 24.1... | 24.1... | 0.25 | 24.1... | 6.08 | 6.08 | 24.1... | 24.1 | 24.1... |
| 24.1... | 7.6 | 7.6 | 24.1... | 6.08 | 30 | 30 | 6.08 | 24.1... | 1.79. |
| 7.6- | 7.6 | .8 | 6.08 | 30 | 30 | 7. | 4.28... | 1.79... | 4. |
| 7.6 | NTt | . 8 | x | 30 | ENTt |  |  |  |  |


| 24.1 | 24.1 . | 24.1... | 24.1... | 24.1 | 24.1... | 24.1... | 24.1 | 24.1 |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 24.1. | 24.1... | 10.3... | 10.3... | 24.1.. | 10.3.. | 10.3.. | 24.1... | 24.1... | 24.1 |
| 24.1... | 10.3... | 6 | 6 | 10.3... | 1.2 | 1.2 | 10.3... | 10.3.. | 24.1... |
| 10.3... | 6 | 6 | 5 | 1.2 | 5.1- | 2.25... | -1.05... | 1.12... | 9.24. |
| $\mathrm{y}^{\text {x }}$ | 6 | Tt | 5 | 1 | 5.1 |  | $\square$ |  | $\square$ |


|  | 24 |  | 24 |  |  | 24. |  | 24.1 |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 24.1. | 24.1... | 24.1 | 24.1... | 24.1. | 24.1.. | 24 | 24.1.. | 24.1 |  |
| 24.1... | 9.24... | 24.1... | 1.94... | 24.1... | 2.94... | 24.1 | 24.1. | 0.34... | 24.1 |
| 9.24... | .3- | 1.94... | 1 - | 2.94... | 24.1... | 0.12... | 0.34... | 2 | 2.349... |
| \|x| 37 |  | $\mathrm{y}^{\text {x }}$ | 1 | $\pm{ }^{38}$ | $x^{2} \times 2$ |  | (1) | 2 | $\pm$ |

Even solving this formula requires only four stack registers. There are no pending operations - each operation is executed individually, one at a time, allowing perfect control of each and every intermediate result. ${ }^{39}$

Note this is another characteristic advantage of RPN. In many real-life applications, intermediate results carry their own value, so further calculations may depend on the numbers you see there - this is called 'exploratory math' and may well occur more frequently in your professional work than evaluating textbook formulas.
${ }^{37}$ You will hardly execute this step manually since you will see immediately that $\boldsymbol{x}$ is positive. In an automatic evaluation of such a formula, however, this step is important.
${ }^{38}$ Note $\mathbf{X}$ contains the result of the numerator here. And the result for the denominator silently traveled into $\mathbf{Y}$ during all the calculations printed above. In the next step, we swap $\boldsymbol{x}$ and $\boldsymbol{y}$ to arrange the operands for dividing the numerator by the denominator. We also filled $\mathbf{Z}$ and $\mathbf{T}$ while calculating (and recorded $\boldsymbol{z}$ and $\boldsymbol{t}$ for sake of completeness then) but present $\boldsymbol{z}$ and $\boldsymbol{t}$ are irrelevant for our result.
${ }^{39}$ Thus, operator precedence is your job. Look up App. 2 for confirmation or reminder.

The argument in school calculators is "Algebraic lets you enter the equation just as you see it on paper!" Well, for one thing, a lot of programming work falls outside of actual equations. For another, even in the field of calculators, in real life we don't usually have an equation in front of us. I think through it, "Let's see - I need A, and then square that; now take B, multiply it by C and add that result to the earlier one. Now I need D raised to the power of E, and divide the earlier result by that. Done. Oops, no, I still need to take the log of that..." That's the way much of real engineering is.

It's only in school that you get canned problems and have the equation in the book and you're supposed to just drop the numbers in the chute and turn the crank - then we wonder why we get graduates who passed all their classes and yet show a disconnect between that knowledge and any understanding of what the problem is in their circuit on the workbench! I've hired a lot of electronics technicians and a few engineers, and I gave all the applicants a circuit-analysis test with a dozen simple problems. Not one of them ever got it all correct. It was kind of frustrating. (Garth Wilson of WilsonMinesCo.com)

Algebraic tells you what you get. RPN tells you how you get there. (Wilson)

Experienced RPN calculator users have determined that by starting every problem at its innermost number or parenthesis and working outwards, you maximize the efficiency and power of your calculator.

> Example (continued):
> If, instead, you had tried solving the formula on p. 44 starting with the numerator of the root straight ahead, stubbornly calculating from left to right, you had needed more keystrokes and six stack registers for the entire solution instead of only four. This way is viable - it is not very smart though.

There may be, however, some problems where four stack registers will simply not suffice regardless of the way you tackle with them:

## Example:

$$
\frac{(1+2)(9+8)+(3+4)(11+6)}{(5-7)(10+12)-(13+14)(15+16)}=?
$$

This highly symmetric formula lacks an unambiguous 'inside', so it does not matter where we start solving it. Let's begin with the numerator, assuming startup default configuration of your WP43:


| $\mathbf{T}$ |  |  |  |  | 51 | 51 | 51 | 51 | 51 | 51 |
| ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: |
| $\mathbf{Z}$ | 51 | 51 |  | 51 | 7 | 7 | 51 | 51 | 51 | 51 |
| $\mathbf{Y}$ | 3 | 3 | 51 | 7 | 11 | 11 | 7 | 51 | 51 | 170 |
| $\mathbf{X}$ | 3 | $4-$ | 7 | $\mathbf{7}$ | $11-$ | 11 | $6-$ | 17 | 119 | 170 |


| T | 51 |  |  | 51 | 170 | 170 |  | 170 | 170 | 170 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Z | 170 | 170 | 51 | 170 | -2 | -2 | 170 | 170 | 170 | -44 |
| Y | 5 | 5 | 170 | -2 | 10 | 10 | -2 | 170 | -44 | 13 |
| X | 5 | 7 | -2 | 10. |  | 2 | 22 | -44 | 13. | 13 |
|  | ENTT |  | - | 10 | NTT | 12 |  |  |  | NTT |


| T | 170 | 176 | 170 | -44 | -44 | -44 | -44 | -44 |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Z | -44 | 170 | -44 | 27 | 27 | -44 | -44 | -44 | -44 |
| Y | 13 | -44 | 27 | 15 | 15 | 27 | -44 | -44 | -44 |
| X | 14 | 27 | 15 | 15 | 16 | 31 | 837 | -881 | 0.04994. |
|  |  |  | 15 | NTt | 16 |  | x |  | 1 |

Pressing [ENTERT in step 4 of last diagram, a stack overflow occurs: the previous top register content (170) is pushed up out of $\mathbf{T}$ and lost, and in consequence a wrong $t$ is repeated leading to a wrong final result. Note that probability for a stack overflow is low but not negligible with the traditional 4-level stack. ${ }^{40}$

For comparison, after setting SSIzE8, the last diagram will look like this instead:

[^16]| D |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| C |  |  |  |  |  |  |  |  |  |
| B |  |  |  |  |  |  |  |  |  |
| A |  |  |  | 170 | 170 |  |  |  |  |
| T | 170 |  | 170 | -44 | -44 | 170 |  |  |  |
| Z | -44 | 170 | -44 | 27 | 27 | -44 | 170 |  |  |
| \| $\mathbf{Y}$ | 13 | -44 | 27 | 15 | 15 | 27 | -44 | 170 |  |
| $\mathbf{X}$ | 14 | 27 | 15 | 15 | 16 | 31 | 837 | -881 | -0.192 96... |
|  |  |  | 15 | NTT | 16 |  |  | $\square$ | (1) |

## We will close with another real-life example:

For many years, solving the following formula for the Mach number of an airplane as a function of its calibrated airspeed (CAS) in knots (here: 350 ) and pressure altitude ( $P A$ ) in feet (here: 25 500) was used for demonstrating the simplicity and coherence of RPN:

$$
\sqrt{5\left(\left[\left\{\left(1+0.2\left[\frac{C A S}{661.5}\right]^{2}\right)^{3.5}-1\right\}\left\{1-6.875 \times 10^{-6} \times P A\right\}^{-5.2656}+1\right]^{0.286}-1\right)}
$$

Solve it like this: ${ }^{41}$

```
350 ENTERT 661.5 (1) EXP x . 2 x 1 + 3.5 y x 1-
6.875 E +// 6 ENTERT 25500 x +/ 1 + 5.2656 +/ yx x
1+ . 286 y x
1-5 \sqrt{ \}{x}\mathrm{ resulting in 0.84, i.e. 84% of the speed of sound.}
You needed only three stack registers for solving this.
```

[^17] The ancient British Imperial seafarer's knot survived in aviation business and navigation:
$$
1 \mathrm{knot}=1 \text { nautical mile } / \text { hour }=463 / 900 \mathrm{~m} / \mathrm{s} \approx 0.5144 \mathrm{~m} / \mathrm{s} \approx 1.85 \mathrm{~km} / \mathrm{h}
$$

The foot is another unit from that heap of pre-modern units made obsolete by SI for two centuries - it survived in aviation as well (see Unit Conversions in OMS 5).
$T N$ for Europeans: CAS does not mean $C \cdot A \cdot S$, nor does $P A$ mean $P \cdot A$ in that formula.

As you have seen, the way to solve a problem using $R P N$ stays the same regardless of problem size and complexity. You are always in control.
While a stack overflow may (seldom but silently) happen on a 4-level stack, you will be clearly on the safe side with an 8-level stack as provided on your WP43, even dealing with the most advanced mathematical expressions you will meet in your professional life as a scientist or engineer. Promised. The 8-level stack warrants absolutely worry-free calculations. ${ }^{42}$
Let's quote a part of the HP-25 OH once more, just replacing all strings 'HP-25' by 'WP43':

Now that you've learned how to use the calculator, you can begin to fully appreciate the benefits of the Hewlett-Packard logic system. With this system, you enter numbers using a parenthesis-free, unambiguous method called $R P N$ (...).
It is this unique system that gives you all these calculating advantages whether you're writing keystrokes for a WP43 program or using the WP43 under manual control:

- You never have to work with more than one function at a time. The WP43 cuts problems down to size instead of making them more complex.
- Pressing a function key immediately executes the function. You work naturally through complicated problems, with fewer keystrokes and less time spent.
- Intermediate results appear as they are calculated. There are no "hidden" calculations, and you can check each step as you go.
- Intermediate results are automatically handled. You don't have to write down long intermediate answers when you work a problem.
- Intermediate answers are automatically inserted into the problem on a last-in, first-out basis. You don't have to remember where they are and then summon them.
- You can calculate in the same order you do with pencil and paper. You don't have to think the problem through ahead of time.

[^18]$R P N$ takes a few minutes to learn. But you'll be amply rewarded by the ease with which the WP43 solves the longest, most complex equations. With RPN, the investment of a few moments of learning yields a lifetime of mathematical bliss.

And calculations with other data types (to be introduced in OMS 2) follow the same simple rules. Thus, at the bottom line, we recommend:

## Set SSIZE8 and let your WP43 care for the arithmetic while you care for the mathematics! ${ }^{43}$

[^19]For the same reason, we omit heading indicators $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$, and T in display. You chose this calculator for yourself, so you are certainly able to remember these four letters naming the bottom four stack registers.

On the other hand, if you feel distracted or even annoyed by a garrulous screen displaying more data than necessary, feel free to use the command DSTACK to reduce the number of stack registers displayed to 3,2 , or even just 1 , letting your brains compete with those of your fellow RPN users since 1972.

Free space will flow in top down $-\boldsymbol{x}$ will always be shown directly above the menu section. See here an exemplary LCD for DSTACK 2:

Multi-line output will be shown entirely always, regardless of the DSTACK setting.
We count on your abilities and are very confident you will succeed.

```
2020-11-19 11:12 \(\mathrm{ClL}_{4}{ }^{\circ} /\) max \(\mathrm{BH}_{2} \mathrm{U}_{\mathrm{E}}\) 雨
15628.352
4294967296
```

| GAP |  | RANGE RANGE 3 |  | DSTACK |
| :---: | :---: | :---: | :---: | :---: | :---: |
| CHINA | EUROPE | INDIA JAPAN | UK | USA |

## Special RPN Tricks, \#1: Top Stack Register Repetition

Whenever a dyadic or triadic function is executed, the stack will drop and the content of its top register will be repeated - cf. pp. 39 and 43 , for instance. You may employ this top stack register repetition for some nice tricks. We will show you two examples on a 4-level stack.
See the following compound interest calculation: ${ }^{44}$

## Example:

Assume your bank pays you $3.25 \%$ p.a. on an amount of 15000 US\$; 45 what would be your status after $2,3,5$, and 8 years?

## Solution:

Here, you are interested in currency values only, so set the display format to DISP FIX (2). This causes the output being rounded to cents (internally, numbers are kept and calculated with far higher precision):

| T |  | 1.03 | 1.03 | 1.03 | 1.03 | 1.03 | 1.03 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Z |  | 1.03 | 1.03 | 1.03 | 1.03 | 1.03 | 1.03 |
| Y |  | 1.03 | 1.03 | 1.03 | 1.03 | 1.03 | 1.03 |
| X | 1.0325 | 1.03 | 15000 | 15990.84 | 16510.55 | 17601.17 | 19373.66 |
| 1.0325 |  | FILL) | 15000 | $\frac{\boldsymbol{x} \boldsymbol{x}}{2 \text { years }}$ | 区 <br> 3 years | $\underbrace{\mathbf{x}}_{5 \text { years }}$ | $\begin{gathered} \boldsymbol{x} \boldsymbol{x} \boldsymbol{x} \\ 8 \text { years } \end{gathered}$ |
|  |  |  |  |  |  |  |  |

## FILL DROP $\downarrow$

 ENTER $\uparrow$Each multiplication consumes $\boldsymbol{x}$ and $\boldsymbol{y}$ for the product $\boldsymbol{x} \times \boldsymbol{y}$ put in $\mathbf{X}$, followed by $\boldsymbol{z}$ dropping into $\mathbf{Y}$, and $\boldsymbol{t}$ copied into $\mathbf{Z}$. Due to top stack register repetition the interest rate is kept as a constant on the stack, so the computation of the accumulated capital value becomes a simple series of $\boldsymbol{X}$ strokes. ${ }^{46}$

[^20]Another application benefitting from top stack register repetition is the Horner scheme for evaluating polynomials. It tells:

$$
\begin{aligned}
& p(x)=a_{0}+a_{1} x+a_{2} x^{2}+\cdots+a_{n-1} x^{n-1}+a_{n} x^{n} \\
& \quad=\left(\left(\cdots\left(a_{n} x+a_{n-1}\right) x+\cdots+a_{2}\right) x+a_{1}\right) x+a_{0}
\end{aligned}
$$

## Example:

Solve $7+6.4 x-2.1 x^{2}+5.2 x^{3}-3 x^{4}$ for $x=0.908$.

## Solution:

This problem can be rewritten to

$$
\{[(-3 x+5.2) x-2.1] x+6.4\} x+7
$$

and is easily solved this way (with display set to DISP FIX (1) ):

|  | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 |
|  | 0.9 | 0.9 | 0.9 | -2.7 | 0.9 | 0.9 | 2.2 | 0.9 | 0.1 | 0.9 | 0.9 |
| .908 | 0.9 | -3- | -2.7 | 5.2 | 2.5 | 2.2 | 2.1- | 0.1 | 6.4 | 5.9 | 12.9 |
| . 908 |  |  |  |  |  |  |  |  |  |  |  |

Note how the values float automatically down the stack to be used in the multiplications. ${ }^{47}$

Both examples will work with the 8-level stack as well - just with $\boldsymbol{d}$ repeated instead of $\boldsymbol{t}$. (FILL) loads the entire stack with $\boldsymbol{x}$, far easier than hitting ENTERT repeatedly.

## Special RPN Tricks, \#2: LASTx for Reusing Numbers

For most commands, your WP43 copies $\boldsymbol{x}$ into the special register $\mathbf{L}$ (for 'Last $\boldsymbol{x}$ ') automatically just before the command is executed - as previous RPN calculators did (cf. p. 39). ${ }^{48}$ What is your benefit?

[^21]
## Example (from the HP-15C OH):

Two close stellar neighbors of Earth are Rigel Centaurus ${ }^{49}$ (4.3 light-years away) and Sirius (8.7 light-years away). Use the speed of light, c ( $2.99792 \times 10^{8} \mathrm{~m} / \mathrm{s}$, or $9.46054 \times 10^{15} \mathrm{~m} /$ year), to figure the distances to these stars in meters.

Solution (with DISP SCI 01 set):


ENTERT
9.46054 E 15

区
8.7

RCL (L)

## 区


4.3

50

$$
4.1 \times 10^{16}
$$

8.7-
$9.5 \times 10^{15}$
$8.2 \times 10^{16}$
$\boxed{R C L}(L)$ is reached by pressing $(\mathbb{R C L}$, then $+/$ - note L printed bottom right of $+/ .^{51}$

Result: Rigel Centaurus has a distance of $4.1 \times 10^{16} \mathrm{~m}$ to our planet, Sirius $8.2 \times 10^{16} \mathrm{~m}$.
(RCL (L) will often allow you keying in lengthy numbers only once (improving input consistency). It may also allow for reusing intermediate results without the need for storing them explicitly.

## Error Recovery: ©, EXIT, and

Nobody is perfect - errors will happen although you are equipped with such a powerful tool. Stay cool - your WP43 allows undoing the last command executed, restoring the calculator state as it was before that error occurred.

[^22]1. If you receive an error message in response to a function call, note it, then press $\leftrightarrows$ or EXIT; this will erase that message and return to the state before that error happened (see pp. 71 and 324f). Then do it right!
2. If you have erroneously executed a wrong function, just press $\curvearrowleft \sim$ to UNDO it immediately. ( $\curvearrowleft$ ) recalls the calculator state as it was before that wrong operation was executed. $\curvearrowleft$ operates on the entire stack (be it 4 or 8 registers deep), statistic storage, and system flags. Since your WP43 features RPN, this is all you need for resuming your work as if said wrong function was not called and executed at all. ${ }^{52}$

## Example:

Assume you pressed - while you were watching an attractive fellow student or collaborator - $\boldsymbol{x}$ inadvertently instead of (1) in the fourth last step solving the lengthy formula on p. 44. Shhh... !! Murphy's Law! Luckily, however, there is absolutely no need to start calculating that formula all over again that user error is easily undone as follows:

| T | ... | 24.1... | 24.1... | 24.1... | 24.1... | 24.1... | 24.1... | 24.1... |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Z | ... | 24.1... | 24.1... | 24.1... | 24.1... | 24.1... | 24.1... | 24.1... |
| Y | ... | 2.94... | 24.1... | 2.94... | 24.1... | 24.1... | 0.34... | 24.1... |
| X | ... | 24.1... | 70.9... | 24.1... | 0.12... | 0.34... | 2 | 2.349.. |
|  | $x \geqslant y$ x |  |  | ص) (1) |  | $\sqrt{x}$ | 2 | $\pm$ |

Fine so far Oops! Undo Resume
Correct result

So don't worry - be happy!

[^23]
## Clearing and Resetting Your WP43

Besides $\uplus$ and $\curvearrowleft \sim$, there are various other ways to remove obsolete, erroneous, or unwanted information from your WP43. The menu CLR comprises these commands:


CLX Clears the stack register $\mathbf{X}$ (i.e. sets it to zero) ${ }^{53}$

CLI Clears all statistical data and returns freed space

CF Clears the flag specified
CLP Clears the program specified
CLMENU Clears the programmable menu

CLALL Clears all programs and data (variables, numbered user flags, and all registers including the stack) ${ }^{55}$

CLSTK Clears all stack registers allocated (not L)

## CLREGS Clears all global and local GP registers ${ }^{54}$

CLFALL Clears all numbered flags
CLPALL Clears all programs
CLCVAR Clears all variables of the current program

RESET Resets your WP43 to startup default (just the contents of $F M$ will stay untouched) ${ }^{56}$

DELITM Deletes the user-defined item selected

For your reference, startup default settings of your WP43 are:
2COMPL, ALL 00, DEG, DENMAX 9999, DSTACK 4, GAP 3, HIDE 0, J/G 1752.0914, LinF, LocR 0, RM 0, TDISP 0, WSIZE 64, and Y.MD. RANGE is set to 6145 .

The system flags AUTOFF, DECIM., DENANY, ENDPMT, MULTx, PROPFR, TDM24, and aCAP are set, all others are clear.

Turn to ReMS 1 for more information about the items mentioned above.

[^24]That's almost all you have to know about calculations with reals on the stack for the time being. Such capabilities did suffice for high flying applications already - see below. There are, however, far more places than just the stack where you may store and save your data in your WP43.

> orbited in space aboard three manned Skylab missions (1973-74). Solar research figured prominently among the wide assortment of experimental research conducted. Used as a backup to on-board computers, the HP-35 calculated predocking rocket burns necessary to align the Apollo Command Module with Skylab. In addition, the HP- 35 helped Skylab crews aim their telescopes at stars in attempts to measure ultraviolet radiation.

## (4p <br> HEWLETT PACKARD

## Addressing and Manipulating Objects in RAM

You know about the stack providing work space and temporary storage for your calculations. For long term storage, feel free to employ additional global registers, variables, and flags. The remaining chapters of this section will tell you how to use them.

The pictures on the next two pages show the address space of your WP43. Depending on the way you configure its memory, a subset of all these addresses will be accessible for you.

Depending on the stack size you choose, either $\mathbf{T}$ or $\mathbf{D}$ will be the top stack register, A - D (*) will be allocated for an 8-level stack, if applicable. The registers $\mathbf{I}$, J, and $\mathbf{K}$ (**) may carry parameters of statistical distributions (see pp. 103ff); I and $\mathbf{J}$ will also serve as index pointers in matrix editing (see pp. 173ff); $\mathbf{K}$ is used as default alpha register in some special cases. Unless required for these purposes, A, B, C, D, I, J, and K may be employed as additional global general purpose (GP) registers.

## Special registers and stack



Statistical data can be stored value by value, point by point, or en bloc in a matrix called STATS. The statistical sums will be accumulated automatically in a separate set of dedicated summation registers, not interfering with your other data (see OMS 2).

Turn overleaf to see what else is provided:
Each GP register (like each stack register) can hold any object you store therein - more than just a real number (see OMS 2). GP registers are beneficial, e.g., for keeping intermediate results for repeated use (see an example on the page after next).
A flag is an elementary item featuring only two states, set and clear (like a bit). Employ user flags for signaling whatever you want. ${ }^{57}$ Flags are most useful for program control, so more about them will be shown and told in OMS 3.

Registers and flags can be accessed individually - see the tables on pp. 63 ff. Addresses $\leq 111$ are for global data accessible from everywhere on your WP43, addresses $\geq 112$ are for local data (see pp. 247f for explanation of the latter).

Program steps will be covered in OMS 3 as well.

[^25]
## GP registers

Local registers for 1 routine

| R. $98=$ R210 |
| :--- |
| R. $97=$ R209 |
| $\ldots$ |



Global registers

Program steps

| 0000 |
| :--- |
| 0001 |
| 0002 |
| $\ldots$ |


|  |
| :--- |
| $\ldots$ |
| 9995 |
| 9996 |
| 9997 |
| 9998 |
| 9999 |

Steps are allocated as required and counted per program.

User flags



## Example for GP register use (with startup default settings):

## ASN SAVE RBR VIEW

$$
\sqrt{3+\left(\frac{1.09}{1.78}\right)^{2}} \times \frac{\ln \left[3+\left(\frac{1.09}{1.78}\right)^{2}\right]}{4 \cos \left[3+\left(\frac{1.09}{1.78}\right)^{2}\right]}=?
$$

Solution:
First, calculate the repeating term $3+\left(\frac{1.09}{1.78}\right)^{2}$ and store it:
1.09 ENTERT 1.78 (1)
EXP $x^{2} 3 \oplus$ STO (K)

Then solve the entire expression, e.g. like this:
$\sqrt{x}$
RCL (K) In
区
RCL (K TRI cos
(1)

4 (1)
solves the $1^{\text {st }}$ factor of the expression, solves the numerator,
2.234647088154349
solves the $2^{\text {nd }}$ part of the denominator,

That's it - solving this expression has become really easy this way.

Variables are named storage locations. Such names shall begin with a letter and may consist of up to 7 characters. Creating a new name, feel free to use all characters except,,$+- \times$, , ^, (, ), =, |, !, :, ;, comma, and space. Remember all variables (and menus) must carry unique names. Your WP43 decodes item names as follows:

- Upper and lower case letters are interpreted differently (e.g. $\mathbf{X} \neq \mathbf{x}$, CHECK $\neq$ Check $\neq$ check).
- Superscript and subscript characters are read as normal characters (e.g. $\mathbf{V a r}^{1}=$ Var $_{1}=$ Var1 - nevertheless, such names are always displayed as created, for easier reading).

Thus, check the item names provided in your WP43 (compiled and listed in the ReM, App. J) for possible name conflicts. In particular, do not attempt to call your variables $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, or $\mathbf{T}$ since these names are reserved for the lowest stack registers (but feel free to create variables named $\mathbf{x}, \mathbf{y}$, $\mathbf{z}, \mathbf{t}, \mathbf{X}_{2}, \mathbf{X X}$, etc.). And since items may appear in menus, we recommend that no two item names share the same first 6 characters.

Like each register, also each variable can hold any type of data. Clearing a register or variable is dispensable most times because you can just overwrite its content.

During input processing in memory addressing (e.g. while entering parameters for storing, recalling, copying, comparing, swapping, or clearing), you will not need all labels presented on the keyboard. Just a subset of 30 labels plus and $\square$ will do instead. The calculator mode supporting exactly these 30 is called transient alpha mode (TAM). It may be automatically set in memory addressing as shown below. Entering TAM, the operational keyboard is temporarily reassigned as shown here:

This kind of picture is called a virtual keyboard - dark red background is used to highlight changed active functionalities in TAM here. White print denotes primary functions here as well, such as the $1^{\text {st }}$ key in row 2 directly accessing (possibly stack) register A. ${ }^{58}$ Also all other lettered registers can be called directly in TAM - the stack registers $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, and $\mathbf{T}$ via unshifted softkeys.

Access to any number-
 ed register stays as easy as can be. $[\rightarrow]$ is for indirect addressing (see pp. 63ff), $\square$ for local addresses (see p. 247). What is not seen on this virtual keyboard cannot be called in TAM.

[^26]Variables defined at execution time will show up in VAR in alphabetical order - so you can easily select the variable of your choice by pressing the respective softkey. You can also access them via $\alpha$ (or create new variables this way): see pp. 63 f for how to do this.

Note that you will need $\square$ or for softkeys only. Menus are context sensitive in TAM. If a comparison is called, for instance, the -shifted row will look like this:


It allows for directly comparing $x$ with 0 . or 1 . as well as with the content of any register or variable (the latter via VAR, see p. 63).

If STO or RCL are called, on the other hand, the menu view will look like this instead:


This allows for storing all your specific calculator settings easily via STO Config and recalling them via RCL Config (see p. 85). Using STO Stack, you can store the entire stack in a block of 4 or 8 registers (depending on stack size set at execution time), RCL Stack recalls it. And max (or min) lets you work with the maximum (or minimum) of $x$ and the content of the source automatically (see the $/ O /$ ); you may use $\Delta$ as shortcut for max and $\nabla$ for min here. ${ }^{59}$

For commands operating on flags, SYS.FL grants access to the system flags provided:


[^27]For commands operating on labels (see OMS 3), PROG will be displayed instead of SYS.FL, granting access to all global labels specified. For all other operations asking for one trailing parameter, the menu will stay with a single row of items as pictured on p. 60.

TAM will be terminated as soon as sufficient characters are entered for the parameter of the respective operation. You may delete pending input keystroke by keystroke using $\boldsymbol{\square}$ and correct it if necessary - or just abort the pending command by EXIT; the latter will leave TAM immediately, returning to the mode set and the menu displayed before, if applicable.

For just looking up or verifying the content of a particular storage location without recalling it nor disturbing the stack, use VIEW.

## RBR VIEW

## Example:

DISP FIX 5

## as expected from previous example.

Note the view into register $\mathbf{K}$ is displayed adjusted to the left immediately below the status bar. This view will vanish with your next keystroke.

For inspecting a set of registers, use (RBR) instead. Press (FLAG STATUS for checking the status of all flags. See OMS 5 on pp. 277ff for more.

You are granted unlimited access to all the registers and user flags allocated; there is nothing like 'memory protection' on your WP43. You are the sole and undisputed master of its memory. Thus, taking care of it is your responsibility - keep suitable records to avoid inadvertently overwriting or deleting your precious data. ${ }^{60}$

You will not get 20000 program steps and 211 registers and 144 user flags all together at the same time - see the ReM, App. B, for reasons and for resource management.

[^28]
## Addressing Tables

## Parameterized Comparisons:

|  | User input Echo | IEST $x<?, x \leq ?, x=?, x \approx ?, x \neq ?, x \geq ?, x>$ ? <br> OP? _ (with TAM set), e.g. $x<$ ? |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2 | User input | (0.) or (1.) | Stack or lettered register (i.e. $\bar{Y}-\mathrm{T}$, (A) - (D), (L), (1) - (K) or variable defined ${ }^{61}$ | Register number (range as specified on p. 66) | [ $\rightarrow$ ] opens indirect addressing | $\boldsymbol{\alpha}^{62}$ <br> turns on alpha input mode (see pp. 203ff) for a (new) variable |
|  | Echo | $\begin{gathered} \text { OP } n ? \\ \text { e.g. } \\ x=0 . ? \end{gathered}$ | $\begin{gathered} \text { OP? } x \\ \text { e.g. } \\ x \geq ? Y \end{gathered}$ | $\begin{gathered} \text { OP? rnn } \\ \text { e.g. } \\ x \neq ? ~ r 23 \end{gathered}$ | OP? $\rightarrow_{\text {_ }}$ | OP? ${ }^{\text {¢ }}$ |
| 3 | User input <br> Echo | Compares $x$ with the number $\mathbf{0}$. | Compares $x$ with $\boldsymbol{y}$. | Compares $x$ with the content of register R23. | See pp. 64 and 67 for more about indirect addressing. | Name ${ }^{63}$ <br> OP? ' $x$ x' e.g. $x>$ ? 'ST1' |

$x>$ ? 'ST1' compares $x$ with the content of the variable 'ST1'.
Press (TEST $x>$ ? $\alpha$ (S)TT 1 ENTERT for this.

[^29]
## Examples: STO UNDF and INPUT UNDF will create UNDF while RCL UNDF will throw an error.

Register operations requiring just one register or variable trailing:


Stores $\boldsymbol{x}$ in the location where $\boldsymbol{r} 45$ is pointing to.
Swaps $\boldsymbol{x}$ and the content of the register where $\boldsymbol{l}$ is pointing to.
Increments the variable called Zähler1.

Clearing a single variable or register is most easily done by storing zero in it. Use DELITM for deleting a user variable from memory and freeing the space allocated for it (see p. 306).

[^30]
## Other operations requiring one trailing parameter:

| 1 User input <br> Echo |  BestF, CF), (FF, SF, (DSTACK, ERR, LOCR, PAUSE, (RM, SIM EQ, TDISP, TONE, WSIZE, ( $\rightarrow$ INT, bit and flag tests, etc. (see the $I O I$ for a complete list) $\begin{gathered} O P_{-} \text {(with TAM set), } \\ \text { e.g. FIX _ } \end{gathered}$ |
| :---: | :---: |
| 2 User input <br> Echo |  |
| User input Echo | Sets flag 109. |

Shows as many stack levels as specified in R12 (see pp. 67f).

Sets fix point format with the number of decimals stored in $\mathbf{A}$.

[^31]For... ...the valid number range is...

| Registers | $\left.\begin{array}{ll}0 \ldots 99 & \begin{array}{l}\text { for direct addressing of } \\ \text { global numbered registers }\end{array} \\ .0 \ldots .98 & \begin{array}{l}\text { for direct addressing of } \\ \text { local registers }\end{array} \\ 0 \ldots 210 & \begin{array}{l}\text { for indirect addressing } \\ (\geq 112 \text { with local registers only) }\end{array}\end{array}\right\}$upper limits <br> depend on <br> current <br> allocation |
| :---: | :---: |
| User flags | 0 ... 99 for direct addressing of global numbered flags <br> $.0 \ldots .31$ for direct addressing of local flags if at least one local register is allocated <br> $0 \ldots 143$ for indirect addressing ( $\geq 112$ with local flags only) |
| Decimals | 0.. 15 (entering any digit except 0 or 1 will terminate |
| Integer bases | $2 \ldots 16$ waiting for a $2^{\text {nd }}$ digit and close input) |
| Bit numbers | $0 \ldots 63$ |
| Word size | $1 . . .64$ bits |

Specifying low numbers and numeric addresses, you may key in 5 ENTERT instead of (0) 5, for instance.

Remember some registers and user flags may be addressed by single letters as well. System flags are called by their names; for some of them, one-letter shortcuts exist (cf. p. 58). Variables are generally called by their names.

Please see the $\operatorname{Re} M$ for all other parameters and their valid ranges, as well as for all system flags provided.

## Indirect Addressing - Working with Pointers

Parameters for many functions can be specified by supplying a register or variable pointing to the respective parameter.

## Example:

Assume $x=12.34, j=45.67$, and $r 12=7.89$. Then $\ldots$
STO (J)
RCL $\rightarrow$ (J) will return 7.89 since $J$ is containing 12.34 at execution time and thus is pointing to R12. And then...
$\mathrm{SF} \rightarrow \mathrm{X} \quad$ will set flag 7 , and...
DISP FIX $\rightarrow \mathrm{X}$ will display 7.8900000 showing 7 decimals.

Since the content of the register specified is used as a pointer to the register wherefrom we want to read (or whereto we want to write), this method is called indirect addressing.

Each and every register of your WP43 can be used for indirect addressing. ${ }^{66}$ And each and every register can be accessed this way. Both statements apply to local registers as well.

Indirect addressing is most beneficial in programs when the parameter for a function is calculated automatically (see examples in OMS 3). It may be even combined with store and recall arithmetic as explained below, and become a mightier tool yet.

## Store and Recall Arithmetic

As mentioned in f. 64 on p. 64, arithmetic operations (and two conditional picks, max and min) can be performed upon the contents of registers or variables by pressing STO or RCL followed by the respective operator $(\oplus, \boxed{\square}, \boldsymbol{x}, \square, \boxed{\Delta}$, or $(\nabla)$, trailed by the address or name of the storage space.

[^32]
## Example for store arithmetic:

123.4

STO K closes input and subtracts 123.4 from $\boldsymbol{k}$. The difference is stored in $\mathbf{K}$. The stack and $\mathbf{L}$ remain unchanged here.

The same result could be achieved by the keystroke sequence

```
123.4
RCL (K
x\geqslanty
-
STO K
RCL (L) but this is far clumsier (replacing one step by five) and would cost one stack register in addition.
```

The general rule for store arithmetic reads:


Example (from the HP-67 OHPG):
During harvest, farmer Flem Snopes trucks tomatoes to the cannery for three days. On Monday and Tuesday he hauls loads of 25 tons, 27 tons, 19 tons, and 23 tons, for which the cannery pays him $\$ 55$ per ton. On Wednesday the price rises to $\$ 57.50$ per ton, and Snopes ships loads of 26 tons and 28 tons. If the cannery deducts $2 \%$ of the price on Monday and Tuesday because of blight on the tomatoes, and 3\% of the price on Wednesday, what is Snopes' total net income?

Solution:

DISP FIX 2
25. ENTERT $27 \oplus$


55 区
STO [J

$$
\left\{\begin{array}{c}
+ \\
\overline{\times} \\
/ \\
\max \\
\min
\end{array}\right\} \text { current } \boldsymbol{x}
$$

## Standard in financial calculations

Total of Monday's and Tuesday's

## tonnage

Gross amount for these days
Take J for accounting

| 2 (EIN) \% | 103.40 | Deduction for these two days |
| :---: | :---: | :---: |
| STO) ${ }^{\text {d }}$ | 103.40 | Subtracted from the total in J |
| 26. ENTERT $28 \pm$ | 54.00 | Wednesday's tonnage |
| 57.5 区 | 3105.00 | Gross amount for Wednesday |
| STO $\dagger$ (J) | 3105.00 | Added to the total in $\mathbf{J}$ |
| $3 \%$ | 93.15 | Deduction for Wednesday |
| STOT ${ }^{\text {R }}$ | 93.15 | Subtracted from the total in J |
| RCL (J) | 8078.45 | Snopes' total net income from his tomatoes |

In analogy to store arithmetic, recall arithmetic is provided as well.

## Example for recall arithmetic:

78.91

RCLD (2) 3 closes numeric input and divides 78.91 by $r 23$. This operation is performed in $\mathbf{X}$ alone. $\mathbf{L}$ is loaded with 78.91. All other stack contents and $r 23$ stay unchanged.

Alternatively, the same result could be achieved by the sequence
78.91

## RCL (2) 3

(1)
but that would replace one step by two and also cost one additional stack register. And $\mathbf{L}$ would differ here, too.

Gain is less than with store arithmetic. Nevertheless, also recall arithmetic is provided on your WP43 for symmetry reasons and since the HP-42S featured it.

## General rule for recall arithmetic:

$$
\text { new } \boldsymbol{x}=\text { old } \boldsymbol{x}\left\{\begin{array}{c}
+ \\
\bar{x} \\
\times \\
\max \\
\min
\end{array}\right\} \begin{gathered}
\text { content of the } \\
\text { register or variable } \\
\text { specified }
\end{gathered}
$$

Stack-wise, both store and recall arithmetic work like monadic functions.

Note these functions may operate on each and every register or variable provided, also on the stack and even on $\mathbf{L}$.

## Example:

What does RCD- Y ?

| $\mathbf{L}$ | old value | 1.0 |
| ---: | ---: | ---: |
|  |  |  |
| $\mathbf{Z}$ | 3.2 | 3.2 |
| $\mathbf{Y}$ | 2.1 | 2.1 |
| $\mathbf{X}$ | 1.0 | -1.1 |
|  |  | RCL $-\mathbf{Y}$ |

Register arithmetic is most beneficial in automatic routines. Indirect addressing may be combined with it, then looking like STO $\boldsymbol{x} \rightarrow \mathbb{K}$, for example.

Although both these techniques have been more important in times when program memory was very limited and processor speed was low, they may be still beneficial today, in particular when you are out to create compact routines executing quickly. See pp. 228ff in OMS 3 for examples and advantages, and look up the $I O /$ for more information.

Up to here you have learned how to address registers, flags, and variables on your WP43. Various methods for accessing these storage places have been introduced. Such calculatory capabilities did suffice for orbiting the Earth and controlling spacecraft docking maneuvers.

There are, however, far more objects than just reals you can manipulate on your WP43 - let's introduce them to you.

## SECTION 2: DEALING WITH VARIOUS OBJECTS AND DATA TYPES

## Some Display Basics

The screen is your window to your WP43 - there you see what is going on and the latest results. Going top down, you find ...

- the status bar,
- space for up to 4 rows of standard numeric output (and more), and
- the menu section displaying up to 3 rows of items (cf. pp. 27f).


Let's look to the numeric rows first - the status bar will be covered later:

1. For startup default DSTACK 4, the top (T) numeric row shows $\boldsymbol{t}$. It will display the output of SHOW (see p. 150) instead, if applicable.

Its left side is for VIEW (cf. p. 62), if applicable, and for echoing command input until it can be executed with all required parameters entered; and will be displayed using ${ }^{f}$ and ${ }^{\boldsymbol{g}}$ until they are resolved (if you pressed or $\square$ erroneously, however, recovery is as easy as $\square \square \square \square=\mathrm{NOP}$ ). You may edit a pending operation keystroke by keystroke using $\boldsymbol{\uplus}$, or cancel it by EXIT (cf. p. 62).
2. For DSTACK 4 or $3, \mathbf{Z}$ numeric row displays $z$.

Its left side is for any error message or output of a binary test, if applicable. $\mathbf{Z}$ row may also be used by SHOW if the top row does not suffice, or for temporary information heading $\boldsymbol{z}$, if applicable.

Temporary information (TI) is any displayed information exceeding the plain contents of $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$ in $R U M$. It will vanish with next keystroke:

- If it is an error message, $\boldsymbol{4}$ or EXIT will just clear it (cf. p. 53) any other keystrokes will be executed in addition.
- Without an error message, also $\boldsymbol{\Psi}$ and EXIT will execute.
- $\square \square$ and $\square$ will remove any $T /$ from the screen most easily and free of side effects. ${ }^{67}$

3. For DSTACK 4, 3, or 2, Y numeric row shows $\boldsymbol{y}$.

Its left side is for $T /$ heading $\boldsymbol{y}$, if applicable.
4. The bottom (X) numeric row displays $\boldsymbol{x}$.

Its left side is for...
a. showing $T /$ heading $x$, if applicable,
b. echoing numeric or alphanumeric input (see pp. 25 and 203ff, respectively). Note $\mathbf{X}$ can take up to 42 digits, a sign, and a radix mark in startup default numeric format or some 40 alphanumeric characters. You may edit pending input keystroke by keystroke using $\boldsymbol{\uplus}$. Numeric input will be checked and interpreted as soon as it is closed, according to the calculator settings at closure time.

## Data Types Supported

You learned in OMS 1 how your WP43 calculates with reals. It can do more for you: it can deal with integers, fractions, complex numbers, angles, times, and dates in various formats. It can also handle real and complex vectors, matrices, and alphanumeric (a.k.a. text) strings. ${ }^{68}$

But how shall your WP43 learn about the particular meaning of your input? Some examples will explain (each input in startup default format):

[^33]| Input | Display | Meaning |
| :---: | :---: | :---: |
| 12345.678901 EXIT | 12345.678901 | Real numbers, see pp. 84ff |
| 12.3 E 45 ENTERT | $12.3 \times 10^{45}$ |  |
| 901.23.4567 ENTERT | > $90123 / 4567$ | Fraction, see pp. 136ff |
| 270 ( $\rightarrow$ ¢ $\rightarrow$ MUL | $1.5 \pi$ | Angle shown in multiples of $\pi$ |
| 123.45678901 d.ms | $123^{\circ} 46^{\prime} 7.89^{\prime \prime}$ | Sexagesimal angle; see pp. 139ff also for other angular formats supported |
| 1234567890 ENTER ${ }^{\text {d }}$ | 1234567890 | Integers of various bases and lengths, see pp. 149ff |
| 1234567890 \# (H) | 1234567890 |  |
| 10100110111 \# [2] | $10100110111_{2}$ |  |
|  | 12.3-i $\times 4.56$ | Complex numbers in rectangula or polar notation; mantissa plus exponent format is settable as well; see pp. 164ff |
| 12.3 col 4.56 - | $12.36-4.56^{\circ}$ |  |
| 1.0203045 | 0001-02-03 | Date, see pp. 197f |
| 12.345678901 (h.ms | 12:34:56.789 01 | Sexagesimal time, see pp. 199f |

Some of these inputs may be interpreted and displayed differently depending on particular mode settings. Startup default displays are printed below in light blue, further widespread formats in grey fields:

|  | DECIM. set | DECCM. |
| ---: | ---: | ---: |
| GAP 4 | 12345.678901 | 12345,678901 |
| GAP 3 | 12345.678901 | 12345,678901 |
| GAP 2, GAP 1, or GAP 0 | 12345.678901 | 12345,678901 |



| Y.MD | D.MY | M.DY |
| :---: | :---: | :---: |
| $0001-02-03$ | 01.02 .0304 | $01 / 02 / 0304$ |

Obviously, your WP43 reacts to your input very flexibly. And you can easily recognize the various data types and formats looking at the screen.

## Processing Data Types

How can you use and combine data of various types in calculations? The matrix below lists in its $1^{\text {st }}$ column ten data types your WP43 supports; and it shows what will happen when you combine various objects: an object of the $D T$ as indicated in one of the narrow columns at right $(y)$ plus or minus an object of the $D T$ in column $1(x)$ will return an object of the $D T$ at the intersection (thus, wherever a $D T$ number is printed at the intersection, the corresponding combination is legal for addition or sub-
traction). ${ }^{69}$ Grey fields point to restrictions, rose fields to asymmetries.

| DT and meaning | $y$ |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 Z Long integer ${ }^{70}$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $2 \mathbb{R}$ Real number | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 |
| 3 C Complex number | 3 | 3 | 3 | - | - | - | 7 | 9 | 9 | 3 |
| 4 Angle (in various formats) ${ }^{71}$ | 4 | 4 | - | 4 | - | - | 7 | - | - | 4 |
| 5 Time interval (in H.MS) | 5 | 5 | - | - | 5 | - | 7 | - | - | 5 |
| 6 Date (in various formats) | $6^{72}$ | $6^{72}$ | - | - | - | $1^{73}$ | 7 | - | - | $6^{72}$ |
| 7 Text string ${ }^{74}$ | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 8 Real matrix or vector | 8 | 8 | 9 | - | - | - | 7 | 8 | 9 | - |
| 9 Complex matrix or vector | 9 | 9 | 9 | - | - | - | 7 | 9 | 9 | - |
| 10 Short integer ${ }^{70}$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | - | - | 10 |

## Example:

A complex number (DT 3) plus or minus a real number (DT 2) will result in a complex number. And a text string plus a time will append the time string as displayed to said text string.

The following matrix shows the resulting data types of products and ratios in the same way (note that neither dates nor text strings can be multiplied or divided, and actually plain numbers cannot be divided by an angle or a time - cf. f. 20 on p. 29):
${ }^{69}$ Else error 24 (Illegal input data type for this operation) will be thrown.
${ }^{70}$ If integers of different types (DT 1 and 10) or different bases are combined by an operation listed here, output will be an integer of the type and base given in $\mathbf{Y}$.
${ }^{71}$ Angular output will be tagged according to the current angular display mode.
${ }^{72}$ A date $\boldsymbol{y}$ plus (minus) a number $\boldsymbol{x}$ takes the integer part of $\boldsymbol{x}$ and adds (subtracts) the respective number of days to (from) said date $\boldsymbol{y}$. A number $\boldsymbol{y}$ plus a date $\boldsymbol{x}$ works alike. A number $\boldsymbol{y}$ minus a date $\boldsymbol{x}$ is not supported. Consult the ReM, App. B, for date limits.
${ }^{73} \mathrm{~A}$ date minus a date returns an integer number of days. Plus is illegal here.
${ }^{74}$ Only additions are supported: First, any numeric $\boldsymbol{x}$ or $\boldsymbol{y}$ will be converted to a text string according to the display format at execution time; for a matrix, its descriptor will be used (e.g. [3×4 C Matrix], see further below). Then, string $x$ will be appended to string $y$.

|  | An object $\boldsymbol{y}$ of $D T$. |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 10 |
| times an object $\boldsymbol{x}$ of the $D T$ below returns a product of the $D T$ printed at the intersection. |  |  |  |  |  |  |  |  |
| 1 Z Long integer ${ }^{70}$ | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 10 |
| $2 \mathbb{R}$ Real number | 2 | 2 | 3 | 4 | 5 | 8 | 9 | 2 |
| 3 C Complex number | 3 | 3 | 3 | 3 | - | 9 | 9 | 3 |
| 4 Angle ${ }^{75}$ | 4 | 4 | 3 | 2 | - | 8 | 9 | 4 |
| 5 Time interval | 5 | 5 | - | - | - | - | - | 5 |
| 8 Real matrix or vector | 8 | 8 | 9 | 8 | - | 8 | 9 | 8 |
| 9 Complex matrix or vector | 9 | 9 | 9 | 9 | - | 9 | 9 | 9 |
| 10 Short integer ${ }^{70}$ | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 10 |
| divided by an object $\boldsymbol{x}$ of the $D T$ below returns a ratio of the $D T$ printed at the intersection. |  |  |  |  |  |  |  |  |
| 1 Z Long integer ${ }^{70}$ | 1/2 ${ }^{76}$ | 2 | 3 | 4 | 5 | 8 | 9 | 10 |
| $2 \mathbb{R}$ Real number | 2 | 2 | 3 | 4 | 5 | 8 | 9 | 2 |
| 3 C Complex number | 3 | 3 | 3 | 3 | - | 9 | 9 | 3 |
| 4 Angle | 2 | 2 | 3 | 2 | - | 8 | 9 | 2 |
| 5 Time interval ${ }^{77}$ | 2 | 2 | - | - | 2 | - | - | 2 |
| 8 Real matrix ${ }^{78}$ | 8 | 8 | 9 | 8 | - | 8 | 9 | 8 |
| 9 Complex matrix ${ }^{78}$ | 9 | 9 | 9 | 9 | - | 9 | 9 | 9 |
| 10 Short integer ${ }^{70}$ | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 10 |

The following two matrices are for powers and roots:

[^34]| ... raised to a power of $x \geq 0$ of the $D T$ below returns a result of the $D T$ printed at the intersection. | A number $\boldsymbol{y}>0$ of $D T \ldots$ |  |  |
| :---: | :---: | :---: | :---: |
|  | 1 | 2 | 10 |
|  |  |  |  |
| 1 or 10 (i.e. long or short integer) ${ }^{70}$ | 1 | 2 | 10 |
| 2 R Real number | 2 |  |  |
| $\ldots$ raised to the power of $\boldsymbol{x}<\boldsymbol{0}$ (with $\boldsymbol{x}$ of $D T 1,2$, or $10^{70}$ ) returns a real number. | $2(1 / 10)^{79}$ |  |  |
| The $\boldsymbol{x}^{\text {th }}$ root ( $x>0$ of the $D T$ below) of $y$ returns a result of the $D T$ printed at the intersection. |  |  |  |
| 1 or 10 (i.e. long or short integer) ${ }^{70}$ | $1^{80} / 2$ | 2 | $10^{80}$ |
| 2 R Real number | 2 |  |  |


|  | A number $\boldsymbol{y}<0$ of $D T \ldots$ |  |  |
| :---: | :---: | :---: | :---: |
|  | 1 | 2 | 10 |
| ... raised to a power of $x \geq 0$ of the $D T$ below returns a result of the $D T$ printed at the intersection. |  |  |  |
| 1 or 10 (i.e. long or short integer) ${ }^{70}$ | 1 | 2 | 10 |
| 2 R Real number | $2 / 3$ |  |  |
| raised to the power of $\boldsymbol{x}<\mathbf{0}$ (with $\boldsymbol{x}$ of $D T$ 1,2 , or $10^{70}$ ) returns a result of $D T$ : | $2 / 3(1 / 10)^{79}$ |  |  |
| The $x^{\text {th }}$ root ( $x>0$ of the $D T$ below) of $y$ returns a result of the $D T$ printed at the intersection. |  |  |  |
| 1 or 10 (i.e. long or short integer) ${ }^{70} \quad \boldsymbol{x}$ odd | 180/2 | 2 | $10^{80}$ |
| 2 (or 1 or 10 with even $\boldsymbol{x}$ ) | 3 |  |  |

Any number of $D T 1$ or 2 raised to complex power will return a complex number, as well as any complex number raised to arbitrary power. Raising a short integer to complex power is not supported, nor are powers involving DTs 4, 5, 6, 7, 8, or 9.
${ }^{79} D T 1$ or 10 can only result for $1^{-1}$ or $(-1)^{-1}$ here.
${ }^{80}$ Square roots of squares, cube roots of cubes, logarithms of powers, etc. stay in the $D T$ they were. For other integers $y>0$, results become real (for $D T 1$ ) or truncated short integers (for $D T$ 10); for $y<0$, results become complex instead of real.

This is the matrix for logarithms:

|  | A number $\boldsymbol{y}>\boldsymbol{0}$ of $D T \ldots$ |  |  |
| :---: | :---: | :---: | :---: |
|  | 1 | 2 | 10 |
| .. combined in $\log _{x} y$ with a number $x>0$ of the $D T$ below returns a result of the $D T$ printed at the intersection. |  |  |  |
| 1 or 10 (i.e. long or short integer) ${ }^{70}$ | $1^{80} / 2$ | 2 | $10^{80}$ |
| $2 \mathbb{R}$ Real number | 2 | 2 | $10^{80}$ |

And for $\boldsymbol{y}<0$ of $D T 1$ or 2 , logarithms will return a complex result if this is allowed (see the chapters about complex numbers below); logarithms of negative short integers are not supported.
Here follows the matrix for some popular monadic functions more:

|  | A number $\boldsymbol{x}>0$ of $D T \ldots$ |  |  |
| :---: | :---: | :---: | :---: |
|  | 1 | 2 | 10 |
| ... processed by an operation printed below returns a result of the $D T$ printed at the intersection. |  |  |  |
| (19), $16 \times, \sqrt{x}, \sqrt[3]{x}$ | $1{ }^{80} / 2$ | 2 | $10^{80}$ |
| (1n) , $\boldsymbol{e}^{\boldsymbol{x}}$ | 2 | 2 | $10^{80}$ |
| $10^{x}$ | 1 | 2 | 10 |

For roots of negative operands see previous page.
This is for integer divisions and remainders:

|  | A number $\boldsymbol{y}$ of $D T \ldots$ |  |  |
| :---: | :---: | :---: | :---: |
|  | 1 | 2 | 10 |
| IDIVR-divided by a number $\boldsymbol{x}$ of the $D T$ below returns an integer ratio in $\mathbf{X}$ and a remainder in $\mathbf{Y}$ of the data types printed at the intersection. |  |  |  |
| 1 or 10 (i.e. long or short integer) ${ }^{70}$ | 1; 1 | 1; 2 | 10; 10 |
| 2 R Real number | 1; 2 | 1; 2 | 10; 2 |

Additionally, explicit $D T$ conversions are available where necessary:

| Any closed object $x$ of $D T \ldots$ |  |  |  |  | ... will be converted in an object $\boldsymbol{x}$ of the $D T$ below by the keystrokes printed at the intersection. |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $\mathbf{1}$ $\mathbf{2}$ <br> long <br> integer real | $4$ <br> angle | 5 <br> time | 6 <br> date | 10 <br> short integer |  |
| \# (1) | - | - | - | \# (1) | 1 IT Long integer |
| . d |  |  |  |  | $2 \mathbb{R}$ Real number |
| ¢ $¢$ |  | - | - | - | 4 Angle |
| h.ms | - | - | - | - | 5 Time |
| \# ... | - | - | - | \# ... | 10 Short integer |

## Recognizing Calculator Settings and Status

Some settings are obvious: radix marks and gap settings are recognized in the numeric display immediately; so are date and time display modes (Y.MD / D.MY / M.DY and CLK24 / CLK12) in the time string in the status bar. Also program-entry mode (PEM) is easily recognized (see pp. 212ff).
Further modes and system states as well as settings for specific data types are indicated in the status bar. the following specific symbols may appear trailing the date and time string there, listed below from left to right in various sets - indicators shown in startup default are printed in a light blue field again: ${ }^{81}$

| Indicator | Set by | Deleted by | Explanation, remarks |
| :---: | :--- | :--- | :--- |
| $\mathbb{C}$ | SF CPXRES | CF CPXRES | With CPXRES, complex results of <br> operations on reals are allowed, <br> like $\sqrt{-1}$. Else a domain error will <br> be thrown in such a case. |
| $\mathbb{R}$ | CF CPXRES | SF CPXRES |  |


| $\llcorner$ | CF POLAR | SF POLAR | Rectangular or polar notation <br> chosen for displaying complex <br> numbers. |
| :---: | :--- | :--- | :--- |
| $\square$ | SF POLAR | CF POLAR |  |

[^35]| Indicator | Set by | Deleted by | Explanation, remarks |
| :---: | :---: | :---: | :---: |
| $4^{\circ}$ | DEG | setting any other ADM | Current angular display mode (ADM) setting: decimal degrees, grades or gon, radians, multiples of $\pi$, mils, and sexagesimal degrees. |
| $4^{9}$ | GRAD |  |  |
| $4^{\text {r }}$ | RAD |  |  |
| 4 $\pi$ | MULT |  |  |
| $4^{-}$ | MIL |  |  |
| $4 "$ | d.ms |  |  |

$\begin{array}{|c|l|l|l|}\hline / \text { max } \\ \text { or } \\ / \mathbf{2 3 4 5}\end{array}$ SF DENANY $\left.\quad \begin{array}{l}\text { Can only be } \\ \text { modified by } \\ \text { DENMAX }\end{array} \begin{array}{l}\text { Fraction display settings. The } \\ \text { current value of D.MAX (i.e. the } \\ \text { maximum displayable denomina- } \\ \text { tor) is shown behind the fraction }\end{array}\right\}$

| $64: 1$ | 1COMPL |  |
| :---: | :--- | :--- |
| $64: 2$ | 2COMPL | setting any <br> other integer |
| sign mode |  |  |
| (ISM) |  |  |

Settings for short integers. First two digits tell the word size, the symbol after the colon tells the ISM. Startup default is 64 bits (the maximum) and 2's complement. CARRY and OVERFLow may follow the ISM but are only lit if set.

| wrap | M.WRAP | M.GROW | Matrix grow mode, displayed in- <br> stead of the short integer settings <br> above as long as a matrix is open <br> for editing. |
| :--- | :--- | :--- | :--- |
| grow | M.GROW | M.WRAP | ( |


| END | ENDP | BEGINP |
| :--- | :--- | :--- |
| BEG | BEGINP | ENDP |

Time Value of Money payment mode, displayed instead of the short integer settings above as long as TVM is open (see OMS 5).

| Indicator | Set by | Deleted by | Explanation, remarks |
| :---: | :---: | :---: | :---: |
| A | $\Delta$ if $\alpha$ is set | ENTERT or <br> EXIT in AIM | © or SF ALPHA will set alpha input mode (AIM). For A or $\alpha$, upper or |
| $\alpha$ | V if A is set | unless in a menu, CF ALPHA | lower case letters can be entered then. AIM will start in $\alpha$ for editing equations or creating variables. |


| $\mathbf{4}$ | CF ssizE8 | SF ssizE8 | Stack size. |
| :---: | :--- | :--- | :--- |
| $\mathbf{8}$ | SF ssIZE8 | CF ssIZE8 |  |


| 0 | program wait ing for user input | program running | Will also be lit if a program is stopped by EXIT or R/S - then $\mathbf{\Theta}$ will be cleared by next keystroke. |
| :---: | :---: | :---: | :---: |
| \% | see remarks | WP43 idling | Function executing in RUM. Not lit with program running. |
| 实 | see remarks |  | Program running (see OMS 3). |


| $\square$ | timer running | idle timer | See the TIMER (a.k.a. stopwatch) <br> application in OMS 5. |
| :---: | :--- | :--- | :--- |


| $\boldsymbol{*}$ | serial I/O in <br> progress | idle commu- <br> nication line |  |
| :---: | :--- | :--- | :--- |
| 国 | data being <br> sent to printer |  |  |


| $\mathbf{U}$ | USER | USER | Toggles user mode (see OMS 6). |
| :---: | :--- | :--- | :--- |


| $\boldsymbol{\psi}$ | see remarks |  | Lit if power is supplied through the <br> USB cable, else clear. |
| :---: | :--- | :--- | :--- |
| $\square$ | low battery | supplied volt- <br> age $>2.5 \mathrm{~V}$ | A low battery will reduce processor <br> speed. Your WP43 may shut off <br> when voltage drops below 2.0 V. |

The startup default configuration is indicated in a status bar like this:

$$
\text { 2022-10-29 14:57 RL4 } \boldsymbol{x}^{\circ} / \max \quad 64: 2
$$

Choosing M.DY and 12h time format, CPXRES, POLAR, MULTt, DENFIX and a 4-digit D.MAX, unsigned short integers, having CARRY, OVERFLow, AIM,
and SSIZE8, a program waiting for input, timer and printer running in background, user mode, and a low battery would be reflected in a status bar like

## 10/29/2202:58pm CO4л /6789f 64:ט

See below and look up the ReM for these system flags and calculator


## Getting Special Information: RBR, STATUS, VERS, etc.

Some commands and tools use the display in special ways. They are listed below:

1. RBR allows browsing all registers currently allocated (see pp. 277f).
2. FLAG STATUS returns free space available, memory currently used, user and system flags set (see pp. 279f).
3. The Matrix Editor for filling matrices and modifying their elements is described comprehensively on pp. 173ff.
4. (厅) calls the timer or stopwatch application (see pp. 280f).
5. a.FN FBR browses all characters defined in the fonts provided.

Further commands throw temporary information as defined on p. 71:
a. (ERR) and (MSG) display the corresponding error message. See the IOI and App. C of the ReM.
b. [cov), $[\boldsymbol{r}],\left[\mathbf{s}_{\mathbf{x y}}\right]$, VIEW, (WDAY), $[\hat{\mathbf{X}}]$, and $(\hat{\mathbf{y}})$ return results with headers.
c. Commands returning two or three values at once (like $\rightarrow P, R \leftrightarrow$,
 tag their output (see e.g. pp. 20 and 105ff).
d. (VERS?) generates a message showing the firmware version running on your WP43 ( $(\mathbf{W H O})$ works in a similar way):
WP43 v0.23.1 2022-10-14

A few far-reaching commands (like CLALL or RESET) ask you for confirmation before executing. Answer either

- Yes (by pressing (3) or ENTERT or XEQ) or
- No (by pressing 10 or EXIT or $\boldsymbol{\top}$ );
any other input will be ignored. Note that such actions explicitly confirmed cannot be undone by $n$.


## Localising Numeric Output

You can summon display preferences for reals, times, and dates all at once according to your region's customs using dedicated commands (all contained in the $2^{\text {nd }}$ view of DISP). In the table overleaf, ...

DSP DISP
E $\begin{gathered}\text { 公 } \\ M\end{gathered}$

- radix mark denotes the decimal separator; ${ }^{82}$
- GAP states the digit group interval - after $\boldsymbol{n}$ digits a narrow blank is displayed (cf. examples on p. 73); this follows ISO 80000-1. ${ }^{83}$
- Date stands for the date format (Y.MD, D.MY, or M.DY). ${ }^{84}$
- JG states the year the Gregorian calendar was introduced in the particular region, typically replacing the Julian calendar (or national calendars in East Asia); check the dates in Wikipedia. ${ }^{85}$
- background colors are chosen like on pp. 73f.

[^36]Most people using radix commas employ multiplication dots while those using radix points need a multiplication cross to avoid misunderstandings （causing further ambiguities in vector multiplication，see pp．181ff）．

| Com－ <br> mand | GAP | Radix <br> mark | Time | Date | JG | Remarks |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| SETCHN | $\mathbf{4}^{86}$ | point | 24 h | Y．MD | 1949 |  |
| SETJPN | 3 | point | 24 h | Y．MD | 1873 |  |
| SETIND | $3^{87}$ | point | 24 h | D．MY | 1752 | Applies to former British India <br> （i．e．to India，Pakistan，Nepal， <br> Bhutan，Myanmar，Bangladesh， <br> and Sri Lanka）． |
| SETEUR | 3 | comma | 24 h | D．MY | 1582 | Also applies to Latin America <br> and－with other JGs－to Indo－ <br> nesia，South Africa，the area of <br> the former Soviet Union，and <br> Vietnam． |
| SETUK | 3 | point | $12 h$ | D．MY | 1752 | Also applies to Australia and <br> New Zealand． |
| SETUSA | 3 | point | 12 h | M．DY | 1752 |  |

Note that you can store the following settings and formats of your WP43 collectively at one location：the entire decimal display format（see next

[^37]
chapter), angular display mode, date and time display settings, parameters of integer and fraction display modes, curve fit model chosen, rounding mode, and the status of all system flags. STOCFG stores this configuration in the register or variable you specify. ${ }^{89}$ RCLCFG recalls such information and will set (or reset) your WP43 accordingly.

## Real Numbers: Changing the Display Format

The numbers you calculate with are reals most frequently (cf. OMS 1). Any number you enter using one $\square$ and/or an $E$ is interpreted by your WP43 as a real unless there is additional information given (cf. pp. 72f). The majority of functions provided by your WP43 operate on reals.

As soon as input of a real is closed, its mantissa will be displayed right adjusted as far as possible (cf. pp. 25f). Startup default format (ALL 0) shows all digits of the number if less than 16 are needed to do so. Your WP43 will automatically turn to mantissa plus exponent format (MEF) if the number is too big or too small to be displayed with a fixed point. This prevents missing unexpectedly big or small results. ${ }^{90}$
There are two flavors of MEF provided: SCI is called scientific notation. SCI 3 will display the age of the universe like $1.380 \times 10^{10}$. ENG looks almost like SCl but the exponent will always be a multiple of three (e.g. $\quad 13.80 \times 10^{9}$ for ENG 3), corresponding to the S/ unit prefixes - thus it is called the engineer's notation.
Furthermore, FIX fixes the radix mark on the screen (thus called fixed point notation) while it floats in the other formats - see the examples.

[^38]You can specify the number of decimals you want to see with FIX, SCI, or ENG (note the parameter of FIX and SCI specifies the number of decimals to be shown, while for ENG it specifies the total number of digits to be displayed for the mantissa, minus 1).

## Examples:

| Format Input | Startup default format (ALL 0, ALLENG) | FIX 5 | SCl 5 |
| :---: | :---: | :---: | :---: |
| $\begin{array}{r} 9871.234567 \\ \text { ENTERT } \end{array}$ | 9871.234567 | 9871.23457 | $9.87123 \times 10^{3}$ |
| (11x | $1.013044512235762 \times 10^{-4}$ | 0.00010 | $1.01304 \times 10^{-4}$ |

The following examples vary according to popular choices for GAP, radix mark, and multiplication symbol (compare p. 73): ${ }^{91}$

|  | FIX 2 | ENG 6 |
| :---: | :---: | :---: |
| 89012345.678 + | -89 012345.68 | $-89.01235 \times 10^{6}$ |
| EXIT | -89 012 345,68 | $-89,01235 \cdot 10^{6}$ |
|  | -8901 2345.68 | $-89.01235 \times 10^{6}$ |

For ALL, you can choose whether it shall overflow either to SCI or ENG, using the system flag ALLENG (or flag A). And by specifying a parameter for ALL you can tell your WP43 up to how many decimal zeros you allow before the format shall be switched.

Once you have chosen a display format, (DSP lets you change the parameter for FIX, SCI, ENG, or ALL without calling the specific command again. DSP is at $\square+E$, no need to open DISP for it.

[^39]Example (beginning with startup default settings, i.e. ALL 0):

| -700 | -700_ |  |
| :---: | :---: | :---: |
| $11 / x$ | $-1.428571428571429 \times 10^{-3}$ | turns to SCI. |
| DSP 2 | $-1.428571428571429 \times 10^{-3}$ | allowing for $\leq 2$ zeros is cient to return the display |
| DSP 3 | -0.001 428571428571 | but 3 zeros are ok. |
| 10 (1) | -1.428 $571428571429 \times 10^{-4}$ | this is too small again. |
| (SF) A | -142.8571428571429×10-6 | turns to ENG. |
| (DSP 4 | -0.000 142857142857 | 4 zeros are ok. |
| 10 (1) | -14.285 $71428571429 \times 10^{-6}$ | too small again. |
| (CF) A | -1.428 $571428571429 \times 10^{-5}$ | returns to SCI. |

Almost all functions for real number display format control are found in DISP: FIX, SCI, ENG, ALL, GAP, rounding, and more. See the ReM.

## Real Numbers: Squares and Cubes and Their Roots

$\sqrt{x}$ is on the keyboard of your WP43, while $x^{2}, x^{3}$, $\sqrt{ }\left(1+x^{2}\right)$, and $\sqrt[3]{x}$ live in EXP (cf. p. 28; note a square can be produced by ENTERT $X$ without EXP as well.) The following example solves some of the most popular problems of ancient mathematics:


What size square has the same area as a circle whose radius is 3 arbitrary units? And what size cube has the same volume as a sphere whose radius is 3 again? And what can we tell about their surface areas?

Solutions (with FIX 3):
The area of a circle is $A_{C}=\pi r^{2}$. The area of a square is $A_{s q}=a^{2}$. The volume of a sphere is $V_{S}=\frac{4}{3} \pi r^{3}$, while its surface is $A_{S}=4 \pi r^{2}$. The volume of a cube is $V_{c u}=a^{3}$, while its surface is $A_{c u}=6 a^{2}$.

Thus,


Furthermore,

## $3 x^{3} \pi x$

$4 \times 3$ returns 113.097 for the volume of the sphere. Then
$\sqrt[3]{x}$
$x^{2} 6 x$
$\mathbf{3} \mathbf{x}^{2} \pi \mathbb{X} 4 \mathbb{X}$ returns 113.097 for the surface of the sphere.
Actually, there was no necessity calculating this last surface here - why?

## Here is a little winter sports problem of our time:

## Example:

Chuck Carver swings down a ski run with moderate $30 \mathrm{~km} / \mathrm{h}$. The curvature of his skis allows for turns with 12 m radius. He claims carving this way without any sliding on a short flat part of the run. If true then how many $g$ he had to withstand there? Can we believe his story?

Solution (with FIX 1):
Centrifugal acceleration is $a_{c}=2 \pi \mathrm{v}^{2} / \mathrm{r}$. Thus, the total acceleration acting along Chuck's body axis is $a_{T}=\sqrt{g^{2}+a_{c}^{2}}$. This means in multiples of g $a_{T} / g=\sqrt{1+\left(a_{c} / g\right)^{2}}$.

30 ENTERT
3.6 (1)

EXP $x^{2}$
12 (1) $2 \pi x$
(CONST) $g_{\oplus}$ (1)
$\sqrt{ }\left(1+x^{2}\right)$

## 30.0

8.3 Chuck's speed in m/s 69.4 36.4

This might be possible to stand shortly for a young sportsman; but the snow under him cannot bear the corresponding forces - it will break, so Chuck will inevitably slide in a greater radius meaning less acceleration.

Another example from an early brochure about the HP-35:


The solid angle a circular detector covers of a point source is given by the formula:

$$
\Omega=2 \pi\left[1-\sqrt{\frac{1}{1+(r / l)^{2}}}\right]
$$

How big is it for $\boldsymbol{r}=2.5 \mathrm{~cm}$ and $\boldsymbol{I}=9.3 \mathrm{~cm}$ ?

## Solution:

### 2.5 ENTER $9.3 \square$ EXP $\sqrt{ }\left(1+x^{2}\right)$ (1/x $+1 \pm 2 \times x$

 returns $\Omega=0.2154$. You can solve for annular detectors in analogy.A technical application found in an HP manual of 1988:
When an incompressible fluid with negligible viscosity flows steadily along a pipe, Bernoulli's equation holds at any point $i$ along the pipe. It's given by

$$
p_{i}+\frac{1}{2} \rho v_{i}^{2}+\rho g H_{i}=K
$$

where $\boldsymbol{p}$ is the pressure, $\boldsymbol{\rho}$ is the fluid density, $\boldsymbol{v}$ is the fluid velocity, $\boldsymbol{g}$ is the acceleration due to gravity, $\boldsymbol{H}$ is the height of the fluid above some arbitrary reference point, and $\boldsymbol{K}$ is a constant. ...

The continuity equation says that the mass of fluid in a pipe is conserved (none enters or exits the pipe). The continuity equation for an incompressible fluid with steady flow is

$$
A_{1} v_{1}=A_{2} v_{2}
$$

where $\boldsymbol{A}$ is the pipe's cross sectional area and $\boldsymbol{v}$ is the fluid's velocity at points 1 and 2, respectively. Informally, the equation states that the flow rate in volume per time is constant at any point in the pipe.

## Example:

A Venturi tube shall be used to measure speed of fluid flow.
The U-shaped tube in the figure below is called a manometer and is used to measure pressure differences. The manometer equation for pressure
difference is $p_{2}-p_{1}=-\rho_{m} g h$, where $\boldsymbol{p}$ is the pressure at points 1 and 2 , $\rho_{m}$ is the density of fluid in the manometer, $\boldsymbol{g}$ is the acceleration due to gravity, and $h$ is the difference in fluid levels in the manometer. When the continuity
 equation, Bernoulli's equation, and the manometer equation are combined, the speed of fluid flow at point 1 in the figure is found to be

$$
v_{1}=A_{2} \sqrt{\frac{2\left(\rho_{m}-\rho\right) g h}{\rho\left(A_{1}^{2}-A_{2}^{2}\right)}}
$$

If the manometer is filled with mercury ( $\rho_{m}=13595 \mathrm{~kg} / \mathrm{m}^{3}$ ) and the fluid in the pipe is water ( $\rho=1000 \mathrm{~kg} / \mathrm{m}^{3}$ ), find the velocity of fluid flow at point 1 . Assume that the cross sectional area of the pipe at point 1 is $0.073 \mathrm{~m}^{2}$ and at point 2 is $0.0507 \mathrm{~m}^{2}$. The difference in mercury levels in the manometer is 9.0 cm . Follow the keystrokes below to find $\boldsymbol{v}_{1}$.
(DISP FIX 2
13595 ENTERT E 3 -
$2 \times$ CONST $g_{\oplus} \times .09 \times$
.073 EXP $x^{2} \quad .0507$ STOK $x^{2}-E 3 x$
(1) $\sqrt{x}$ RCL $X$
since the least precise input carries 2 digits.
12595.00
22232.66 0.51 2.76
4.55

The fluid velocity at point 1 is $4.6 \mathrm{~m} / \mathrm{sec} .{ }^{92}$

An high flying problem excavated in an earlier calculator manual:

## Example:

Finding himself floating dangerously close to the jagged peaks of the Canadian Rockies, intrepid balloonist Chauncy Donn frantically cranks open the helium valve on his spherical balloon. Gas from the helium tank increases the balloon's radius from 7.5 meters to 8.25 meters. Donn clears the mountain tops safely. How much did the volume of the balloon increase? ${ }^{93}$

[^40]
## Solution:

Since the volume of a sphere is $V=\frac{4}{3} \pi r^{3}$, the difference of two such volumes is $\Delta V=\frac{4}{3} \pi\left(r_{2}^{3}-r_{1}^{3}\right)$. One decimal shall do.

| 8.25 EXP $x^{3}$ | 561.5 |
| :--- | :--- |
| $7.5 x^{3}-$ | 139.6 |
| $\pi(\pi)$ | 438.7 |

4 区 3 returns
$584.9 \mathrm{~m}^{3}$ for the volume increase.

## Real Numbers: Percent Change

$\Delta \%$ calculates the percentage of change from $\boldsymbol{y}$ to $\boldsymbol{x}$.
$\triangle \%$ FIN
+/-
Example (continued from previous chapter):
This is a volume increase of how many percent?
Solution:
$7.5 \mathrm{x}^{3}$
$8.25 x^{3}$
$\Delta \% \quad$ returns
421.9
561.5
33.1 \% increase of the balloon volume.

Another example, more down-to-earth:
How about designing an almost optimum bicycle gearing for hilly areas? Feel free to choose sprockets and gear clusters to your liking.

## Solution:

As long as drag may be neglected, an optimum gearing will show equal velocity ratios between subsequent gears (or uniform percental increase of distances per crank revolution). There are several ways you can reach this, depending on the number of sprockets chosen at front and rear.
One inexpensive way is taking three front sprockets of 48,36 , and 24 teeth and a standard 7 -gear cluster featuring $13,15,17,20,23,26$, and 30 teeth. This will result in the following distances travelled per crank revolution (d/rc in meter) for a 26 " MTB:

| Gear | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Front | 24 |  |  |  |  |  |  |  |  |  |  |  |
| Rear | 30 | 26 | 23 | 20 | 26 | 23 | 20 | 23 | 20 | 17 | 15 | 13 |
| $d / r_{c}$ | 1.660 | 1.915 | 2.165 | 2.490 | 2.873 | 3.247 | 3.734 | 4.330 | 4.979 | 5.858 | 6.639 | 7.660 |
| $\Delta \%$ | 15.4 | 13.0 | 15.0 | 15.4 | 13.0 | 15.0 | 16.0 | 15.0 | 17.7 | 13.3 | 15.4 |  |

Assuming you pedal steadily with a frequency of 60 rpm , such a bicycle gearing will allow for conveniently riding with velocities between 6 and $28 \mathrm{~km} / \mathrm{h}$ (or up to $42 \mathrm{~km} / \mathrm{h}$ for 90 rpm ).
Using also three functions provided in STAT ( $(\mathbf{\Sigma}+],[\overline{\mathbf{x}}]$, and $[\mathbf{s}]$, explained on pp. 105ff), you can determine a mean speed difference per gear step of $(14.9 \pm 1,4) \%$ here. This gearing turns out being quite uniform and convenient for town and country. ${ }^{94}$ Feel free to try and check other configurations - you can obtain up to 13-gear clusters nowadays (in 2023).

## Real Numbers: Logarithms and Powers (a.k.a. Antilogs)

Your WP43 features four logarithmic functions: two on its keyboard and two more in EXP (cf. p. 28):
[ $\ln$ 〕 computes the natural logarithm of $\boldsymbol{x}$, i.e. the logarithm of $\boldsymbol{x}$ to base of Euler's $\mathbf{e}$. ( $\ln$ ) inverts $\left[\mathbf{e}^{\mathbf{x}}\right.$ ].

lg ) returns the decadic (a.k.a. common) logarithm, i.e. the logarithm

[^41]of $\boldsymbol{x}$ to base 10. ${ }^{95}(\lg )$ inverts $\left(10^{*}\right)$.
EXP $\mathrm{lb} \times$ calculates the binary logarithm, i.e. the logarithm of $x$ to base 2. ( $\mathbf{l b} \mathbf{x}$ ) inverts [ $\mathbf{2}^{\mathbf{x}}$ ).

EXP $\log _{x} y$ is the most general of these logarithmic functions, returning the logarithm of $\boldsymbol{y}$ to base $\boldsymbol{x}$. $\left(\log _{x} y\right)$ inverts $\left[y^{x}\right]$.

The operating manual of the world's very first electronic pocket calculator featuring transcendental functions, the HP-35 (cf. p. 56), presented just a single example for this then new class of pocket-able functions:

Although logarithms were originally used to speed multiplication and division, they have particular significance in scientific and engineering problems. There is, for example, a logarithmic relationship between altitude and barometric pressure. Suppose you wish to use an ordinary barometer as an altimeter. After measuring the sea level pressure (30 inches of mercury) you climb until the barometer indicates 9.4 inches of mercury. How high are you? Although the exact relationship of pressure and altitude is a function of many factors, a reasonable approximation is given by: ${ }^{96}$

$$
\frac{\text { altitude }}{[\text { feet }]}=25000 \times \ln \left(\frac{30}{\text { pressure } /[\text { inches of } \mathrm{Hg}]}\right)
$$

## Solution:

(DISP FIX 00 should suffice here.
30 ENTERT 9.4 (1)
(In)
25 E 3 returns 29 012. .
[We suspect that you may be on Mt. Everest (29 028 feet).]

Note the concise and factual style of this example. The HP-35 was a calculator made by engineers for engineers, and its manual was alike. Above example was reprinted in the HP-45 OH. Thereafter, it underwent slight modifications:

[^42]${ }^{96}$ Emphases in these quoted examples were added by me.


Example (from the HP-21 OH):
Having lost most of his equipment in a blinding snowstorm, ace explorer Buford Eugobanks is using an ordinary barometer as an altimeter. After measuring the sea level pressure ( 30 inches of mercury) he climbs until the barometer indicates 9.4 inches of mercury. Although the exact relationship of pressure and altitude is a function of many factors, Eugobanks knows that an approximation is given by the formula ...

This problem stayed this way in subsequent calculator manuals, though the explorers changed. A picture of the scenery was added in 1976, and not every snowstorm was worth mentioning anymore. Then, however, a switch of units reached the Himalayas - and also the weather and the methods changed:


## Example (from Solving Problems with Your HP Calculator of 1978):

With most of his equipment lost in an avalanche, mountaineer Wallace Quagmire must use an ordinary barometer as an altimeter. Knowing the pressure at sea level is 760 mm of mercury, Quagmire continues his ascent until the barometer indicates 238 mm of mercury. Although the exact relationship of pressure and altitude is a function of many factors, Quagmire knows that an approximation is given by the formula:

$$
\frac{\text { altitude }}{[\mathrm{m}]}=7620 \times \ln \left(\frac{760}{p /[m m \text { of } \mathrm{Hg}]}\right)
$$

Where is Wallace Quagmire?

## Solution:

## 760 ENTERT 238 (1)

## (In) 7620 区 returns 8847.

Quagmire appears to be near the summit of Mt. Everest (8848 m).
And it seems neither he nor his barometer returned from this expedition since this example did neither show up in the HP-41C OHPG nor later anymore.

For standard SI units, the altitude approximation formula reads:

$$
\frac{\text { altitude }}{[\mathrm{m}]}=7620 \times \ln \left(\frac{1013}{p /[\mathrm{mbar}]}\right)=7620 \times \ln \left(\frac{101300}{p /[\mathrm{Pa}]}\right)
$$

Beyond the barometric scale, there are more logarithmic scales used in science and engineering, e.g.

- in astronomy for assessing the brightness of stars or
- in chemistry for the power of acids $(\mathrm{pH})$; most popular may be
- the decibel (dB) in acoustics and electronics (see $\underline{U \rightarrow}$ on pp. 292f) and
- the so-called upwardly unlimited Richter scale for magnitudes of earthquakes. ${ }^{98}$



## Example:

One of the strongest earthquakes observed recently was the one causing the devastating tsunami in the Indian Ocean near Indonesia in December 2004. It had a magnitude of 9.1. Another one near Japan in March 2011 - with a magnitude of 9.0 - led to another tsunami and the 'Fukushima nuclear accident'. Compare them with the 'great San
 Francisco earthquake' of 1906 with a magnitude of 7.9.

Solution (with FIX 0):
The formula for comparing the energies released in two different earthquakes (with their magnitudes known) reads

$$
E_{2} /_{E_{1}}=10^{1.5\left(M_{2}-M_{1}\right)}
$$

Again, no decimals are needed here - we can

[^43]continue with the display settings as they are:
9.1 ENTERT $7.9 \quad 1.5 \times\left(10^{x}\right)$ returns
63. and
9 ENTERT $7.9-1.5 \times\left(10^{x}\right)$ returns 45. .

So the energy released in said Japanese earthquake in 2011 was 45 times greater than the so-called 'great San Francisco earthquake'. And said earthquake in the Indian Ocean was even 63 times more intense. ${ }^{99}$

Even small numeric differences will gain significance when raised to powers. Human brains are not well equipped for such operations, so we recommend taking good care (and your WP43) in such cases:

## Example:

What difference in magnitude will cause double destruction?

## Solution:

Rewriting the formula of previous example results in

$$
\Delta M=\frac{2}{3} \lg \left(E_{2} / E_{1}\right)
$$

Thus, for double destruction we need a magnitude difference of

$$
\text { DSP } 01
$$

2 (Ig) $2 \times 3$ equalling 0.2 only.

But there are also friendlier applications of logarithms, e.g. in electronics:

## Example:

How many bits are required if the unsigned integer $3.75 \times 10^{9}$ shall be the maximum to be handled by a microprocessor?

## Solution:

3.75 E 9 EXP lb $\times$ returns

If we had a tri-state logic, however,
RCL (L) $3 \log _{x} y$ returns

[^44]Using the inverse of the general logarithm $\left(\log _{x} y\right]$, i.e. $y^{x}$, your WP43 allows for raising any positive real number to an arbitrary real power, as well as any negative real number to an arbitrary integer power, all returning real results. Compare e.g. the Mach number formula on p. 48.

Let's return to Fukushima for a really down-to-earth example for application of logarithmic functions:

Locations in a distance of 30 km to the nuclear (fission) power plant Fukushima Daichi being devastated by the tsunami in March 2011 showed radioactivity in the soil of some $1 \ldots 3 \mathrm{MBq} / \mathrm{m}^{2}$ corresponding to an annual radiation dose of 4 mSv in 2013. Assume this was mainly caused by ${ }^{137} \mathrm{Cs}$ then; this radioactive caesium isotope has a half-life of 30.2 years. ${ }^{100}$
$0.125 \mu \mathrm{~Sv} / \mathrm{h}$ or higher


Original map found at http://www.odonata.jp/ico2012/radioactivity/radiation_contour large map.gif

[^45]To the best of our knowledge today (2023), an unborn child must not receive a dose of more than 1 mSv before birth. So when will it be reasonably safe to let the evacuated inhabitants of the villages in that
 area return to their homes finally?

Solution (with FIX 0):
Assuming there will be no further nuclear accident there, the caesium set free in 2011 will stubbornly decay following the laws of physics. Having had a radioactivity $a_{0}$ at time zero, the activity $a$ at a later time $t$ will be

$$
a=a_{0} \times 2^{-\left(t / T_{1 / 2}\right)} \Rightarrow t=T_{1 / 2} \times l b\left(\frac{a_{0}}{a}\right)
$$

1 mSv in 9 months corresponds to an annual dose of $4 / 3 \mathrm{mSv}$. The 2013 annual dose of 4 divided by $4 / 3$ equals 3 , and

3 EXP $\mathrm{lb} \times 30.2 \times$ returns 48. years.
So you can recommend reproductive people shall rather not stay pregnant in that area earlier than 2061. Senior inhabitants may return far sooner. ${ }^{101}$


#### Abstract

With a probability of $94.6 \%,{ }^{137} \mathrm{Cs}$ decays emitting an electron with kinetic energy of up to 512 keV plus a $\gamma$-ray of 662 keV . These facts are for your information only, they do not affect the calculation here. ${ }^{101}$ Note that different limits are considered 'reasonably safe' for the public by authorities of different nations. By nature, all such limits are arbitrary to some extent since we talk about probabilities here, and there are no step functions in probability but smooth transitions (see the chapter after next). Furthermore, a large fraction of world-wide knowledge about radiation damage in human bodies in the long range is still based on extrapolation of experiences collected since 1945 following two large-scale events in Japan (and 67 more near Bikini until 1958). Another experiment well known by the public was started in the USSR in 1986 - Belarus and the Ukraine have to bear the consequences until today (and for centuries to come). Mankind knows of the physics of radioactivity for some 120 years only so far, that's not long (just 4 half-lifes of ${ }^{137} \mathrm{Cs}$ ).


There are further risks linked to agriculture in the area around Fukushima - they are beyond the scope of this simple sample calculation though.
Also note this example covers a worst case scenario. Actually, radioactivity is washed to deeper layers of soil with rain and time, reducing the activity seen at the surface. And there are mitigation efforts in the area (many $\mathrm{km}^{2}$ ): at some places the contamination was washed off houses and trees, and the top layers of contaminated soil were removed, storing them in big black plastic bags 'elsewhere'. 2.3 million $\mathrm{m}^{3}$ of soil are deposited there already, 12 million more are expected by the Japanese authorities - an area of $1.6 \mathrm{~km}^{2}$ is provided for 'interim storage'.
Cost of disposal is going to be $1.9 \times 10^{12} ¥$ as estimated by the Japanese administration in 2019 (cf. Frankfurter Allgemeine Zeitung of 2019-03-10 - this article is behind a

The formula above is a nice example of a mathematically simple law of physics linking science and society quite closely. ${ }^{102}$


#### Abstract

Similar considerations apply to waste of nuclear fission power plants while it is true they do not emit any carbon dioxide, there are many kilotons of radioactive material produced decaying with half-lifes exceeding thousand years (about 400 kt present already plus some $12 \mathrm{kt} / \mathrm{a}$ ). This means you have to 'put them away' safely for really long times - a task kept under wraps for decades but not solved by waiting so far. ${ }^{103}$


#### Abstract

paywall now). Furthermore, 1.2 million $\mathrm{m}^{3}$ of tritium-contaminated water are stored separately (equivalent to a cube of about 100 m edge length). These flow slowly into the Pacific since 2022 (cf. Frankfurter Allgemeine Zeitung of 2021-04-13). Tritium is radioactive featuring a half-life of 12.3 years and emitting short range radiation ( $\alpha$ particles). Thus, you shall not drink or inhale it. Check locally applicable radiation limits. Success of the mitigation efforts mentioned may reduce the waiting time calculated above; failure will not extend it at least. Today is too early for a definitive assessment we still know too little about long term effects of radioactivity in biology. ${ }^{102}$ In consequence of this natural law, radioactivity (once set free) decreases slowly: even after 10 half-lifes you will still see $1 / 1024$ of the original activity (take 137 grams of ${ }^{137} \mathrm{Cs}$, for example, being $6.022 \times 10^{23}$ atoms; after 302 years, $5.881 \times 10^{20}$ atoms are present still). Although radioactivity decreases continuously, it will actually never vanish - it will just become less relevant with time.


${ }^{103}$ Surprise! Mankind has absolutely no experience with locking something away reliably for many thousand years (look at the pyramids of Gizeh, for instance). Note the final repository site and the material there must also be tagged properly (KEEP OFF !!) in a way staying readable and comprehensible for all that time - no experience either.
Sad real-life example: A huge concrete coffin holds almost 90 million liters (i.e. $90000 \mathrm{~m}^{3}$ or a cube of 45 m edge length) of US nuclear waste on the Marshall Islands (remember Bikini!). Now sea-level rise (caused by anthropogenic global warming which very few were aware of 60 years ago) is eating away at the dome, and the USA are not interested in helping the tiny Pacific Ocean republic to do anything about it (see the Los Angeles Times of 2019-11-10). So much about taking responsibility. And 60 are a lot less than 5000 years, so you can estimate the surprises in future millennia, if there will be any humans left noticing them.
You might meet some people talking about 'transmuting' the entire long-living radioactive waste by converting it to isotopes with significantly shorter half-lifes by some nuclear reactions. Ask them if this is physically possible for all that material, and how much energy is needed for that transmuting process. It might outweigh the energy 'produced' by nuclear power plants before.
As a matter of fact, the companies who made profits with those nuclear power plants for decades are very reluctant definitely solving the waste problem they created so far. They would not, certainly, if they expected making some money from doing that. And note no

Another industrial example, dealing with mechanical transducers:

To measure loads, you can use a bending steel ring supported at its outer radius $\boldsymbol{d}$ and loaded at its inner radius $\boldsymbol{c}$ with a force $\boldsymbol{F}$ as sketched (deformation is widely overstated here). The stress on its bottom face is


$$
\sigma=\frac{3}{\pi} F \frac{d-c}{h^{2} a \ln (d / c)} \text { with } a=\frac{d+c}{2} .
$$

There, a reasonable stress level for measuring is $200 \mathrm{~N} / \mathrm{mm}^{2}$ under full load. Assume you want to build a compact transducer for 5000 kg with an outer radius $\boldsymbol{d}=25 \mathrm{~mm}$ and choose $\boldsymbol{c}=15 \mathrm{~mm}$, how big shall be $\boldsymbol{h}$ ? ${ }^{104}$

Solution:

$$
h=\sqrt{\frac{3}{\pi} \times \frac{F}{\sigma} \times \frac{d-c}{a \ln (d / c)}} .
$$

You can solve it like this:
DISP) FIX (0) 1
25 ENTERT 15 (In
25 (ENTERT $15+2(1) \quad x \mid 1 / x$
25 ENTERT $15-\mathbb{x}$
5 E 3 (CONST $g_{\oplus} \times 200$ (1)
3 ( $\pi$ (1) $\sqrt{x}$
returns
15.1 mm for $\boldsymbol{h}$.
insurance company on this planet offers any police covering nuclear power plants. Now you have all the information required to assess nuclear fission power plants.
Nuclear fusion power plants differ. They are predicted being operational in 40 years (but the same was said 50 years ago already). As far as I know today, they will not produce any long-lived radioactive material in operation. We will be most happy when this turns out being true.
104 The formula above was derived by the author developing load cells for electronic industrial scales in 1987.

## Real Numbers: Hyperbolic Functions

Hyperbolic functions tell us something about free hanging ropes, cables, chains, and the like. Your WP43 provides three hyperbolic functions and their inverses in the -shifted rows of EXP and TRI:
sinh Hyperbolic sine.
cosh Hyperbolic cosine.
tanh Hyperbolic tangent.
arsinh Inverse hyperbolic sine.
arcosh Inverse hyperbolic cosine.
artanh Inverse hyperbolic tangent.

We found the following example for applying these functions in the HP-32 OH though we modified it a bit: 105


In Upper Lagunia, a tram ${ }^{106}$ carries tourists between two peaks in the Baruvian Alps that are the same height and 437 meters apart. How long does it take the tram to travel from one peak to the other if it moves along its cable at 135 meters per minute? Before the tram latches onto the cable, the angle from the horizontal to the cable at its point of attachment is found to be $43^{\circ}$.

## Solution:

The travel time is given by $t=(d / v) \times \tan \alpha / \operatorname{arsinh}(\tan \alpha)$
Let's set DISP FIX (2) since this will suffice.

Then 43 TRI tan
ENTERT
arsinh (1)
437 区
135 (1)
is an intermediate result we need twice. duplicates it on stack for numerator and denominator. 489.30 m is the length of the cable.
3.62 , i.e. a bit more than $31 / 2$ minutes.

[^46]${ }^{106} T N$ : British readers might frown here at least.

## Real Numbers: Probabilities - Factorials, Combinations, Permutations, and Distributions

Besides the keyboard commands $\Delta \%$ and x!, you find a lot of operations for probability and statistics in your WP43, going far beyond the Gaussian distribution. It contains all the preprogrammed functions implemented in WP 34S and more presumably the maximum set available
 in a pocket calculator world-wide.
These operations are stored in the adjacent menus PROB and STAT.
PROB includes the functions for combinations and permutations.

|  | NBin: | Geom: | Hyper: | Binom: | Poiss: |
| :---: | :---: | :---: | :---: | :---: | :---: |
| LgNrm: | Cauch: |  | Expon: | Logis: | Weibl: |
| Norml: | $\mathrm{t}:$ | $\mathrm{C}_{y \mathrm{x}}$ | P $_{\mathrm{yx}}$ | F | $\chi^{2}$ |

Example (from the HP-32 OH):
Willie's Widget Works wants to take photographs of its product line for advertising. How many different ways can the photographer arrange their eight widget models?

## Solution:

The total number of possible arrangements possible is given by the factorial $8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1=8$ !

$8 \times 4$ ! returns 40320 for this number.

Example (continued):
The photographer looks through his viewfinder (in 1978) and decides that he can show only five widgets if his camera is to capture the intricate details of the widgets ... How many different sets of five widgets can he select from the eight?

Solution:
The number of sets equals the number of possible combinations (i.e. the number of possible different sets of $\boldsymbol{y}$ different objects taken in quantities of $\boldsymbol{x}$ objects at a time; no object appears more than once in a set, and different orders of the same $x$ objects are not counted separately here):

## 8 ENTERT 5 PROB $\mathbf{C}_{\mathbf{y x}}$ returns 56 for this number.

Example (continued):
Again, there are different arrangements feasible. How many pictures of different widget arrangements are possible within these limits?


## Solution:

The number of possible arrangements is according to the statement on previous page. Thus,
5 x! returns 120 for that number. And ...

## $\mathbf{x}$ returns 6720 for the number of significantly different pictures.

This is the number of possible permutations of 5 items out of 8 (i.e. the number of possible different arrangements of $\boldsymbol{y}$ different objects taken in quantities of $\boldsymbol{x}$ objects at a time; no object appears more than once in an arrangement, and different orders of the same $\boldsymbol{x}$ objects are counted separately here). It can be obtained in one step by keying in

$$
8 \text { ENTERT } 5 \text { Pyx returning } 6720.107
$$

Furthermore, PROB comprises nine continuous and five discrete distributions ${ }^{108}$ for calculating probabilities, confidence intervals, etc. These functions share a few features:

- Discrete distributions (like Poisson, binomial, negative binomial, geometric, and hypergeometric) are confined to integers. Whenever

[^47]BEWARE: This is a very rudimentary sketch of this topic only - turn to a good textbook to learn dealing with statistics properly.
TNG: PMF and PDF = Wahrscheinlichkeitsdichte, CDF = Verteilungsfunktion bzw. Wahrscheinlichkeitsverteilung.
your WP43 sums up a probability mass function (PMF) $p(n)$ to get a cumulated distribution function (CDF) $P(m)$, it starts at $n=0$. Thus,

$$
P(m)=\sum_{n=0}^{m} p(n)
$$

- Continuous distributions (like Cauchy, chi-square, exponential, Fisher's F, log-normal, logistic, normal, Student's $t$, and Weibull) operate on reals. Whenever your WP43 integrates a function, it starts at left end of the integration interval. Thus, integrating a continuous probability density function (PDF) $f(x)$ to get a CDF works as

$$
P(x)=\int_{-\infty}^{x} f(\xi) d \xi
$$

- Many frequently used continuous PDFs look more or less like the ones plotted in the upper diagram below. The lower diagram shows their corresponding CDFs, using the same scale and colors.


Typically, any CDF will start at 0 with a slope of almost zero, become steeper then, and run out at 1 with its slope returning to zero. This holds even if the respective PDF do not look as nice and symmetric as the normal distributions plotted here.

Thus, you will get the most precise results for the CDF on its left side using $P$. But on its right side, where $P$ approaches 1 , the error probability $Q=1-P$ will be more precise. Thus, also the right sided $Q$ is computed in your WP43 for each distribution, independently of $P$. The definitions are:

$$
\begin{array}{ll}
\text { - for discrete distributions: } & Q(m)=\sum_{n=m} p(n) \\
\text { - for continuous distributions: } & Q(x)=\int_{x}^{\infty} f(\xi) d \xi
\end{array}
$$

- With an arbitrary CDF, e.g. NORML $_{\wedge}$ (returning $P$ ), you will find the names $\mathrm{NORML}_{\mathbf{\Lambda}}$ used for the function returning $Q, \mathrm{NORML}^{-1}$ for the
 inverse of the CDF (the socalled quantile function), and NORML $_{P}$ for its PDF on your WP43. This naming convention also applies to the binomial, Cauchy (a.k.a. Lorentz or Breit-Wigner), exponential, Fisher's F, geometric, hypergeometric, logistic, log-normal, negative binomial, Poisson, Student's $\boldsymbol{t}$, and Weibull distribution. Just the chisquare distribution is denoted differently following mathematical tradition. See PROB on p. 118 or the ReM.

Find application examples of distributions in next chapters.

## Real Numbers: 1D Statistics - Sampling Data, Calculating Means, Standard Deviations, and Confidence Limits

A wealth of commands for sample and population statistics is in STAT, applicable in one or two dimensions. After initial clearing of all statistical records by [CLI], use $[\boldsymbol{\Sigma}+$ ] to accumulate your experimental data. Weighted data require the data value in $\mathbf{X}$ and its weight in $\mathbf{Y}$, pairs of data or coordinates of data points shall be entered in $\mathbf{X}$ and $\mathbf{Y}$. ( $\boldsymbol{\Sigma}-$ ] is provided for easy data correction - it will remove the last point entered.

Data analysis functions are found in STAT as well: e.g. the arithmetic mean $\overline{\mathbf{x}}$, sample and population standard deviations $s$ and $\sigma$, and the standard error $\mathbf{s}_{\mathbf{m}}$ (a.k.a. standard deviation of the mean).

## Example (from the HP-32 OH):

Norman Numbercruncher, a rising young math professor at Mammoth University, has developed a new test for measuring the mathematical abilities of college freshmen. To evaluate its effectiveness, he administers the test to the 746 students in Calculus I. Exhausted after grading the tests, Numbercruncher decides to randomly select 8 of the 746 tests and estimate the standard deviation of all the scores from the sample of 8 . The scores on the tests selected were 79,94 , $68,86,82,78,83$, and 89 . What standard deviation (SD) does Numbercruncher calculate?

## Solution:

STAI CL $\Sigma$

| CLE | $\bar{x}_{G}$ | $\varepsilon$ | $\varepsilon_{\mathrm{p}}$ | $\varepsilon_{\mathrm{m}}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $\Sigma$ - | $\bar{x}_{w}$ | $S_{\text {w }}$ | $\sigma_{w}$ | $\mathbf{s}_{\text {mw }}$ |  |
| $\Sigma+$ | $\bar{x}$ | 5 | $\sigma$ | $\mathrm{S}_{\mathrm{m}}$ | SUM |

0 ENTERT $79 \mathbf{\Sigma +}$ returns 001 data point
$\boldsymbol{\Sigma}+$ takes $\boldsymbol{x}$, stores it in STATS, displays feedback incl. TI (cf. p. 71), and disables ASL (cf. f. 29). - Continue, overwriting $x$ with each new entry:


DISP FIX 3
$\mathbf{s}$ returns $s_{\mathbf{x}}=7.836$ the $S D$ estimated for the 746 students based on a sample of $8 .{ }^{109}$

When your data constitutes not just a sample of a population but rather all of the population, the $S D$ of the data is the true population $S D$ (denoted $\sigma$ )... The difference between the values of $s$ and $\sigma$ is small, and for most applications can be ignored. Nevertheless, if you want to calculate the exact value of the population $S D$ for an entire population,

[^48]you can easily do so with just one keystroke on your WP43.

## Example (continued):

Suppose the data from the previous example represented all the final exam scores from Numbercruncher's seminar on transcendental functions. Since this is the first time Numbercruncher has given this seminar, he wants to calculate the SD of the test scores to determine how good his exam was. Numbercruncher takes his calculator in hand, enters the data, then proceeds as follows
o returns $\sigma_{x}=7.330$ for the SD for all scores on final exam. ${ }^{109}$

## This is a no-brainer with your WP43. Thus, let's try something completely different:

## Example:

Archibald is champion of the Golden Bow, his archers club. In his standard exercise, aiming at a target disk of 1.5 m diameter at a distance of 50 m , his arrows scatter symmetrically around the center of the target showing quite a small variance. Actually, Archibald's statistics tells his arrows scatter with a long-term SD of 1 foot at that distance. Assuming his shots are distributed
 normally around the center of the disk, how often must he walk further than 50 m to collect an arrow? ${ }^{110}$

Solution (with FIX 3):

0 STO (1)

$0.305=1$ foot in meters, Archibald's SD. store this SD for later.
${ }^{110}$ Many of our customers live in a country where long range weapons play a significantly greater role than in (most) civilized societies, hence this explanatory example.
In that country, next civil war will certainly not founder on lack of personal firearms. Foreigners travelling there, beware and also take good care of your kids! Even in your wildest dreams, you can't imagine all who may carry firearms there and where. Note many, many innocent people have been shot in that area in three centuries just since they were in the wrong place at the wrong time, had a wrong skin color, wrong customs and traditions, or just a wrong way of life in the opinion of the men and women behind the guns - entire peoples were extinguished but the term genocide wasn't coined yet.


## Example (continued):

One of his buddies and competitors, Bill, also sends his arrows to the same target disk with his hits scattering symmetrically around the center of said disk, too. He, however, has to pick up about one out of 15 arrows in the green on average. What is his SD in the target plane?

## Solution:



There are applications of this methodology in industry, where the scattering (a.k.a. variation, variance) of a production process is compared with its tolerance limits. Resulting from such comparisons, so-called capability indices are computed, directly linked to the amount of scrap to be expected in the process investigated. Please consult applicable literature and standards - look for process capability.
On the other hand, we may continue with our example as is, guiding to more advanced statistics:

Bill quietly practiced in a Zen cloister during his summer vacation. Returning, he went to the Golden Bow immediately on next weekend and sent 50 arrows to his club's standard disk. Only two missed, with one of them scratching the very edge of the disk. Cheers! But is this just a lucky chance success (within
the usual scattering of results to be expected) or probably related to his training efforts?

## Solution:

Calculate Bill's new SD:

| 1.5 ENTERT 50 (1) | 0.030 |
| :---: | :---: |
| 2 (1) | $0.015=1.5 \%$ misses on either side. |
| Norml ${ }^{\mathbf{- 1}}$ | -2.170 , the corresponding lower limit of the standardized normal distribution. |

+1/ . 75 x $\quad 0.346 \mathrm{~m}=$ Bill's new $S D$.
Now, is this significantly better than his old $S D$ ( $\mathrm{s}_{0}$ )? Statisticians have found it is better (with a confidence level of $95 \%$ ) if it is lower than the $95 \%$ confidence limit of $\mathrm{s}_{0}$. Assume $\mathrm{s}_{0}$ was based on 60 shots, then the formula for its single-sided lower 95\% confidence limit reads:

$$
\sigma_{L}=s_{o} \times \sqrt{\frac{59}{\left(\chi_{59 ; 0.95}^{2}\right)^{-1}}}
$$

The expression in the denominator is the inverse chi-square for $95 \%$ probability and 59 degrees of freedom (dof). Calculate inside out as usual:


Looks like Bill's training made a difference! ... Well, ... with 95\% confidence. If we had required $99 \%$ instead, the lower confidence limit had been 0.337 m (you can verify this easily now), and then Bill's new weekend result had been an insufficient indicator for a significant improvement. 111
${ }^{111}$ Applying statistics may cause you have more doubts than without - but such is life: Doubts increase with knowledge; only brainless people have never any doubts at all and therefore may feel GREAT most easily.
Generally, standard confidence limits and levels (also those defined for indicating significant differences) may depend on the country, industry, or science you are working in. Note the term significant is well defined in statistics - this definition may deviate from common language. Also, significance isn't everything, standard errors may be a better measure if applicable. Be sure to check the pertinent valid standards before blindly copying exemplary calculations presented in this OM.

## Real Numbers: 2D Statistics - Curve Fitting and Forecasting

STAT contains also functions for curve fitting, featuring ten different regression models (linear, exponential, logarithmic, power, root, hyperbolic, and more - see the ReM), their parameters, the forecasting functions $[\hat{\boldsymbol{x}}]$ and $[\hat{\boldsymbol{y}}]$, and the coefficient of correlation $[\mathbf{r}]$. The fit model applied will be displayed heading numeric output after any command related to fitting (like CORR, L.R., $\hat{\mathrm{x}}$, and $\hat{\text { y }}$ ). And after (L.R.), even the generic formula of the regression model applied will be shown (see examples below).

The command BESTF tells your WP43 to select the regression model fitting your data 'best' (i.e. resulting in the biggest absolute coefficient of correlation, close to 1). Then an elevated asterisk (*) will trail the name of the model chosen this way automatically (please check the $I O I$ for the nine models available and the parameter of BESTF). Like with all autofunctionality, you should know what you are doing here.

## Example (from the HP-27 OH, continued):

If Galileo had wished to investigate quantitatively the relationship between the time ( $\boldsymbol{t}$ ) for a falling object to hit the ground and the height $(\boldsymbol{h}$ ) it has fallen, he might have released a rock ${ }^{112}$ from various levels of the Tower of Pisa (which was leaning even then) and timed its descent by counting his pulse. The following data are measurements Galileo might have made:

| $\boldsymbol{t}$ (pulses) | 2 | 2.5 | 3.5 | 4 | $4.5^{113}$ |
| ---: | :---: | :---: | :---: | :---: | :---: |
| $\boldsymbol{h}$ (Pisan feet) | 30 | 50 | 90 | 130 | 150 |

Unlike Galileo, you are equipped with a WP43 - so what can you learn from this reported experiment? Let's look what can be found:

## DISP FIX 3

## STAI CL $\Sigma$

| $C L \Sigma$ | $\bar{x}^{\mathbf{G}}$ | $\varepsilon$ | $\varepsilon_{\mathrm{p}}$ | $\varepsilon_{\mathrm{m}}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $\Sigma-$ | $\bar{x}_{\mathrm{w}}$ | $\mathbf{s}_{\mathrm{w}}$ | $\sigma_{\mathrm{w}}$ | $\mathbf{s}_{\mathrm{mw}}$ |  |
| $\Sigma+$ | $\bar{x}$ | $\mathbf{s}$ | $\sigma$ | $\mathbf{s}_{\mathrm{m}}$ | SUM |

[^49]Your next input after $\boldsymbol{\Sigma}+$ will overwrite $\boldsymbol{x}$ and shift previous $\boldsymbol{y}$ up: ${ }^{114}$

| 50 | ENTERT $2.5 \Sigma+$ | 90 ENTERT $3.5 \Sigma+$ |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 130 | ENTERT $4 \mathbf{\Sigma +}$ | 150 | ENTERT | 4.5 | $\Sigma+$ |  |
|  |  | 005 data points |  |  |  | $\begin{array}{r} 150.000 \\ 4.500 \end{array}$ |
| $\nabla$ |  | GaussF | CauchF |  | BestF |  |
|  |  | ParabF | HypF | RootF |  |  |
|  |  | LinF | ExpF | LogF | PowerF | OrthoF |


#### Abstract

BestF 448 instructs your WP43 to choose out of all 2-parameter fit models provided the one suiting these experimental data best (see the IOI for more).


$\nabla$

| ASSESS | $\bar{x}_{\text {RMS }}$ | $x_{\text {max }}$ | $x_{\text {min }}$ | HISTOG | PLOT |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\mathbf{s}(\mathrm{a})$ | $\bar{x}_{H}$ |  |  |  |  |  |
| L.R. | $r$ | $s_{x y}$ | cov | $\hat{x}$ | $\hat{y}$ |  |

L.R.

|  |  | 4.500 |
| :--- | :--- | :--- |
| Power* | $a_{1}=$ | 1.994 |
| $y=a_{0} x^{\wedge} a_{1}$ | $a_{0}=$ | 7.723 |

Your WP43 chose power regression as the model fitting these data best. Let's check the correlation coefficient:
$r$
returns
Power* $r=$
0.998

This is a very good correlation. To be precise, what are the estimated errors of the curve parameters?

114 Thus, accumulating $2 D$ data will slowly overwrite the stack - use STOS if you want to preserve it and recall it by RCLS after accumulation. ( $\Sigma+$ mechanics was created for $1 D$ data on the HP-80, HP-45, until the HP-29C - it stayed this way also for $2 D$ data. Compare the $1^{\text {st }}$ example in previous chapter.)

$$
\begin{aligned}
& s\left(a_{1}\right)= \\
& s\left(a_{0}\right)=
\end{aligned}
$$

Hence, the equation expressing the overall results of these experiments best is $h=(7.72 \pm 0.09) \times t^{1.99 \pm 0.08}=7.72(9) \times t^{1.99(8)}$ with $\boldsymbol{t}$ measured in
 pulses and $\boldsymbol{h}$ in Pisan feet (this equation shows two alternative ways for specifying the uncertainties). See the plot produced by

## ASSESS

 and Z00M showing the data points and the fit curve. Galileo could not know around 1600 yet, but we know today that$$
h=\frac{1}{2} g t^{2} .
$$

Determining the size of a Pisan foot and Galileo's heartbeat frequency during his reported arduous experimental work with rocks is left as an exercise for the reader.

In addition, we found the following example in various HP calculator
 manuals of $1976-78$. It reads typical for the thinking at that time:

Big Lyle Hephaestus, ${ }^{115}$ owner-operator of the Hephaestus Oil Company, wishes to know the slope and $y$-intercept of a least squares line for the consumption of motor fuel in the United States of America ${ }^{116}$ against time since 1945 (in 1978!). He knows the data given in the table:

| Motor fuel demand <br> (millions of barrels) | 696 | 994 | 1330 | 1512 | 1750 | 2162 | 2243 | 2382 | 2484 |
| ---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Year | 1945 | 1950 | 1955 | 1960 | 1965 | 1970 | 1971 | 1972 | 1973 |

[^50]Solution (continuing with BESTF 448):
Hephaestus could draw a plot of motor fuel demand against time. However, with his WP43, Hephaestus has only to key the data into the calculator using the [ $\Sigma+]$ key, then press (L.R.).
(DSP) 0 ( 1
(STAT CL

| 696 | ENTERT | 1945 | $\Sigma+117$ | 994 | ENTERT | 1950 | $\Sigma+$ |
| :--- | :--- | :--- | :--- | :---: | :--- | :--- | :--- |
| 1330 | ENTER | 1955 | $\Sigma+$ | 1512 | ENTERT | 1960 | $\Sigma+$ |
| 1750 | ENTERT | 1965 | $\Sigma+$ | 2162 | ENTERT | 1970 | $\Sigma+$ |
| 2243 | ENTERT | 1971 | $\Sigma+$ | 2382 | ENTER | 1972 | $\Sigma+$ |

2484 ENTERT $1973 \Sigma+$
(4) L.R.

| Linear* | $a_{1}=$ | 61.2 |
| :--- | :--- | ---: |
| $y=a_{0}+a_{1} x$ | $a_{0}=$ | -118290.6 |

Your WP43 selected linear regression as the mathematical model fitting these data best. Let's check the correlation coefficient:
$r$ returns Linear* $\mathbf{r}=1.0$

Based on this good correlation result, Hephaestus confirms the automatic choice and is even tempted to extrapolate the calculated trend of motor fuel demand to (then) future years.

## Example (continued):

If Hephaestus wishes to predict the demand for motor fuel for the years 1980 and 2000, he keys in the new $\boldsymbol{x}$ values and presses ( $\hat{\boldsymbol{y}}$. Similarly, to determine the year that the demand for motor fuel is expected to pass 3500 million barrels, Hephaestus keys in $\mathbf{3 5 0 0}$ (the new value for $\boldsymbol{y}$ ) and presses $[\hat{\mathbf{x}}]$.

| $1980 \hat{y}$ | returns | Linear* $\hat{y}=$ | 2808.6 |
| :--- | :--- | :--- | :--- |
| $2000 \hat{y}$ | returns | Linear* $^{\hat{y}}=$ | 4031.9 |

These were forecasts (i.e. extrapolations based on the fit model employed) of the demands in 1980 and 2000 at that time.

3500
$\hat{\boldsymbol{x}}$
returns

$$
\text { Linear* } \hat{x}=
$$

1991.3

[^51]

- the demand was expected to pass 3.5 billion barrels in 1992. ${ }^{118}$

Another example (from the $\mathrm{HP}-15 \mathrm{COH}$ ):
Agronomist Silas Farmer has developed a new variety of high-yield rice, and has measured the plant's yield as a function of fertilization. Use the [ $\boldsymbol{\Sigma}+$ ] function to accumulate the data below to find the values for $\Sigma x, \Sigma x^{2}, \Sigma y, \Sigma y^{2}$, and $\Sigma x y$ for nitrogen fertilizer application $(x)$ versus grain yield $(\boldsymbol{y})$.


| $\boldsymbol{x}$ | Nitrogen applied (kg/ha) | 0.0 | 20.0 | 40.0 | 60.0 | 80.0 |
| :--- | :--- | ---: | ---: | ---: | ---: | :--- |
| $\boldsymbol{y}$ | Grain yield (t/ha) | 4.63 | 5.78 | 6.61 | 7,21 | 7,78 |

## Solution:

We leave aside the data entry steps, sums, mean and SD of HP's original example here, being confident you can do them yourself using STAT now. ${ }^{119}$ Let's proceed to the linear regression Silas really is interested in:

## DSP 2

## STAT $\mathbf{L i n F}$

T L.R. returns

| Linear | $a_{1}=$ | 0.04 |
| :--- | :--- | :--- |
| $y=a_{0}+a_{1} x$ | $a_{0}=$ | 4.86 |

Let's check the correlation coefficient:
$r \quad$ returns $\quad$ Linear $r=0.99$

[^52]

Plot the data using ASSESS again. The plot looks good enough to work with the fit line in the range covered by data points, e.g. find how much grain is expected with $70 \mathrm{~kg} / \mathrm{ha}$ fertilizer: ${ }^{120}$
$70 \hat{y}$ returns $\quad 7.56 \mathrm{t} / \mathrm{ha}$.

## Real Numbers: Chi-Square Statistic - Checking Expectations

The chi-square statistic measures the goodness of fit between two sets of frequencies. ${ }^{121}$ It's used to test whether a set of observed frequencies differs from a set of expected ones sufficiently to reject the hypothesis under which the expected frequencies were obtained.

In other words, you are testing whether discrepancies between the observed frequencies $\left(\mathrm{O}_{\mathrm{i}}\right)$ and the expected frequencies $\left(\mathrm{E}_{\mathrm{i}}\right)$ are significant, or whether they may reasonably be attributed to chance. The formula generally used is

$$
\chi^{2}=\sum_{i=1}^{n} \frac{\left(O_{i}-E_{i}\right)^{2}}{E_{i}}
$$

If there is a close agreement between the observed and expected frequencies, $X^{2}$ will be small. If the agreement is poor, $X^{2}$ will be large. ${ }^{122}$ Let's demonstrate the application of such a chi-square statistic using the following example:

[^53]

A suspect dice from a Las Vegas casino is brought to an independent testing firm to determine its bias, if any. The dice is tossed 120 times and the following results obtained:

| Number | 1 | 2 | 3 | 4 | 5 | 6 |
| ---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Frequency | 25 | 17 | 15 | 23 | 24 | 16 |

Solution (with FIX 0):
Expected frequency is $120 / 6=20$ for each number here. For calculating $X^{2}$, just enter:

| 25 | ENTERT | $20-$ |  | $x^{2}$ | 25 |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 17 | ENTERT | $20-$ | $x^{2}$ | $\pm$ | 34 |
| 15 | ENTERT | $20 \square$ | $x^{2}$ | $\pm$ | 59 |
| 23 | ENTERT | $20-$ | $x^{2}$ | $\pm$ | 68 |
| 24 | ENTERT | $20 \square$ | $x^{2}$ | $\pm$ | 84 |
| 16 | ENTERT | $20 \square$ | $\mathrm{x}^{2}$ | $\pm$ | 100 |
|  | (1) |  |  |  | 5 |

Now, is this $X^{2}$ large or small? Statisticians have found it is to be considered 'small' if $X^{2}$ is less than the value of the inverse $X^{2}$ CDF for the degrees of freedom (here $\boldsymbol{n}-1=5$ ) and the significance level applicable (here 5\%). As experienced above already, also this $\chi^{2}$ function is provided in your WP43. Simply key in:

5 STO (1)
.95 PROB $\chi^{2}:\left(\chi^{2}\right)^{-1}$

5 for the degrees of freedom; 11.

Since 5 is less than $11, x^{2}$ is small enough to conclude that this dice is fair (with $95 \%$ confidence). ${ }^{123}$

[^54]Note that a confidence level of 95\% equals an error probability of 5\% and a significance level of $5 \%$. These $5 \% / 95 \%$ are typical settings for the USA, as quoted in this example. For comparison, e.g. in Germany $1 \% / 99 \%$ are standard. See also further examples below.

TNG: Confidence limit = Vertrauensbereichsgrenze, confidence level = Vertrauensniveau

## Real Numbers: Some Industrial Problems Solved

To get an idea of further real-life opportunities covered by your WP43 and of some constraints inherent to statistics, see the applications shown in this chapter. All of them are demonstrated using the traditional 4-level stack but will work with the 8-level stack as well.

## Application 1 (scrap rate, confidence limits):

Assume you own a little tool shop, produce axle pins automatically in series, and want to know the quality of the parts you produce after setting up the machine. You drew a representative sample of pins (all being nominally equal parts!) and precisely measured their real sizes using a proper instrument. How can you know your batch will be ok?

## Example:

A batch of turned pins is produced on a precision lathe. A sample of 10 pins is drawn and their diameters measured with a precision gauge: ${ }^{124}$ 12.356, 12.362, 12.360, 12.364, 12.340, 12.345, 12.342, 12.344, 12.355, and 12.353 mm . From earlier large scale investigations, you know that diameters from this production process follow a normal distribution.

Do you want to know what pin diameters you will get in larger scale batch production? Statistics cannot tell you about all of them but it will tell you where to find almost all (e.g. 99\%) of them.

Example (continued):
DISP) FIX 3
STAT CLE
0 ENTERT1 $12.356 \Sigma+$


Continue accumulating the remaining measured sample data:

| 12.362 | $\Sigma+$ | 12.360 | $\Sigma+$ | 12.364 | $\Sigma+$ | 12.340 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 12.345 | $\Sigma+$ | 12.342 | $\Sigma+$ | 12.344 | $\Sigma+$ | 12.355 |
| $\Sigma+$ |  |  |  |  |  |  |

[^55]Knowing these pins are drawn from a Gaussian process, you get the best estimates for mean and standard deviation of your batch by
$\bar{x}$

STO (1)
s

$$
\begin{array}{lr}
\bar{y}= & 0.000 \\
\bar{x}= & 12.352
\end{array}
$$

|  | 12.352 |
| :--- | ---: |
| $s_{y}=$ | 0.000 |
| $s_{x}=$ | 0.009 |

We stored $\overline{\mathrm{x}}$ and $\mathrm{s}_{\mathrm{x}}$ for the next steps already.


Thus, based on the ten pins analyzed, you may expect $0.5 \%$ of all pins with diameters less than
.005 PROB

| 0.005 |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | NBin: | Geom: | Hyper: | Binom: | Poiss: |
| LgNrm: | Cauch: |  | Expon: | Logis: | Weibl: |
| Norml: | t: | $\mathrm{C}_{\mathrm{yx}}$ | $\mathrm{P}_{\mathrm{yx}}$ | F: | $\chi^{2}$ : |

Normla


Norml $^{-1}$

[^56]and another $0.5 \%$ with diameters greater than

If you should observe significantly more than $0.5 \%$ of your pins beyond either limit, this indicates your process may be running out of control.

Assume these pins shall have a nominal diameter of 12.35. Then - based on this sample analysis - you can safely commit holding a tolerance of $\pm 0.05$ (you will hardly produce any scrap as long as your process continues running the way you found it). If your customer would try, however, to force you to accept a tolerance of $\pm 0.02$, you must expect some losses:

| 12.35 ENTERT ENTERT | 12.350 |
| :---: | :---: |
| . $02-$ | 12.330 = lower limit. |
| Norml | $0.006=$ lower scrap $=0.6 \%$. |
| x $x^{2}$ | 12.350 |
| . $02 \pm$ | 12.370 = upper limit. |
| Norml ${ }_{\text {L }}$ | 0.021 = upper scrap $=2.1 \%$. |
| $\pm$ | 0.026 = total scrap. |

What will hurt you even more than these $2.6 \%$ scrap you must expect now (i.e. more than 1 out of 40 pins) will be the inevitable necessity to establish a very precise and reliable sorting tool or device to ensure only pins within $12.350 \pm 0.020$ will pass to your customer. Thus, stay firm (if you can afford it) and refuse that customer request to constrict your tolerance limits - it may well be you cannot afford becoming weak here. Remember the most cost effective parts filter in production is the one you do not need.

Are you interested in the mean pin diameter of your batch? So you know how much space you must provide to store a stack of e.g. 50 pins? Determine the applicable mean and the size of its variation; then use them to find both upper and lower limit confining the mean with a probability of e.g. $95 \%$.

## Example (continued):

Since we have got a sample drawn out of a Gaussian process, the arithmetic mean is applicable, the standard error tells its variation, and Student's $t$ is required. For the latter, we need its degrees of freedom:
(E) $n$
$1-$ STO (1)
STAI $\mathbf{s}_{\mathbf{m}}$
10.000 recall the number of points.
9.000 store the degrees of freedom.
0.003 is the standard error.

Having 95\% inside means having 2.5\% outside at either end (cf. previous plot). ${ }^{126}$ Thus, one must generally take 0.025 and 0.975 as arguments in two subsequent calculations using the quantile function of $t$ to get both $95 \%$ limits below and above the sample result:

| . 025 PROB t: $t^{-1}(p)$ | -2.262 |
| :---: | :---: |
| ( | -0.006 |
| (STAI $\bar{x}$ | 12.352 |
| $\boldsymbol{x}^{2} \boldsymbol{y} \mathrm{R}^{\mathbf{R}}{ }^{127}$ | 12.352 |
| $x \geqslant y$ | -0.006 |
| $\pm$ | 12.346 |
| (RCL (L) | -0.006 |
| $2 \boldsymbol{x} \square^{128}$ | 12.358 |

Now you know what to expect for the future average diameter of such batches. Hence a rail being
$50 \times 617.900$ long inside will suffice for holding 50 pins in $97.5 \%$ of all cases.
12.346 and 12.358 are the $95 \%$ confidence limits of the mean calculated above. So there is a chance of $2.5 \%$ that the mean will be $<12.346$ and an equal chance that it will be $>12.358$. These chances ${ }^{129}$ are an inevitable
${ }^{126}$ The value of $95 \%$ is called the confidence level of this calculation. In this example, you calculate the 95\% confidence limits for the mean value. Instead of 95\%, also 99\% are frequently applied (cf. f. 123). We recommend checking the applicable valid standards before blindly copying any example calculations here. Of course, you are free to apply other confidence levels wherever they fit your needs and you can justify them.
${ }^{127} \overline{\mathbf{x}}$ returns both $\bar{x}$ and $\bar{y}$ (as was shown above). Only $\bar{x}$ is interesting here, so pressing $x^{\geqslant}$y $R \downarrow$ moves $\bar{y}$ quickly out of the way. In a program, DROPy will be a better alternative since it leaves the stack order as is (see OMS 3).

128 The upper confidence limit can be calculated this easy way since $t^{-1}(p)$ is symmetric around the mean value. Else you had to repeat above calculation for an input of 0.975.
${ }^{129}$ Statisticians call these chances 'probabilities of a type I error' or 'probabilities of an error of the $1^{\text {st }}$ kind'.
TNG: Type I error = Fehler 1. Art.
consequence of the fact that you know something about a small sample only (drawn out of a large population), but want or have to tell something about said total population. If you cannot live with these uncertainties or the widths of the confidence limits, do not blame statistics but draw a larger sample or collect more precise data instead. At the bottom line, improving precision and stability of production is the key.

## Application 2 (quick and easy measuring system analysis): <br> Your colleagues of R\&D have happily specified that particle accelerator beam pipes made of a special stainless steel shall have a magnetic susceptibility $\leq 0.01$. How can you check and verify whether the best susceptibility meter available in the laboratory of your organization is sufficiently precise to control series production of such pipes? Or do you need to invest in a better instrument? ${ }^{130}$

Solution (measure the precision of your measuring system):

1. Collect $\geq 30$ metal samples covering the susceptibility range you are interested in. This range could extend e.g. from 0.00 to about 0.02 here. ${ }^{131}$ Mark each sample unambiguously (e.g. by numbering it). Prepare a table with as many numbered rows as you have samples.
2. Use the measuring system under investigation to measure each sample carefully under controlled conditions. Write each measured value next to the respective sample number; record as many decimals as possible.
3. Measure all samples a $2^{\text {nd }}$ time under the same conditions, but following another sample sequence (just shuffle the samples). Do not allow for looking at the $1^{\text {st }}$ data measured (hiding these data will be very helpful if you acquire the values manually)! Record also each $2^{\text {nd }}$ measured value in the row carrying the respective sample number, in a $2^{\text {nd }}$ column.

## 4. Get your WP43. Press STAT CL $\Sigma$ to clear all statistical data. Then enter

 all ( $\geq$ ) 30 pairs of values using $\boldsymbol{\Sigma +}$. The $1^{\text {st }}$ measured value shall be $\boldsymbol{y}$, the[^57]$2^{\text {nd }} \boldsymbol{x}$ - thus, input will be $\boldsymbol{m v 1}$ ENTERT $\boldsymbol{m v 2} \boldsymbol{\Sigma}+$ for each sample (alternatively, you can provide your data in a matrix - see pp. 192f).
5. Call $\boldsymbol{\square} \square$ PLOT for plotting these points in a new window. The plot should look like an ant trail along the diagonal (see a real-world plot here).
6. Press CENTRL to fit an orthogonal regression line to your data points. Check if
 it deviates significantly from the diagonal $\boldsymbol{y}=\boldsymbol{x} .{ }^{132}$ Coarsely, this will apply (with 95\% confidence) if $a_{0} \pm 2 s\left(a_{0}\right)$ excludes 0 or $a_{1} \pm 2 s\left(a_{1}\right)$ excludes 1 .

If it does then your measuring system may be running up still (wait some more time for warmup) and / or show an unstable zero (check and fix); then restart this test at step 2.
7. Else (i.e. if $a_{0} \pm 2 s\left(a_{0}\right)$ includes 0 and $a_{1} \pm 2 s\left(a_{1}\right)$ includes 1 ) press $\mathbf{s}_{\mathbf{m i}}$ to push the precision of your measuring instrument under investigation on the stack. This precision applies under the boundary conditions prevalent during steps 2 and 3 of this test and is definitely more realistic than most catalog values given by manufacturers. ${ }^{133}$ Then enter $\mathbf{3 0 ~} \boldsymbol{x} \times 1 / x$ and multiply with the width of the tolerance zone you want or have to control (. 01 in this example). If you get a result $\geq 1$ then this measuring instrument, device, or system may be used for controlling series production with this tolerance zone under these conditions (i.e. it is a capable instrument for this control job); else you shall seek a more precise instrument etc., better measuring conditions, or a wider tolerance (see the ReM for more).

## Application 3 (significant changes):

## Assume you have drawn a sample out of an arbitrary industrial production process at day 1. Then you have changed the process parameters, waited for stabilization, and have drawn another sample of same size at

[^58]day 2 (there may well have been a longer time interval between both sampling days). Being serious, you have meticulously measured and recorded a critical quantity (e.g. a characteristic dimension) for each specimen investigated at both days. Now, do these two samples show any significant difference?

The following three-step test is well established. It may save yourself easily some unwanted embarrassments in your next presentation or after your next publication: ${ }^{134}$

1. Accumulate your sample data. Then let your WP43 compute the averages and standard errors for both samples, and their normalized distance $d=|\bar{x}-\bar{y}| / \sqrt{s_{m x}^{2}+s_{m y}^{2}}$. The calculation could look like this:
STAI $\mathbf{s}_{\mathbf{m}} \quad$ returns both standard errors in $\mathbf{X}$ and $\mathbf{Y}$.
EXP $x^{2} x^{\geqslant} \geqslant y x^{2} \oplus \sqrt{\boldsymbol{x}}$ so this is the entire denominator.
(STAT $\bar{x} \quad$ returns both $\bar{x}$ and $\bar{y}$.
$\square(|x|)$
thus, this is the numerator
$x^{2} \geqslant y$ (STO (D) and this is $d$ (choose another target register if SsIzE8 is set).
2. Let your WP43 calculate the critical limit $t_{c r}$ of Student's $t$ for $f$ degrees of freedom and a probability of $97.5 \%$ now:
( $\Sigma \mathrm{n}$
$1-$ STO (I)
.975 PROB $t_{\text {: }} t^{-1}(p)$
recall the number of samples measured.
calculate the degrees of freedom $f$ and store them for Student's $t$.
as mentioned above, the requested quantile function lives in PROB. It takes the degrees of freedom stored in I to get $t_{c r}$.

If $d<t_{c r}$ then the test indicates the difference between both samples is due to random deviations only. Congratulations - you have got a robust process regarding the parameters you changed!
Else continue.
3. Let your WP43 compute a new critical limit $t_{c s}$ for $f$ and $99.5 \%$ :

$$
.995 \mathbf{t}^{-1}(p) \quad \text { get } t_{c s}
$$

[^59]If $d \geq t_{c s}$ then the test indicates a significant difference between both samples. Congratulations - your parameter change caused a significant effect!

Else (i.e. for $t_{c r} \leq d<t_{c s}$ ) you simply cannot decide seriously based on the information you have - your samples may contain too little data or your measurements were not precise enough or the process observed is scattering too far etc. Though do not let your audience lead you in temptation: stay silent or mumble something like "investigation in progress" at the utmost management may want a quick answer but there is no solid basis for it here so far.

## Application 4 (operating characteristics):

Assume you draw a representative sample of 20 parts out of a production batch of 100 parts and check this sample thoroughly. What is the probability $P$ to find at least one random defect in such a sample if the overall probability for a defect in such a batch is $5 \%$ or $1 \%$ ?

This is a textbook example for applying the hypergeometric distribution. $P(n \geq 1)$ equals $100 \%-p(n=0)$. Thus, the solution is:
DSP 3

100 STO K
20 STO J
0.05 STO (1)

0.01 STO (1)

0 Hyper $A$
$1-$ +
store batch size
store sample size store 5\% overall defect probability

| returns | 0.319 for $p(\boldsymbol{n}=0)$ |
| :--- | :--- |
| returns | 0.681 for $P(\boldsymbol{n} \geq 1)$ |

store $1 \%$ overall defect probability
returns $\quad 0.800$ for $p(\boldsymbol{n}=0)$
returns $\quad 0.200$ for $\mathrm{P}(n \geq 1)$

Even with 5\% defects in the batch the odds are about 1 out of 3 that no defect at all is detected in such a relatively large sample. Note that such sample tests will be definitely not adequate for controlling industrial processes with overall (target) defect probabilities less than $1 \%$.

## Application 5 (lifetime of punching tools):

Assume being responsible for a workshop punching lots of little parts from sheet metal. The dies you use, manufactured in batches according to the state of the art, will only withstand a limited number of punching cycles. For each individual die of a batch, your technicians meticulously record the number of cycles performed when it breaks.

## Example with FIX 3:

Following cycle numbers were recorded for a batch of 20 dies, arranged according to their lifetimes: 37974,51 340, $62590,77837,79637$, 106 147, 110 632, 112 434, 130 502, 156 003, 166 979, 167 339, 183 419, 205 518, 213 999, 252 171, 254 619, 301 190, 322 290, and 386315.
Derive a characteristic lifetime for this batch from the data collected.

## Solution:

Such lifetime data follow Weibull distributions. For evaluating above record, we shall compute summed transformed 'frequencies' with $\boldsymbol{n}=20$ and $i$ counting the dies:

$$
y_{i}=\ln \left\{\ln \left[\frac{1}{1-\left(\frac{\boldsymbol{i}-0.3}{\boldsymbol{n}+0.4}\right)}\right]\right\}
$$

Solving this problem cries for a program. Programming is covered in OMS 3 - though for the time being, just press GTO $๑ \cdot$, $(P / R)$, and then key in:

LBL $\boldsymbol{\alpha} \sqrt{\boldsymbol{x}} \boldsymbol{x} \boldsymbol{e}^{\boldsymbol{x}} \mathbf{9}$ ENTERT (will be echoed LBL 'FREQ').
$.3-$
20.4 (1)
$1 \quad x^{2} \geqslant y-$
completes the denominator.
$1 / \times x$ ( $\ln$ ) ( $\ln$ ) RTN
Press $\mathbb{P / R}$ again to leave program memory.
Now enter 1 XEQ PROG (FREQ) and you will get $y_{1}$. Repeat for 2 to 20. The results will be:

| $\boldsymbol{i}$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: |
| $\boldsymbol{y}_{i}$ | -3.355 | -2.442 | -1.952 | -1.609 | -1.340 | -1.116 | -0.921 |


| $\boldsymbol{i}$ | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| ---: | ---: | ---: | ---: | ---: | ---: | ---: | ---: |
| $y_{i}$ | -0.747 | -0.587 | -0.438 | -0.297 | -0.160 | -0.026 | 0.107 |


| $i$ | 15 | 16 | 17 | 18 | 19 | 20 |
| ---: | ---: | ---: | ---: | ---: | ---: | ---: |
| $y_{i}$ | 0.243 | 0.384 | 0.535 | 0.704 | 0.910 | 1.216 |

Call (STAT CLE. Then enter these $y_{i}$ and the corresponding numbers of recorded cycles $c_{i}$ via $[\boldsymbol{\Sigma}+]$ into STATS as usual. Plotting these twenty data points ( $c_{i}, y_{i}$ ) over a logarithmic horizontal cycle scale should result in a straight line.
Instead of using such a scale you can let your WP43 perform a logarithmic curve fit through these points: choose LOGF and call ASSESS - it will return a plot as shown here.

The resulting characteristic lifetime of these twenty dies is

## $0 \hat{x}$



After tool composition or manufacturing processes were refined, you receive a new batch of dies from your supplier. Based on a record of their lifetime data, you can now assess the benefit of these changes if there is any (see the Weibull distribution in the $I O I$ and further links there).

Note you must neither use $\overline{\mathbf{x}}$ nor $\mathbf{s}$ here.

## Real Numbers: Frequencies and Histograms

For the evaluation of $1 D$ (or $2 D$ ) samples comprising $\geq 16$ (better $\geq 25$ ) data points, you can categorize these data in bins and analyze their frequencies within these bins.

## How to proceed for new data:

1. Press CLR CL to clear all prior statistical data.
2. Enter your sample data into STATS using STAT $\Sigma+$ as usual.
3. Press $\triangle$ and open HIST.
4. Within this submenu, choose HISTOX if the relevant data are in the $1^{\text {st }}$ column of STATS (e.g. if you entered only $\boldsymbol{x}$ data) or HISTOY if they are in the $2^{\text {nd }}$ column. The data will be copied from the corresponding column of STATS to a new matrix called HISTO. Your WP43 will propose values for the lower limit of the lowest bin, the number of bins, and the upper limit of the top bin.
5. Set different values for $\psi$ BIN, nBINS, and/or $\uparrow$ BIN, if you want. ${ }^{135}$
6. Open HPLOT for plotting the histogram.
7. Optionally, you may fit a 'normal' Gauss curve to your histogram by calling HNORM.

## Example:

Plot a histogram for the following data points: 13,$13 ; 4,5 ; 15,12 ; 6,4$; 12,$13 ; 3,2 ; 17,15 ; 20,17 ; 10,10 ; 11,10 ; 1,1 ; 13,11 ; 8,8 ; 10,8 ;$ 12, 10; 10, 9.

## Solution:

## CLR CLE

13 ENTERT 13 STAT $\Sigma+$
5 ENTERT $4 \Sigma$ etc. until 9 ENTERT $10 \Sigma+$


Choose HISTOX and you will get: $|$| nBINS $=$ | 4. |
| :--- | ---: |
| $\psi B I N=$ | 1. |
| $\uparrow B I N=$ | 17. |

To continue with these auto- 2022-09-30 18:24 matic parameters just open Histogram( $x$ )
To continue with thes
matic parameters jus
HPLOT and you will see:


[^60]Let's see how this histogram compares with a Gauss curve. Just press F1 (= HNORM) and you will get:


You can do the same for the $y$ column of your STATS data now by EXITing this diagram and selecting HISTOY:

Continue with these automatic parameters - open HPLOT and you will see:


STAT encompasses many more statistical functions (e.g. covariances, means and standard deviations for weighted data, geometric means and scattering factors) - just look them up there and check the respective entries in the IOI.
 them individually by calling its name (no need to memorize any register numbers in this matter anymore).

More examples of statistical applications can be found in the manuals of various vintage HP calculators, especially of the HP-27 and HP-21S.

We strongly recommend you consult a good statistics textbook for more information about statistical methods in general, the terminology used, and the mathematical models provided, before applying them.

## Real Numbers: Summary of Functions

The majority of the functions your WP43 features are for calculations operating on reals. It provides many more than the numeric functions shown on pp. 20ff, 29ff, and 87 ff in various applications and examples. See all real functions listed below:

- General mathematics:
- Monadic functions:
 and ( $\mathrm{Ib} x$ ), $\left[10^{x}\right.$ and $(19), e^{x}$ and $(10)$, (sin), (cos), (tan), and their inverses work as shown above and you learned in school (see also pp. 136ff for more information about angular I/O),
for (sinh], [cosh], (tanh), and their inverses compare pp. 101f, $\left[e^{x}-1\right]$ and $[\ln (1+x)]$ return more accurate results for $x \approx 0$, (ceil) returns the smallest integer $\geq \boldsymbol{x}$, while (floor) returns the greatest integer $\leq \boldsymbol{x}$,
(SDL) $\boldsymbol{n}$ shifts digits left by $\boldsymbol{n}$ decimal positions, equivalent to multiplying $x$ times $10^{n}$,
(SDR) $\boldsymbol{n}$ shifts digits right by $\boldsymbol{n}$ decimal positions, equivalent to dividing $x$ by $10^{n}$,
$\left[(-1)^{\times}\right]$returns $\cos (\pi x)+i \sin (\pi x)$ for non-integer $x$;
\# converts a closed real to an integer of base 2... 16 (see Integers some chapters below); $(\rightarrow B I N),(\rightarrow O C T),(\rightarrow D E C)$, and $(\rightarrow H E X)$ cover the most popular bases.
- Dyadic functions:
$\oplus, \boxplus, \mathbb{X}, \rrbracket,\left(\mathbb{y}^{x}\right.$, and $(\sqrt[x]{y})$ work as was shown above and you learned in school;
use (IDIV) for integer division (and (IDIVR) if you want also the remainder returned in $\mathbf{Y}$ ),
(e.g. 7.8 ENTERT 3.2 (INTS IDIV returns 2)
$\left[\log _{x} y\right]$ for the logarithm of $\boldsymbol{y}$ for the base $\boldsymbol{x}$
(e.g. 625 ENTERT 5 EXP $\log _{x} y$ returns 4),
(RMD] for the remainder of $\boldsymbol{y} / \boldsymbol{x}$ (see p. 157 for examples),
(MOD for $\boldsymbol{y} \bmod \boldsymbol{x}$ (see p. 158 for examples),
(max] (or $(\min )$ ) for the maximum (or minimum) of $\boldsymbol{x}$ and $\boldsymbol{y}$; and

$$
\begin{aligned}
& \text { returns }\left(\frac{1}{x}+\frac{1}{y}\right)^{-1} \text { for } x \times y \neq 0 \text { and } \\
& 0 \text { else, being handy in electrical } \\
& \text { engineering in particular. }
\end{aligned}
$$

- Triadic functions:

[ $\times$ MOD] returns $(z \times y) \bmod \boldsymbol{x}$ for $x>1, y>0, z>0$;
[ $\wedge$ MOD] returns $\left(z^{y}\right) \bmod x$ for $x>1, y>0, z>0$.
- Isolating parts of numbers: Use the monadic functions ...
(EXPT) for the exponent of $\boldsymbol{x}$ and (MANT) for its mantissa,
(Ix|) for the absolute value of $\boldsymbol{x}$,
[FP] (or (IP)) for the fractional (or integer) part of $\boldsymbol{x}$, preserving its sign, and [sign] for the signum of $\boldsymbol{x}$; SIGN returns 1 for $\boldsymbol{x}>0,-1$ for $\boldsymbol{x}<0$, and 0 for $\boldsymbol{x}=0$ or non-numeric data.
- Rounding:
(RDP) $\boldsymbol{n}$ rounds $\boldsymbol{x}$ to $\boldsymbol{n}$ decimal places in FIX format (e.g. $1.23456789 \times 10^{-95}$ RDP 99 returns $1.2346 \times 10^{-95}$ ),
(ROUND) rounds $\boldsymbol{x}$ using the current display format, ${ }^{136}$
(ROUNDI) rounds $\boldsymbol{x}$ to next integer ( $1 / 2$ rounds to 1 ),
(RSD] $\boldsymbol{n}$ rounds $\boldsymbol{x}$ to $\boldsymbol{n}$ significant digits, and
SHOW displays all 34 digits of $\boldsymbol{x}$ until next keystroke.

[^61]- Conversions:
$\Leftrightarrow$ P converts rectangular to polar coordinates (cf. pp. 20f), while $R \leftrightarrows$ converts vice versa.


For angular, time, and date conversions see pp.
139 ff and 197ff, for unit conversions look up pp. 292ff.

- Boole's algebra:
[AND], (NAND), (OR), (NOR), (XOR), (XNOR), and (NOT) operate on reals like they did in the HP-28S, i.e. $\boldsymbol{x}$ and $\boldsymbol{y}$ are interpreted before executing the operation. Zero is 'false' ( $=0$ ); any other number is 'true' (= 1).

$$
\text { Example: } 13.5 \text { ENTERT } \mathbf{- 7 . 2} \text { BIT AND returns } 1 .
$$

- Probability \& statistics (unless introduced and shown on pp. 102-129): $[\Gamma(x)]$ calculates the Gamma function,
[ $\ln \Gamma$ ] returns the natural logarithm of the Gamma function, allowing also for calculating great factorials:

Example: What is 5432!?
Remember $\Gamma(x+1)=x$ ! So, entering 5433 X.FN $\ln \Gamma 10$ In (1) returns 17931.48037401087 as decadic logarithm of the result. Then calling (PART) FP $10^{x}$ will return $3.02255 \ldots$ for its mantissa.
Thus, $5432!\approx 3.02255 \times 10^{17931}$.
[RAN\#] returns a uniformly distributed (pseudo) random real number

$$
x \in[0,1),
$$

[SEED] stores a seed (i.e. a start value) for RAN\#,
(RANI\#) returns a uniformly distributed (pseudo) random integer number $\in[\boldsymbol{x}, \boldsymbol{y}]$; you can use it for throwing dices.

The other contents of PROB cover combinations, permutations, and the 14 distributions introduced on pp. 103ff.
$\underline{\underline{\Sigma}}$ contains all the accumulated sums of your STATS data, callable by their names.

In STAT, you find the summation commands $[\boldsymbol{\Sigma}+],[\boldsymbol{\Sigma}-]$, and [CLE], various mean values ( $[\overline{\mathbf{x}}],\left[\overline{\mathbf{x}}_{\boldsymbol{W}}\right],\left[\overline{\mathbf{x}}_{\mathbf{G}}\right],\left[\overline{\mathbf{x}}_{\mathbf{H}}\right],\left[\overline{\boldsymbol{x}}_{\text {RMS }}\right\rceil$ ), sample standard deviations ( $[\mathbf{s}],\left[\mathbf{s}_{\mathbf{w}}\right)$ ) and standard errors ( $\left[\mathbf{s}_{\mathbf{m}}\right],\left[\mathbf{s}_{\mathbf{m}}\right]$ ), population standard deviations ( $[\mathbf{G}],\left[\boldsymbol{\sigma}_{\mathbf{w}}\right]$ ), various scattering factors ( $\left.[\boldsymbol{\varepsilon}],\left[\varepsilon_{\mathbf{m}}\right],\left[\boldsymbol{\varepsilon}_{\mathbf{p}}\right]\right)$, correlations and covariances, as well as all commands and menus related to curve fitting ([L.R.), the ten fit models, [ASSESS] for plotting and assessing fits, forecasting for x and y), [HIST0] for histograms, [PLOT] for assessing the performance of measuring instruments and systems in real life, etc.

Turn to the $\operatorname{Re} M$ for comprehensive information about all the probability and statistical functions provided on your WP43.

- Percentages:
\% calculates $x y / 100$, leaving $y$ as is (so you can easily calculate another percentage of the same base after CLX).


## Example (from the HP27 OH):

If you buy a new car, you have to figure the sales tax percentage, then add that to the purchase price to find the total cost of the car. ... For example, if the sales tax on a $\$ 6200$ car is $5 \%$, what is the amount of the tax and total cost of the car?

## 6200 ENTERT 5 FIN \% returns 310. US\$ for the sales tax; returns 6510 . US\$ for the total cost.

If the dealer gives you a $10 \%$ discount on the car, what will your total cost be?

## 6200 ENTERT

$10 \%$ returns 5 580. US\$ for the discounted price;
$5 \% \oplus$ returns 5 859. US\$ for the total cost.
calculates the percentage of change from $\boldsymbol{y}$ to $\boldsymbol{x}$, returning $100^{x-y} / y$, leaving $y$ as is (for same reason as with \%). Feel free to use $\Delta \%$ also for calculating markup ${ }^{137}$ or margin:

[^62]
## Example:

You purchase ink cartridges for 21.99 US\$ wholesale and retail them for 26.50 US\$. What percent is your markup and what percent is your margin? ${ }^{138}$
21.99 ENTERT $26.5 \triangle \%$ returns 20.5 \% markup.
26.5 ENTERT $21.99 \triangle \%$
returns -17.0, i.e. 17 \% margin.
\%MRR calculates the mean rate of return in \% per period with $\boldsymbol{y}=$ present value, $\boldsymbol{x}=$ future value after $\boldsymbol{z}$ periods, ${ }^{139}$
$\% \mathrm{~T}$ calculates $100^{x} / y$ (called "\% of total"), leaving $y$ as is,
$\% \Sigma$ returns $100^{x} / \sum x$, and
$\%+$ MG calculates a sales price by adding a margin ${ }^{138}$ of $x \%$ to the cost $\boldsymbol{y}$; ${ }^{139}$ you may use $\%+$ MG for calculating net amounts as well - just enter a negative percentage in $\boldsymbol{x}$.

## Example:

Total billed $=221,82 €$, VAT $=19 \%$. What is the net?

$$
221.82 \text { ENTERT } 19 \text { †/L FIN \%+MG returns } 186,40(€) .{ }^{140}
$$

[^63]- Advanced mathematics (see the ReM, App. I for comprehensive information about the functions following):
- Monadic functions:
$\left[\mathbf{B}_{\mathrm{n}}\right]$ and $\left[\mathbf{B}_{\mathrm{n}}{ }^{*}\right]$ return the Bernoulli numbers,
(erf) and (erfc) the error function and its complement,
(FIB) the extended Fibonacci number,
[ $\left.\boldsymbol{g}_{d}\right]$ and $\left[\boldsymbol{g}_{d}{ }^{-1}\right]$ the Gudermann function and its inverse, and
[NEXTP) the next prime number greater than $\boldsymbol{x}$;
[sinc] returns $\sin (x) / x$ and $[\boldsymbol{\operatorname { s i n }} \boldsymbol{\pi} \pi]$ returns $\sin (\pi x) / \pi x$ for $\boldsymbol{x} \neq 0$ and 1 for $\boldsymbol{x}=0$,
[ $W_{p}$ ] returns the principal branch of Lambert's $W$ for given $x \geq-1 / \mathrm{e},\left(W_{m}\right)$ the negative branch of it,
( $W^{-1}$ ) returns $\boldsymbol{x}$ for given $\mathrm{W}_{\mathrm{p}}(\geq-1)$,
$[\mathrm{K}(\mathrm{m})]$ and $[\mathrm{E}(\mathrm{m})]$ return the complete elliptic integrals of the $1^{\text {st }}$ and $2^{\text {nd }}$ kind, and
( $\zeta(x)]$ Riemann's Zeta function.
Call $\left[H_{n}\right]$ for the Hermite polynomials for probability and
$\left(\mathrm{H}_{\mathrm{np}}\right)$ for the Hermite polynomials for physics,
( $\mathrm{L}_{\boldsymbol{n}}$ ) for Laguerre's polynomials and
[ $\mathrm{L}_{\mathrm{n} \alpha}$ ] for Laguerre's generalized polynomials,
( $\mathbf{P}_{\mathbf{n}}$ ) for the Legendre polynomials,
$\left[T_{n}\right.$ ] for the Chebyshev polynomials of $1^{\text {st }}$ kind and
$\left[U_{n}\right]$ for the Chebyshev polynomials of $2^{\text {nd }}$ kind.
- Dyadic functions:
[AGM] returns the arithmetic-geometric mean,
$\left[J_{y}(x)\right]\left(\left[Y_{y}(x)\right]\right)$ the Bessel function of $1^{\text {st }}\left(2^{n d}\right)$ kind and order $y$, $[\Pi(n, m)]$ the complete elliptic integrals of the $3^{\text {rd }}$ kind,
$[s n(u, m)],[\mathrm{cn}(u, m)]$, and $[\mathrm{dn}(u, m)]$ the Jacobi elliptic sine, cosine, and amplitude functions,
$[\mathbf{Z}(\varphi, m)]$ returns Jacobi's Zeta function,
$[\beta(x, y)]$ Euler's Beta function and $(\ln \beta)$ its natural logarithm, $\left[\mathbf{I} \Gamma_{\mathbf{p}}\right]$ and $\left[\mathrm{I} \Gamma_{q}\right]$ return the regularized gamma functions, $\left[\boldsymbol{\gamma}_{\mathrm{x}}\right]$ ] the lower and $\left[\boldsymbol{r}_{\mathrm{xy}}\right]$ the upper incomplete gamma function.
- Triadic function:
[ $\mathrm{I}_{\mathbf{x y z}}$ ] returns the regularized beta function.


## Rational Numbers (Fractions)

On your WP43, you can work with fractions with absolute values between 1000000 and 0.000 1; maximum denominator is 9999 (DENMAX accepts greater maximum denominators but reduces them as soon as input is closed). ${ }^{141}$

A fraction may be entered directly by keying in a $2^{\text {nd }} \square$ in numeric input. Herein, the $1^{\text {st }}$. is interpreted as a blank space, the $2^{\text {nd }}$ as a fraction mark:

| Examples: <br> Key in: | ... and get in startup default format (FRGSRN) |
| :---: | :---: |
| (1) 2 3 3 EXIT | 12 3/4 |
| -10 $0^{2}$ EXIT | $01 / 2$ |
| (3) $\odot \bigcirc$ EXIT | $31 / 2$ takes last denominator entered ${ }^{142}$ |
| (1) (1) 4 EXIT | $10 / 1^{143}$ input was $1 / 4$ |
| (1) 15 EXIT | $0 \%$ input was $0 \% 15$ |

Any closed real on the stack will be displayed as a fraction after ab/c is pressed, after a fraction is entered as demonstrated, or after $\boldsymbol{x}$ is combined with a fraction by an arithmetic operation. If the fraction
 displayed is slightly greater or slightly less than the underlying (hidden) real, < or > will head this fraction, respectively (see the examples following). Get a more verbose notation (e.g. $x=1 / 2$ or $x<3 / 7$ ) with FRCSRN set, applying to the other stack registers in analogy. ${ }^{144}$

[^64]Vice versa, any closed number displayed as a fraction will be replaced by its real value again after .d or DISP ALL, FIX, SCI, or ENG. And a closed fraction will be decomposed to its long integer numerator in $\mathbf{Y}$ and denominator in $\mathbf{X}$ by (PART DECOMP; this can be simply reverted by division.

There are two fraction display modes: proper and improper fractions. ${ }^{145}$ toggles them, starting with proper fractions as illustrated below. The following example comprises most aspects of fraction display (assuming startup default settings):


Now, press ab/c for converting this improper fraction to a proper one. ${ }^{146}$
You will get
11372 105/512
11 ( 1 > $10334713 / 5632$
This fraction is less than the underlying real $\boldsymbol{x}$, deviating less than $0.5 / 5632$ from it.

Now, let's reduce the maximum denominator by

64 MODE DENMAX

## R】

CF SYS.FL DENANY

## 64

> 1033 41/49
< 1033 27/32
since Denany allows for denominators $2,4,8,16$, 32, and 64 here only (DENFIX is clear since startup).

[^65]This last fraction shown is greater than the underlying real $\boldsymbol{x}$; the difference is $<0.5 / 32$ (and $>0.5 / 64$ - else the display would read $103353 / 64$ instead).

Note that a decomposition of a fraction by DECOMP preserves full precision always, returning $\boldsymbol{y}=5822569$ for the numerator and $\boldsymbol{x}=5632$ for the denominator here. So you can revert DECOMP simply by dividing.

Another example, now with maximum denominator 12:
DENANY allows for any denominator $\leq 12$ (i.e. $12,11,10, \ldots, 4,3$, and 2 ).
DENANY \& DENFIX allows for denominators being factors of 12 (i.e. 12, 6, 4, 3, and 2).
DENANY \& DENFIX allows for a fix denominator 12 only.

## Before closing this chapter about rational numbers, let's not forget those isolated irrational islands in the vast sea of S/ where you may come across dimensions like in the following example:

A calculator stand is specified to measure $9^{\prime \prime} \times 31 / 2^{\prime \prime} \times 5 / 8$ " (in 2019). It goes without saying that your WP43 will support you also in such harsh environments. Only absolute greenhorns, however, will expect that a tight thin-walled box around this stand will displace


Instead, a magic conversion factor from cubic inches to so-called fluid ounces is required now, even depending on the country you are in! 147
Though do not despair: In OMS 5 you will learn how to do this magic using your WP43 - it takes just a little more time and effort than calculating with rational units.

[^66]
## Angles and Trigonometric Functions

For dealing with angles, you may choose out of six angular display modes (ADM) on your WP43: DEG, RAD, GRAD, MULT, MIL, and D.MS. ${ }^{148}$ Angles are entered as reals and interpreted according to the current ADM as indicated in the status bar by $4^{\circ}, 4^{\mathbf{r}}, 4^{\mathbf{g}}, \Varangle \pi$, or $4^{-}$(cf. p. 80) when a function expecting angular input is called.

Exception: Sexagesimal degrees must be entered in the format ddddd.mmsshh d.ms - with ddddd standing for integer degrees, mm for angular minutes, ss for seconds, and hh for hundredth of seconds.


## Examples:

12.3456543
45.6789018
returns
returns
$12^{\circ} 34^{\prime} 56.54^{\prime \prime}$,
$46^{\circ} 8^{\prime} 29.01^{\prime \prime}$.

There are some functions (e.g. ARCSIN) operating on reals and returning angles. Then the returned values will be automatically tagged according to the current $A D M$. Assume ALL 2 set for the following examples:

| In ADM ... | ... 1 TRI arcsin will return ... |
| :---: | :---: |
| decimal RADians $\boldsymbol{4}^{\boldsymbol{r}}$ | $1.570796326794897{ }^{\text {r }}$ |
| MUL tiples of $\boldsymbol{\pi} \quad \Varangle \pi$ | $0.5 \pi$ |
| decimal DEGrees $\mathbf{4}^{\circ}$ | 90. ${ }^{\circ}$ |
| Degrees Minutes Seconds $\boldsymbol{4}^{\prime \prime}$ | $90^{\circ} 0^{\prime} 0.00^{\prime \prime} 149$ |
| GRADians or gon $\mathbf{4}^{\mathbf{g}}$ | 100.9 |
| MIL $4^{-}$ | 1 600.- |

[^67]Whenever you see a number formatted alike on your WP43 you know it is an angle. - Other functions presume their inputs being angles, e.g. SIN. There, decimal inputs are generally interpreted as angles of current ADM.

22 angular conversions are provided in $(\underline{x} \rightarrow$ :

|  |  | decimal degrees | decimal radians | multiples of $\pi$ | gradians / gon | mil | current ADM or tagging |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| sexag. degrees | - | D $\rightarrow$ D.MS | - | - | - | - | $\rightarrow$ D.MS |
| dec. degrees | D.MS $\rightarrow$ D | - | $\mathrm{R} \rightarrow \mathrm{D}$ | - | - | MIL $\rightarrow$ D | $\rightarrow$ DEG |
| dec. radians | - | $\mathrm{D} \rightarrow \mathrm{R}$ | - | $M \pi \rightarrow R$ | - | MIL $\rightarrow$ R | $\rightarrow$ RAD |
| multiples of | - | - | $\mathrm{R} \rightarrow \mathrm{M} \boldsymbol{\pi}$ | - | - | - | $\rightarrow$ MUL $\pi$ |
| gradians / gon | - | - | - | - | - | - | $\rightarrow$ GRAD |
| mil | - | D $\rightarrow$ MIL | R $\rightarrow$ MIL | - | - | - | $\rightarrow$ MIL |
| current ADM | D.MS $\rightarrow$ | DEG $\rightarrow$ | RAD $\rightarrow$ | MULT $\rightarrow$ | GRAD $\rightarrow$ | MIL $\rightarrow$ | - |

Example (with FIX 5):
MODE MUL $\boldsymbol{\pi} \quad$ Choose multiples of $\pi$ as ADM and $\boldsymbol{\Varangle \pi}$ will appear in the status bar and stay there for the time being.
150 11/x
(x $\rightarrow$ RAD
$\rightarrow$ DEG
$\rightarrow D . M S$
$\rightarrow M I L$
$\rightarrow M U L \pi$

$$
\begin{aligned}
& 0.00667 \text { So } \pi / 150 \ldots \\
& 0.02094^{r} \text { are } 0.02094 \text { radians } \\
& 1.20000^{\circ} \text { or exactly } 1.2^{\circ} \\
& 1^{\circ} 12^{\prime} 0.00^{\prime \prime} \text { or } 72 \text { angular minutes } \\
& 21.33333^{-} \text {or } 21.33 \ldots \text { mil } \\
& 0.00667 \pi \text { equivalent to } \pi / 150 \text { still. }
\end{aligned}
$$

Note $\rightarrow$ RAD 'knew' it had to convert from multiples of $\pi$ since this function expects angular input and takes the current ADM setting into account. Angular output of operations is tagged and will stay so. Thus, $\rightarrow$ DEG above converted from radians, $\boldsymbol{\rightarrow} \mathbf{D} . \mathrm{MS}$ from decimal and $\boldsymbol{\rightarrow} \boldsymbol{M U L} \boldsymbol{\pi}$ from sexagesimal
will display down to that fraction always and cannot be shortened (cf. examples above)..
degrees since the respective inputs were tagged. Also .d converts sexagesimal degrees into decimal degrees but without tagging.

You have learned about trigonometric functions in school. Thus, we demonstrate their operation on angles with one example only:


Lovesick sailor Oscar Odysseus dwells on the island of Tristan da Cunha ( $37^{\circ} 03^{\prime} \mathrm{S}, 12^{\circ} 8^{\prime} \mathrm{W}$ ), and his sweetheart, Penelope, lives on the nearest island. Unfortunately for the course of true love, however, Tristan da Cunha is the most isolated inhabited spot in the world. If Penelope lives on the island of St. Helena ( $15^{\circ} 55^{\prime} \mathrm{S}, 5^{\circ} 43^{\prime} \mathrm{W}$ ), use the following formula to calculate the great circle distance that Odysseus must sail in order to court her. ${ }^{150}$

## Solution:

The formula for the great circle distance $\boldsymbol{d}$ in nautical miles is:

$$
d=60 \times \arccos \left[\sin \left(B_{s}\right) \sin \left(B_{d}\right)+\cos \left(B_{s}\right) \cos \left(B_{d}\right) \cos \left(L_{d}-L_{s}\right)\right]
$$

with $B_{s}$ and $L_{s}$ being the latitude and longitude of the start (Tristan da Cunha) and $B_{d}$ and $L_{d}$ being the latitude and longitude of the destination (St. Helena). ${ }^{151}$ Hence, with the numbers inserted, this formula reads:

$$
\begin{aligned}
d=60 \arccos [ & \sin \left(37^{\circ} 03^{\prime} S\right) \sin \left(15^{\circ} 55^{\prime} S\right) \\
& \left.+\cos \left(37^{\circ} 03^{\prime} S\right) \cos \left(15^{\circ} 55^{\prime} S\right) \times \cos \left(5^{\circ} 43^{\prime} W-12^{\circ} 18^{\prime} W\right)\right]
\end{aligned}
$$

Set the appropriate number of decimals for display and calculate from inside out, remembering the trigonometric functions assume their input being in the current $A D M$ as indicated in the status bar.


## DISP FIX 2

Since we will use sexagesimal degrees throughout this calculation, we set ADM accordingly. We will not need more decimals displayed.

12.18 d.ms

[^68]| $\square$ | -6 ${ }^{\circ} 35^{\prime} 0.00$ " |  |
| :---: | :---: | :---: |
| TRI) cos | 0.99 |  |
| 15.55 STO [J cos | 0.96 |  |
| W | 0.96 |  |
| 37.03 STO K cos | 0.80 |  |
| - | 0.76 |  |
| RCL (K) sin | 0.60 |  |
| (RCL (J) $\mathbf{s i n}$ | 0.27 |  |
| - | 0.16 |  |
| $\dagger$ | 0.93 |  |
| arccos | $22^{\circ} 15^{\prime} 36.15^{\prime \prime}$ |  |
| .d | 22.26 | convert to a real.. |
| $60 \times$ returns | 1335.60 | nmi or |
| $U \rightarrow$ xi nmi $\rightarrow$ m | 2473535.88 | , i.e. some 2474 km that Odysseus |

## Mixed Calculations: Coordinate Transforms in 2D, Flight Directions, Courses over Ground, etc.

Two functions are provided for converting polar or rectangular coordinates in two dimensions. Input and output data are both in stack registers $\mathbf{X}$ and $\mathbf{Y}$ here.
$\leftrightarrow$ P converts 2D Cartesian coordinates $\boldsymbol{x}$ and $\boldsymbol{y}$ to polar magnitude or radius $\boldsymbol{r}$ in $\mathbf{X}$ and angle $\boldsymbol{\vartheta}$ in $\mathbf{Y}$.


## Example (assuming startup default settings):

Convert $(\boldsymbol{x}, \boldsymbol{y})=(6,4.5)$ to polar. Two decimals shall do.

## Solution:


$36.87^{\circ}$ 7.50
i.e. a vector of magnitude 7.5 pointing up right from the origin with an angle of some $37^{\circ}$ to the positive $\boldsymbol{x}$-axis.
$R \backsim$ does the reverse: it converts $2 D$ polar magnitude or radius $r$ in $\mathbf{X}$ and angle $\boldsymbol{\vartheta}$ in $\mathbf{Y}$ to Cartesian coordinates $\boldsymbol{x}$ and $\boldsymbol{y}$. Both functions honour the ADM settings and tags as described in previous chapter.

## Example (continued):

Convert the returned angle of the conversion executed above to radians, and then convert the resulting coordinates $(\boldsymbol{r}, \boldsymbol{\vartheta})$ to rectangular.

## Solution:



Note angular input can range from $-\infty$ to $+\infty$; angular output (unless of angular conversions), however, is confined to $-180^{\circ}$ to $+180^{\circ}$ or its equivalents, i.e. $-\pi$ to $+\pi$ in radians, -200 g to +200 g in grades, and -1 to +1 in multiples of $\pi$.


## Example (from HP-29C OHPG):

Engineer Trigo Slothrop has determined that in the RC circuit shown..., the total impedance is $77.8 \Omega$
 and voltage lags current by $36.5^{\circ}$. What are the val-
 ues of resistance $\boldsymbol{R}$ and capacitive reactance $X_{c}$ in the circuit?

## Solution:

The situation is drawn at left. Of the total impedance, $\boldsymbol{R}$ is its component parallel to
the $\boldsymbol{x}$-axis, and $\boldsymbol{X}_{c}$ is its perpendicular component parallel to the $\boldsymbol{y}$-axis:
DISP) FIX 0 ( 1
-36.5 ENTERT 77.8

|  | $y=$ | -46.3 |
| :--- | :--- | :---: |
|  | $x=$ | 62.5 |
|  | i.e. a resistance of $62.5 \Omega$ and a reactance of $46.3 \Omega$. |  |

Note you can use $\rightarrow P$ and $R \leftrightarrow$ also to convert $3 D$ cylinder coordinates to Cartesian and vice versa, since $\boldsymbol{z}$ is kept unchanged.

Having learned about $\Leftrightarrow P$ and $R \leftarrow$ as well as about $[\Sigma+$, we can profit from combining these functions now.

## Example (from the HP-25 OH):

The instruments in fearless bush pilot Apeneck Sweeney's converted P-41 indicate an air speed of 125 knots and a heading of $225^{\circ}$. However the aircraft is also being buffeted by a steady $25-k n o t$ wind that is blowing from north to south. What is the actual course and speed of the aircraft?


## Solution:

Combine the vector indicated on the aircraft instruments with the wind vector to yield the actual course and speed. Convert the vectors to rectangular, then combine the $\boldsymbol{x}$ - and $\boldsymbol{y}$-coordinates in the statistic storage. Finally, recall the summed $x$ - and $y$ coordinates and convert them to polar coordinates giving the actual vector of the aircraft. (North becomes the $\boldsymbol{x}$-coordinate in order that the problem corresponds with navigational convention.)
(CLR CLE clears the statistics storage.
(DISP) FIX 2
225 ENTERT 125 indicated heading and air speed.

| R | returns | -88.39 |
| :--- | :--- | :--- |
|  | $x=$ | -88.39 |

(STAT $\Sigma+$
180 ENTERT 25
 returns

$$
y=
$$

$$
0.00
$$

$$
x=
$$

$$
-25.00
$$

| £ + | adds $\boldsymbol{x}$ and $\boldsymbol{y}$ to the statistics storage. |  |  |
| :---: | :---: | :---: | :---: |
| SUM | recalls the summation registers $\Sigma \mathbf{x}$ and $\boldsymbol{\Sigma y}$ |  |  |
| $\rightarrow \mathrm{P}$ | returns | $\vartheta=$ | $-142.06^{\circ}$ |
|  |  | $r=$ | 143.77 |
| $x^{2} \geqslant 9360 \pm$ | returns |  | $217.94^{\circ}$ |

(we have to change the angle to become positive for being in line with navigational convention).

So, Mr. Sweeney is actually flying at 143.77 knots on a course of $217.94^{\circ}$ over ground. ${ }^{152}$

A similar problem appeared first in the HP-55 OH and was copied for some years. We quote it from the $H P-33 \mathrm{OH}$ :



## Example:

On his way to search for an albino caribou, grizzled bush pilot Apeneck Sweeney's converted Swordfish aircraft has a true air speed of 150 knots and an estimated heading of $45^{\circ}$. The Swordfish is also being buffeted by a headwind of 40 knots from a bearing of $25^{\circ}$. What is the actual ground speed and course of the Swordfish? ${ }^{153}$

[^69]
## Start of solution:

Taking into account that a bearing of $25^{\circ}$ equals a heading of $25^{\circ}+180^{\circ}=$ $205^{\circ}$, the corresponding headwind vector may be added (cf. the HP-97 OHPG). - We leave it to you to solve this problem using $\Sigma+$ (but give you the results for crosschecking: $51.94^{\circ}$ and 113.24 knots).

Additionally, here is an advanced problem from a universe far, far away:

## Example from the HP-32 OH: ${ }^{154}$



Federation starship Felicity has emerged victorious from a furious battle with the starship Thanatos from the renegade planet Maldek. ${ }^{155}$ However, its automatic pilot is kaput, and its main thrust engine is locked on at 37.2 meganewtons (MN) directed along an angle of $25.2^{\circ}$ from the star Ultima. ${ }^{156}$ Consulting the ship's star map, the navigator reports a hyperspace entrance vector of 51 MN at an angle of $41.3^{\circ}$ from Ultima. To what thrust and angle should the auxiliary engine be set, for Felicity to achieve alignment with the hyperspace entrance vector?

## Solution (with FIX 01):

The required thrust vector of the auxiliary engine is equal to the hyperspace entrance vector minus the thrust vector of the main engine. The vectors are converted to rectangular coordinates using $(\mathbb{R} \leftrightarrows$, and their difference is calculated using $\Sigma+$... This difference is recalled to $\ldots \mathbf{X}$ and $\mathbf{Y} \ldots$ using SUM).
${ }^{154}$ This example is from 1978. Note the $1^{\text {st }}$ episode of Star Wars was launched in 1977.
${ }^{155}$ TN: a) Thanatos (Өávatos) is the ancient Greek word for 'death', pronounced like 'Tunnatoss' in English but like 'Tanatos' in Spanish, Portuguese, Italian, French, German, Swedish, and Finnish, for example.
b) Why was it called Maldek instead of e.g. Bendek? Subliminal viewer manipulation...
${ }^{156} T N$ : Latin for 'the last' (fem.).

Then, these rectangular coordinates of the auxiliary engine thrust vector are converted to polar coordinates using $\leftrightarrows P$.

| CLR | CLI |  | clears all statistical data. |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 41.3 | ENTERT | 51 | hyperspace entrance vector |  |  |
| R+ |  |  | returns | $y=$ | 33.7 |
|  |  |  |  | $x=$ | 38.3 |

STAI $\boldsymbol{\Sigma}+\quad$ puts the $x$ and $y$ components of the hyperspace entrance vector into the statistics storage.
25.2 ENTERT 37.2 main engine thrust vector

| R | returns | $y=\quad 15.8$ |  |
| :---: | :---: | :---: | :---: |
| +4 $x^{2} \mathrm{y}$ + $x^{2} \times y$ | inverts both its Cartesian components. ${ }^{157}$ |  |  |
| $\boldsymbol{\Sigma}+$ | adds the $\boldsymbol{x}$ and $\boldsymbol{y}$ components of the negative main engine thrust vector to the statistics storage. |  |  |
| SUM | recalls the summation registers: |  |  |
|  |  | $\begin{aligned} & \Sigma y= \\ & \Sigma x= \end{aligned}$ | $\begin{array}{r} 17.8 \\ 4.7 \end{array}$ |
| $\rightarrow \mathrm{P}$ | returns | $\vartheta=$ | $75.4{ }^{\circ}$ |
|  |  | $r=$ | 18.4 |

meaning the auxiliary engine shall be set at 18.4 MN and an angle of $75.4^{\circ}$ from Ultima. ${ }^{158}$

See the operating manuals of other vintage HP calculators (especially the HP-27) for further applications in the area of angular mathematics (e.g. triangle solutions, navigation, and surveying).

[^70]
## Angles: Summary of Functions

The number of functions operating on and with angles is quite limited. They are important nevertheless. All are listed below:

- General mathematics:
- Monadic functions:
[sin], (cos), and [tan] operate on angles and return reals, [arcsin], [arccos], and [arctan] operate on reals and return angles,
for (sinc) and $[$ sincr $]$ see the $I O I$ in the $R e M$, please,
(+/ ) returns $\boldsymbol{x} \times(-1)$ for closed input (a.k.a. 'unary minus').
- Dyadic functions:
$\oplus, \boxed{\square}, \boxed{x}$, and $\square$ work as specified in the matrices on pp. 75f, [max] (or (min)) returns the maximum (or minimum) of $\boldsymbol{x}$ and $\boldsymbol{y}$.
- Rounding commands (RDP, ROUND, ROUNDI, RSD, and SHOW) operate on angles as they do on reals (cf. pp. 130f).
- Conversions:
$\rightarrow$ P converts rectangular to polar coordinates (cf. pp. 20f), while $R \leftrightarrow$ converts vice versa. Cf. examples on pp. 142ff.


Angular conversions are covered comprehensively on p. 140.

## Integers: Input and Displaying

Any single number you enter without using $\square$ or $E$ (or CC, see below) is regarded as an integer by your WP43 (cf. pp. 72f). Your WP43 allows for integer computing in 15 bases from binary to hexadecimal.

Any single number displayed without any punctuation on your WP43 is an integer (e.g. a counted value, see examples below). And it will stay integer as long as it is exclusively combined with other integers and only integer functions operate on it; else it will be converted to another $D T$ (cf. the matrices on pp. 75 f and $R e M S 3$ ). On the other hand, any closed integer $\boldsymbol{x}$ will be converted to a real number by .d; even an open integer will be converted to an angle by any angular conversion (cf. p. 78).

Your WP43 provides two kinds of integers: integers of finite length (called short) and of almost infinite length (called long).

Long integers are useful e.g. for numeric tasks. If you enter a number of arbitrary length just without using $\square, E$, or $[C C$, it is taken as a long integer of base 10 by your WP43. For example,

## 1111111111 EXP $\mathbf{x}^{2}$ returns <br> 1234567900987654321

Note the number is adjusted to the right again when closed, though no point (or comma) is displayed. Such a 19-digit result is shown with ease. Big long integers ( $\geq 10^{21}$ ) will be displayed using a smaller font. Very big ones ( $\geq 10^{43}$ ) will be shown with an exponent instead of their least significant digits; nevertheless, all their digits are kept internally, thus long integers can be of very high precision (up to 1000 digits).

> Example (a mathematical problem solved in 2019 only):
> Integers $\boldsymbol{n}$ from 1 up to 100 could all ${ }^{159}$ be expressed as sum of three cubes $\boldsymbol{n}=\boldsymbol{i}^{3}+\boldsymbol{j}^{3}+\boldsymbol{k}^{3}$ - except 42 (sic!). In September 2019, two mathematicians of Bristol and Boston published that the numbers 12602123297335 631, 80435758145817515 , and -80538738812075974 should solve this problem. ${ }^{160}$ Verify!

[^71]
# 12602123297335631 EXP $x^{3}$ returns $2001387454481788542313426390100466780 \times 10^{12}$ 

SHOW shows all digits of this number:
2001387454481788542313426390100466780457779
044591
$80435758145817515 x^{3} \oplus$ (SHOW returns
5224135990369791502809661448536532471497643
62110466
$-80538738812075974 x^{3} \oplus$ returns

This is all you need to know about entering and displaying long integers go to $p .157$ for further information about calculating with them.

Short integers feature a finite word size (user settable up to 64 bits) and are especially useful for computer logic and close-to-hardware system design tasks including debugging. ${ }^{161}$ A short integer of a certain base (2, $3, \ldots, 16$ ) is entered specifying its digits trailed by \#base. You may abbreviate \# 10 by \# (D) and \# 16 by \# (H).

Open INTS (see its top view here) for direct access to the digits $\mathbf{A} .$. $F$ required for numeric input in bases $>10$. From the $2^{\text {nd }}$ integer input on,


[^72]you can save keystrokes: If you enter a new number omitting \#, $\square$, and [E], your WP43 takes it as a short integer of the base you specified last before - as long as you did not enter any other $D T$ in between. ${ }^{162}$

Word size and integer sign mode (ISM) set are indicated in the status bar using a format ww : x. Therein, ww denotes the word size in bits, and x is 1 or $\mathbf{2}$ for 1 's or 2's complement, u for unsigned, or $\mathbf{s}$ for sign-andmantissa mode (cf. p. 80); these ISM's control the handling of negative numbers (see examples below).

CARRY and OVERFLow - if set - will be shown as ${ }_{c}$ or ${ }^{\circ}$ or ${ }^{\circ}$, respectively, trailing ISM display in the status bar. Both are system flags ${ }^{163}$ (cf. p. 58) - if you want to set, clear, or check them one by one, use the commands provided in FLAG.

## GTO FLAG

## Example:

## Enter (FLAG SF SYS.FL (LEAD.0) (or SF (L))

INTS $\triangle$ WSIZE $1 \mathbf{2}$ This allows seeing all bits at a glance.
147 EXIT
$\triangle \rightarrow$ BIN
1COMPL

Enter 147 as a (decimal) long integer.
Convert 147 to a binary short integer 1 's complement, and you will see ${ }^{164}$
$000010010011_{2}$ and -after (4t)-
$111101101100_{2}$.
Obviously $\dagger$ + $/$ in 1COMPL flips every bit, equivalent to NOT here.
Return to the original number via $\ddagger$,

## $000010010011_{2}$ and - after + +2 - <br> $111101101101_{2}$.

Note the negative number equals the inverse +1 in 2COMPL.

[^73]Return via + $+/$ again, press SIGNMT and you will see $000010010011_{2}$ and -after t⿴囗 - $\quad 100010010011_{2}$.
Negating a number will just flip the top bit in SIGNMT (hence its name).
Return via ${ }^{+/ L}$ once more, press UNSIGN and you will get

## $000010010011_{2}$ and -after tro - $111101101101_{2}^{2}$.

Note the $2^{\text {nd }}$ number looks like in 2COMPL but an overflow is indicated here - see ${ }^{\mathbf{0}}$ in the status bar trailing the ISM. ${ }^{165}$ Thus, pressing $\dagger^{+/ L}$ will not suffice for returning to the original number here anymore; you must clear OVERFLow explicitly by (FLAG CF SYS.FL (OVERFL) (or (CF) (B).

As seen, positive numbers stay unchanged in all these four modes. Negative short integers, on the other hand, are displayed in different ways. Therefore, taking a negative short integer in one ISM and switching to another one will lead to different interpretations.

## Example (continued):

The fixed bit pattern representing

$$
\begin{aligned}
& -147_{10} \text { in } 12: 2 \text { will be displayed as... } \\
& -146_{10} \text { in } 12: 1 \text {, as... } \\
& -1901_{10} \text { in } 12: \mathrm{s} \text {, and as... } \\
& 3949_{10} \text { in } 12: \mathrm{u} \text {. You can verify this easily. }
\end{aligned}
$$

Keeping ISM and changing bases will produce different views on the constant bit pattern as well.

[^74]
## Example (continued):

## Compare the outputs for different bases in 16:2

| -16123 EXIT | -16 123 to base 10 corresponds to... |
| :---: | :---: |
| (\#) [2] | $1100000100000101_{2}$ to base 2, |
| (\#) 3 | -211010011 ${ }_{3}$ to base 3, |
| (\#4) | $30010011_{4}$ to base 4, |
| (\#) 5 | -1 $003443_{5}$ to base 5, |
| (\#) 6 | -2023516 to base 6, |
| (\#) 7 | -650027 to base 7 , |
| \# 8 | $140405_{8}$ to base 8, and |
| (\#) 9 | -24 104 g to base 9 . |

You may have noticed that the displays for bases 2, 4, and 8 look similar, presenting all 16 bits to you, while a signed mantissa is displayed for the other bases instead. There are also different separator intervals (four bits for binary, two digits for bases 4, 8, and 16, and three for all other bases); they are fixed unless GAP 0 is set by you.

These different display formats take into account that bases 2, 4, 8, and 16 are most convenient for bit and byte manipulations and further close-to-hardware applications. The bases in between will probably gain most interest in dealing with different number representations and calculating therein, where base 10 is the common reference standard. ${ }^{166}$

Let's look at bigger words now: ${ }^{167}$

## Example (continued):

Enter (CF) (L)
INTS $\triangle$ WSIZE 64

## UNSIGN

[^75]${ }^{167}$ Note up to 16 hexadecimal digits are possible for unsigned short integers.

## - 93 A 14 C 6 \# (cf. the menu INTS shown on p. 150).

Then your WP43 will display

## 93 14 C6 $_{15}$

In binary representation, this number will need 28 digits and would look like

## $1001001110100001010011000110_{2}$.

Obviously, your WP43 cannot display such a big binary number in a single row this way. Look what it does instead - enter (\# (2) for converting $\boldsymbol{x}$ to binary and you will see:
$1001001110100001010011000110_{2}$
This binary number is displayed using the small font provided. If leading zeros were turned on via (SF (L) all 64 bits will be displayed in one row making use of a minimal font:

lool olll loles ool alos $1 \mathrm{llos} . \mathrm{ell}_{2}$

$\ldots$ with the 36 most significant bits all containing $0 .{ }^{168}$

## Integers: Bitwise Operations on Short Integers

As mentioned, your WP43 carries all the bitwise operations you may know from vintage HP-16C Computer Scientist, plus some more you may have learned with WP 34S. You find them all in BIT. Generally, bits in a word are counted from right to left, starting with number 0 for the least significant bit. This convention is important for specifying correct bit numbers in the

## \# BIT EXP

 operations BC?, BS?, CB, FB, and SB.The examples in this chapter all deal with 8 -bit words showing leading zeros for easy reading:

```
SF) (L)
BIT \ WSIZE }
```

[^76]Let's look at the bitwise dyadic functions (all found in the $1^{\text {st }}$ view of BIT):

| SB | BS? | \#B | FB | BC? | св |
| :---: | :---: | :---: | :---: | :---: | :---: |
| NAND | NOR | XNOR | MIRROR |  |  |
| AND | OR | XOR | NOT | MASKL | MASK |


| Common <br> input | $\mathbf{Y}$ | $01101011_{2}$ |
| ---: | :---: | :---: |
|  | $\mathbf{X}$ | $10111001_{2}$ |
| Operation | Symbol | Output |
| AND | $\wedge$ | $00101001_{2}$ |
| NAND | $\bar{\wedge}$ | $11010110_{2}$ |
| OR | $\vee$ | $11111011_{2}$ |
| NOR | $\bar{v}$ | $00000100_{2}$ |
| XOR | $\underline{v}$ | $11010010_{2}$ |
| XNOR |  | $00101101_{2}$ |

Find seven shift and rotate functions in the table below (all from the $2^{\text {nd }}$ view of BIT) with schematic pictures here like they were printed on the backplane of the HP-16C (the boxed C represents the carry bit, indicated in the status bar if set).

Common input $10110011_{2}$

| Operation | Schematic picture if applicable | E. g. | Output |
| ---: | :---: | :---: | :---: |
| Clear Bit |  | CB 4 | $10100011_{2}$ |
| Flip Bit |  | FB 5 | $10010011_{2}$ |
| Set Bit |  | SB 6 | $11110011_{2}$ |
| Negate $^{169}$ |  | NOT (ᄀ) | $01001100_{2}$ |
| Mirror |  | MIRROR | $11001101_{2}$ |



| Operation | Schematic picture if applicable | E. g. | Output |
| :---: | :---: | :---: | :---: |
| Rotate Left | C | $\begin{aligned} & \text { RL } 1 \\ & \text { RL } 2 \end{aligned}$ | $\begin{aligned} & 01100111_{2} \\ & 11001110_{2} \end{aligned}$ |
| Rotate Left through Carry | Cr $\longmapsto$ | $\begin{aligned} & \text { RLC } 1 \\ & \text { RLC } 2 \end{aligned}$ | $\begin{aligned} & 01100110_{2} \\ & 11001101_{2} \end{aligned}$ |
| Rotate Right |  | $\begin{aligned} & \text { RR } 1 \\ & \text { RR } 2 \\ & \text { RR } 3 \end{aligned}$ | $\begin{aligned} & 11011001_{2} \\ & 11101100_{2} \\ & 01110110_{2} \end{aligned}$ |
| Rotate Right through Carry | $\xrightarrow{\square}$ | RRC 1 <br> RRC 2 <br> RRC 3 | $\begin{aligned} & 01011001_{2} \\ & 10101100_{2} \\ & 11010110_{2} \end{aligned}$ |
| Shift Left | C5 $\longleftarrow \sim 0$ | $\begin{aligned} & \text { SL } 1 \\ & \text { SL } 2 \end{aligned}$ | $\begin{aligned} & 01100110_{2} \\ & 11001100_{2} \end{aligned}$ |
| Shift Right | $\longrightarrow \longrightarrow C$ | $\begin{aligned} & \text { SR } 1 \\ & \text { SR } 2 \end{aligned}$ | $\begin{array}{ll} 01011001_{2} \\ 00101100_{2} \end{array}$ |
| Arithmetic Shift Right |  | ASR 3 | $\begin{aligned} & \text { in 1/2COMPL: } \\ & 11110110_{2} \\ & \text { in UNSIGN: }{ }^{170} \\ & 0001010_{2} \\ & \text { in SIGNMT: } \\ & 10000110_{2} \end{aligned}$ |

See the IOI for these and further commands operating on bit level on short integers (LJ and RJ, MASKL and MASKR, \#B, and the tests BS? and $B C ?)$.

[^77]Finally, note that no such operation will set OVERFL. CARRY is only settable by shift or rotate functions as demonstrated above. And ASR is the only bitwise operation being sensitive to ISM - ASR is the link to integer arithmetic operations.

## Integers: Arithmetic Operations

Of the four basic arithmetic operations (,,$+- \times$, and / ), the first three work with both kinds of integers as they do with reals; the only difference lies in precision: up to 64 digits precision for short integers in binary representation on your WP43 or even almost infinite precision for long integers. Take $\dagger$ as a multiplication times -1 , and $y^{x}$ as repeated multiplication. Depending on input parameters and mode settings, OVERFL or CARRY may be set in such an operation (see pp. 160ff).

Divisions, however, must be handled differently in integer domain since the quotient cannot feature a fractional part here. Generally, the formula

$$
\frac{a}{b}=(a \operatorname{div} b)+\frac{1}{b} \times \operatorname{rmd}(a ; b)
$$

applies; therein, the horizontal bar denotes real division, div represents integer division, and rmd stands for the remainder of the latter. While remainders for positive parameters are simply found, remainders for negative dividends or divisors may lead to confusion sometimes. The formula above, however, is easily employed for calculating such remainders (also for reals - see the $1^{\text {st }}$ row of examples here):

$$
\begin{array}{lll}
\frac{25}{7}=3+\frac{1}{7} \times 4 & \left(\text { and for a real case: } \frac{25}{7.5}=3+\frac{1}{7.5} \times 2.5\right) \\
\frac{-25}{7}=-3+\frac{1}{7} \times(-4) & \Rightarrow & \operatorname{rmd}(-25 ; 7)=-4 \\
\frac{25}{-7}=-3+\frac{1}{-7} \times 4 & \Rightarrow & r m d(25 ;-7)=4 \\
\frac{-25}{-7}=3+\frac{1}{-7} \times(-4) & \Rightarrow & \\
r m d(-25 ;-7)=-4
\end{array}
$$

In general, $\operatorname{rmd}(a ; b):=a-b \times(a \operatorname{div} b)$ applies.

Unfortunately, there is a $2^{\text {nd }}$ function doing almost the same: it is called mod. With the same pairs of numbers as above, it returns:

$$
\begin{aligned}
& \bmod (25 ; 7)=4 \\
& \bmod (-25 ; 7)=3 \\
& \bmod (25 ;-7)=-3 \\
& \bmod (-25 ;-7)=-4
\end{aligned}
$$

So mod (read modulo) returns the same as rmd only if both parameters have equal signs. The general formula for mod reads:

$$
\begin{array}{ll}
\bmod (a ; b):=a-b \times \text { floor }\left(\frac{a}{b}\right) \quad \text { with e.g. floor }\left(\frac{25}{7}\right)=3 \text { and } \\
& \text { floor }\left(-\frac{25}{7}\right)=-4
\end{array}
$$

Also this formula applies to reals as well. Use it straightforwardly for calculating e.g.

$$
\bmod (25.3 ;-7.5)=25.3-(-7.5) \cdot(-4)=-4.7
$$



These four functions are called IDIV, RMD, MOD, and FLOOR on your WP43 for obvious reasons. They are found in INTS (cf. p. 150), together with more integer operations like CEIL, ${ }^{171} \times$ MOD, and ${ }^{\wedge}$ MOD (see p. 163 for an example); (MOD is also printed on the keyboard as -shifted function of $\square$.

Furthermore, many exponential and logarithmic operations, $x^{2}$ and $\sqrt{x}, x^{3}$ and $\sqrt[3]{x}, x!$, COMB and PERM, as well as SIN, COS, and TAN operate on integers, too. Note that some of these functions will stay in integer domain while others may or will return real or even complex numbers. See the summary on pp. 162ff for further information.

[^78]
## Integers: Overflow and Carry with Short Integers

OVERFL and/or CARRY may be touched in arithmetic operations on short integers on your WP43 under certain conditions. Note that for each word size and ISM there is a maximum and a minimum integer displayable let's call them $I_{\max }$ and $I_{\text {min }}$.

## Example:

For 4-bit words (i.e. WSIZE 4), we get

$$
\begin{array}{llll}
\text { - } I_{\max }=15 & \text { and } & I_{\min }=0 & \text { for } \mathbf{4 : \mathbf { U } ,} \text {, while } \\
\text { - } I_{\max }=7 & \text { and } & I_{\min }=-8 & \text { for } \mathbf{4 : 2}, \\
\text { - } I_{\max }=7 & \text { and } & I_{\min }=-7 & \text { for } \mathbf{4 : 1} \text { and } \mathbf{4 : s . s .}
\end{array}
$$

Let's start from 1 incrementing by 1 and see what will happen in these various modes. And whenever OVERFL and/or CARRY will be lit in the status bar in this course, we will clear them (using
(CF SYS.FL (OVERFL) (or CF (B) and/or
CF SYS.FL (CARRY) (or CF (C) before next increment:

| 4:U |  | 4:2 |  | 4:1 |  | 4:s |  |
| :--- | :---: | :--- | :---: | :--- | :--- | :--- | :--- |
| $0001_{2}$ | 1 | $0001_{2}$ | 1 | $0001_{2}$ | 1 | $0001_{2}$ | 1 |
| $0010_{2}$ | 2 | $0010_{2}$ | 2 | $0010_{2}$ | 2 | $0010_{2}$ | 2 |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| $0111_{2}$ | 7 | $0111_{2}$ | 7 | $0111_{2}$ | 7 | $0111_{2}$ | 7 |
| $1000_{2}$ | 8 | $1000_{2} 0$ | -8 | $1000_{2} 0$ | -7 | $1000_{2} \mathbf{\circ}$ | -0 |
| $1001_{2}$ | 9 | $1001_{2}$ | -7 | $1001_{2}$ | -6 | $0001_{2}$ | 1 |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $0010_{2}$ | 2 |
| $1110_{2}$ | 14 | $1110_{2}$ | -2 | $1110_{2}$ | -1 |  |  |
| $1111_{2}$ | 15 | $1111_{2}$ | -1 | $1111_{2}$ | -0 |  |  |
| $0000_{2} \mathbf{\circ}$ | 0 | $0000_{2} \mathbf{c}$ | 0 | $0001_{2} \mathbf{c}$ | 1 |  |  |
| $0001_{2}$ | 1 | $0001_{2}$ | 1 | $0010_{2}$ | 2 |  |  |

For comparison, we start another turn from 1 following the same rules but decrementing by 1 instead:

| 4:U |  | 4:2 |  | 4:1 |  | 4:5 |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $0001{ }_{2}$ | 1 | $0001{ }_{2}$ | 1 | $0001{ }_{2}$ | 1 | $0001{ }_{2}$ | 1 |
| $000 \mathrm{O}_{2}$ | 0 | 00002 | 0 | $0000{ }_{2}$ | 0 | $0000_{2}$ | 0 |
| $1111_{2}{ }^{\text {c }}$ | 15 | $1111_{2}$ c | -1 | $1110_{2}$ c | -1 | $1001_{2}$ c | -1 |
| $1110_{2}$ | 14 | $1110_{2}$ | -2 | $1101{ }_{2}$ | -2 | $1010_{2}$ | -2 |
| 10012 | 9 | $1001{ }_{2}$ | -7 | $1000{ }_{2}$ | -7 | $1111_{2}$ | -7 |
| $1000_{2}$ | 8 | $1000_{2}$ | -8 | $0111_{2}{ }^{\text {o }}$ | 7 | $1000{ }^{\text {a }}$ - | -0 |
| 01112 | 7 | $0111_{2}{ }^{\text {o }}$ | 7 | $0110_{2}$ | 6 | $1001{ }_{2}$ | -1 |
| $0110_{2}$ | 6 | 0110 ${ }_{2}$ | 6 | $\ldots$ | $\ldots$ | $1010_{2}$ | -2 |
| $\cdots$ | $\ldots$ | $\cdots$ | $\ldots$ | $\cdots$ | $\ldots$ |  |  |
| $0010_{2}$ | 2 | $0010_{2}$ | 2 | 00012 | 1 |  |  |
| 00012 | 1 | 00012 | 1 | $\mathrm{OOOO}_{2}$ | 0 |  |  |

The most significant bit is \#2 in 4:s and \#3 in all other modes here.

With these results, $I_{\max }$, and $I_{\text {min }}$, the general rules for setting and clearing CARRY and OVERFL are as presented in the table following:

| Operation | Effect on CARRY | Effect on OVERFL |
| :---: | :--- | :--- |
| Shift and rotate | As demonstrated on pp. 155f. | None. |
| Boole's, MIRROR | None (cf. pp. 155f). | None. |
| $\|\mathrm{x}\|$, ABS | None. | Clears OVERFL <br> (but sets it for $x=I_{\text {min }}$ <br> in 2COMPL). |
| ,+ RCL+, STO+ <br> INC, etc. | Sets CARRY if there is a carry out <br> of the most significant bit, <br> else clears CARRY. | Sets OVERFL if the re- <br> sult exceeds $\left[I_{\text {min }} ; I_{\text {max }}\right]$ <br> else clears OVERL. |


| Operation | Effect on CARRY | Effect on OVERFL |
| :---: | :---: | :---: |
| $\begin{gathered} \text {-, RCL-, STO-, } \\ \text { DEC, etc. } \end{gathered}$ | Sets CARRY in a subtraction $\boldsymbol{m}-\boldsymbol{s}$ <br> - in 1COMPL or 2COMPL if the binary subtraction causes a borrow ${ }^{172}$ into the most significant bit, <br> - in UNSIGN if $\boldsymbol{m}<\boldsymbol{s}$, <br> - in SIGNMT if $\boldsymbol{m}<\boldsymbol{s} \underline{\&} \boldsymbol{m} \cdot \boldsymbol{s}>0$ <br> Else clears CARRY. | Sets OVERFL if the result exceeds $\left[I_{\text {min }} ; I_{\text {max }}\right]$, else clears OVERFL. <br> Thus, in UNSIGN, +1/ always sets OVERFL and $(-1)^{x}$ does so for odd $\boldsymbol{x}$. |
| $\begin{aligned} & \times, \text { RCL× } \times \text { STO× } \\ & \text { +/-, }(-1)^{\mathrm{x}}, x^{2}, x^{3} \\ & \text { LCM, } x!\text {, etc. } \end{aligned}$ | None. |  |
| $2^{\text {x }}$ | Clears CARRY. <br> Sets CARRY only if $x=-1$, or in UNSIGN if $x=w s i z e$ or in the other modes if $\boldsymbol{x}=$ wsize -1 |  |
| $\mathrm{y}^{\mathrm{x}}, 10^{\mathrm{x}}$ | Sets CARRY for $\boldsymbol{x}<0$ (as well as for $0^{0}$ ), else clears CARRY. | Sets OVERFL if the result exceeds [ $\left.I_{\text {min }} ; I_{\text {max }}\right]$, else clears OVERFL. |
| $e^{x}$ | Sets CARRY for $\boldsymbol{x} \neq 0$, else clears CARRY. |  |
| DBL× | None. | Clears OVERFL. |
| $\begin{aligned} & \text { /, RCL / }, \\ & \mathrm{STO} /, \mathrm{DBL}^{2} \text {, } \\ & \mathrm{LN}, \mathrm{LOG}_{10} \\ & \mathrm{LOG}_{2}, \mathrm{LOG}_{\mathrm{x}} \mathrm{y} \end{aligned}$ | Sets CARRY if the remainder is $\neq 0$, else clears CARRY. | Clears OVERFL (but sets it in 2COMPL for the division $I_{\text {min }} /(-1)$ ). |

172 See the examples on previous page.
TN: The so-called borrow in subtraction seems to be a specialty of the USA. Compare the subtle methodic differences in manual subtracting explained in http://de.wikipedia.org/wiki/Subtraktion\#Schriftliche Subtraktion (the corresponding article in the English Wikipedia is significantly less instructive).
TNG: Sowohl carry wie borrow entsprechen einem Übertrag.

## Integers: Summary of Functions

Many of the numeric functions operating on reals also work for integers. Functions operating on short integers will return DT 10 except for the cases given on pp. 75ff. Functions operating on long integers, on the other hand, may return results of $D T 1,2,3$, or 4 :

- General mathematics:
- Monadic functions:
$\sqrt{\boldsymbol{x}},(\sqrt[3]{x}),(l b x)$, and $(1 g)$ return long integers if possible (else real or complex numbers) for long integer input ${ }^{173}$ or just the integer part of the solution for short integer input;
$\boldsymbol{e}^{\boldsymbol{x}}$ and In operating on long integers return real or complex numbers always; for short integers, they act in analogy to $\sqrt{\boldsymbol{x}}$;
\# converts a closed integer to another base; [ $\rightarrow$ BIN], [ $\rightarrow 0 C T$, $[\rightarrow D E C]$, and $[\rightarrow H E X]$ cover the most popular bases.
[|x|), $\left[\mathbf{x}^{2}\right],\left[\mathbf{x}^{3}\right],\left[2^{\mathbf{x}}\right]$, and $\left[10^{x}\right]$ return integers as you expect, and
†/ works for short integers as demonstrated on pp. 151f. For long integers, it works as for reals.
SHOW displays all ( $\mathbf{\leq} 296$ ) digits of a long integer $\boldsymbol{x}$ until next keystroke. For even greater long integers, see the IOI.
- Dyadic functions:
$\oplus, \square$, and $\boldsymbol{x}$ return integers as expected (cf. p. 72),
(1) returns an integer or a real as specified on p. 76,
(IDIV) returns just the integer part of the division,
(RMD) returns the remainder of $\boldsymbol{y} / \boldsymbol{x}$ (cf. pp. 157f for examples), (IDIVR) combines IDIV and RMD,
MOD returns $\boldsymbol{y} \bmod \boldsymbol{x}$ (cf. p. 158 for examples),
$y^{x}$ works as specified on p. 77,
$\sqrt[x]{y})$ and $\left(\log _{x} y\right)$ return results in analogy (see pp. 77f); ${ }^{173}$
[max] (or (min)) return the maximum (or minimum) of $\boldsymbol{x}$ and $\boldsymbol{y}$,

[^79](GCD) returns the Greatest Common Divisor of $\boldsymbol{x}$ and $\boldsymbol{y}$ and (LCM) their Least Common Multiple.

- Triadic functions:
[ $\times$ MOD] returns $(z \times y) \bmod x$ for $x>1, y>0, z>0$, and [ MOD ] returns $\left(z^{y}\right) \bmod x$ for $x>1, y>0, z>0$ (e.g. 73 ENTERT 55 ENTERT 31 INTS ${ }^{\text {EMOD }}$ returns 26).
- Boole's algebra:
(AND), (NAND), (OR], (NOR), (XOR), (XNOR), and (NOT) operate bitwise on short integers as shown on p. 155. They operate on long integers like they do in the HP-28S, cf. p. 131.
- Bitwise operations are for short integers exclusively:
(CB), (FB), (SB), (ASR), (SL), (SR), (RL), (RLC), (RR), (RRC), and (MIRROR) work as demonstrated on pp. 154ff.
[LJ] (or (RJ)) justifies the bit pattern to the left (or right) within its word size,
(MASKL) and (MASKR) create mask words,
[ $B C$ ? ) and [ $B S$ ?] test if the specified bit is clear or set,
(\#B] counts the number of bits set in $\boldsymbol{x}$.
- Probability (cf. pp. 102f):
x! returns the factorial,
$\left[C_{y x}\right]\left(\left(P_{y x}\right)\right)$ computes the number of combinations (permutations),
(RANI\#) returns a (pseudo) random integer number $\in[\boldsymbol{x}, \boldsymbol{y}]$.
- Advanced mathematics (see the ReM, App. I): $\left[\mathbf{B}_{\mathrm{n}}\right]$ and $\left[\mathbf{B}_{\mathrm{n}}{ }^{*}\right]$ return the Bernoulli numbers, (FIB) the Fibonacci number, and (NEXTP) the next prime number greater than $\boldsymbol{x}$.


Many more functions accept integer input but will return different, mostly real output. See the IOI and ReMS 3 for more about them.

## Complex Numbers: Introduction

So far, we dealt with reals only (mathematically, fractions and integers are mere subsets of reals). Your WP43 can do more for you. Mathematicians can deal with more complex numbers than reals; these are called complex numbers. If you do not know of them, feel free to skip the next three chapters; you can profit from your WP43 perfectly without them.

If you know of complex numbers, however, note your WP43 supports most functions operating on reals also for complex numbers.

Complex results in calculations: As long as you put in just reals, you will get only real results in startup default configuration; to allow for complex results, set CPXRES (or flag (D). Try $1+/ \sqrt{\boldsymbol{x}}$ and see the different returns.

Use CC to enter a complex number directly (cf. p. 73). With startup default settings, CC separates and concatenates the real and imaginary part in numeric input.

## $R \uparrow C P X$ <br> $|x| \measuredangle$ 

Examples (with startup default settings):
$3+i \times 4$ is keyed in 3 CC (ENTERT while the display (set e.g. to FIX 5) shows in lowest numeric row:

3
CC
4
ENTERT

## 3.

3. $+\mathrm{i} \times$
4. $+i \times 4$
$3.00000+i \times 4.00000$

You enter the real part first - CCC closes it - the imaginary part second as you write the number. ${ }^{174}$ Input of negative complex numbers works in full analogy to real number input (cf. p. 25). Following our example above,

[^80]\[

$$
\begin{aligned}
& 3-i \times 4 \text { is keyed in } 3 \text { CC 4 }+ \text { ENTER } \boldsymbol{4} \text {, } \\
& -3+i \times 4 \quad \text { is keyed in } \\
& -31-i \times 42 \quad \text { is keyed in } \\
& \text { Alternatively, use } 3 \text { CC (4) } 2 \text { ENTERT }+/ \rightarrow \text { here. }
\end{aligned}
$$
\]

Choosing scientific notation, e.g. SCI 5, this last number will be displayed like

## $-3.10000 \times 10^{1}-\mathrm{i} \times 4.20000 \times 10^{1}$

Depending on display formatting, this may be shown more compact (allowing for SCI 6):

## $-3,100000 \cdot 10^{1}-\mathrm{i} \cdot 4,200000 \cdot 10^{1}$

With a complex number entered or at least one complex input parameter in arithmetic operations or function calls, your WP43 will set CPXRES automatically (indicated by $\mathbb{C}$ in the status bar).

Instead of rectangular notation, a complex number may be written in polar notation as well. Polar notation is set by SF SYS.FL (POLAR) (or (SF (X), causing $\odot$ lit in the status bar. Then, for a new complex number, its magnitude (or radius) $\boldsymbol{r}$ shall be entered first and its phase (or argument) $\boldsymbol{\vartheta}$ second. $\vartheta$ may be entered in any angular display mode (cf. p. 139); though often radians or multiples of $\pi$ will make most sense here.


## Example:

Set polar notation and MODE RAD. Then a complex number $\left(5 ; 1.2^{r}\right)$ is keyed in (5) COC(2) ENTERT while the display (set e.g. to FIX 2) shows in lowest numeric row successively:

| (5) | 5 |  |
| :---: | :---: | :---: |
| (cC) | 5. 4 |  |
| (1) (2) | 5. ¢ 1.2 |  |
| ENTERT |  | $5.00 \nless 1.20^{r}$ |

[^81]Special cases: If a negative magnitude is entered, it is made positive and $\boldsymbol{\vartheta}$ is increased by $\pi$ and then normalized (i.e. $\boldsymbol{\vartheta}$ will stay in the interval $(-\pi, \pi]-$ cf. p. 143; greater phase input is legal, but the output will be normalized always). If $\mathbf{0}$ is entered for $\boldsymbol{r}$ then $\boldsymbol{\vartheta}$ will be set to 0 as well.

Composing and decomposing a complex number: Instead of entering a complex number directly, you can generate it from two closed reals provided in $\mathbf{X}$ and $\mathbf{Y}$. If $h_{L}$ is lit, CC will take $\boldsymbol{y}$ as real part and $\boldsymbol{x}$ as imaginary part composing the complex number. If $\odot$ is lit, CC will take $\boldsymbol{y}$ as magnitude and $\boldsymbol{x}$ as phase of the new complex number (compare numeric input above).

## Example:

MODE DEG
These three entries return to startup default settings.
CF X
CF (1)
4 ENTERT - 3 EXIT

Note EXIT closes input without disturbing the stack.
(CC) composes a complex number out of $x$ and $y$ now, lights $\mathbb{C}$ and returns ${ }^{175}$

$$
\text { 4. }-i \times 3 .
$$

SF $X$ turns $\boldsymbol{L}$ to $\odot$ and displays 5. $4-36.86989764584402^{\circ}$

Vice versa, CC may also cut a closed complex number $\boldsymbol{x}$ into two reals in $\mathbf{X}$ and $\mathbf{Y}$ following the same rules.

## Example (continued):

(CC) returns $\quad$| $r=$ | 5. |
| :--- | :--- |
|  | $\ddots=$ |
|  | $-36.86989764584402^{\circ}$ |

$\mathfrak{c} \odot$ remain lit in the status bar.

[^82]RAD CC returns
CF $\mathbb{X}$ turns $\odot$ to $\boldsymbol{L}$ and shows

## 5. $4-6.435011087933 \times 10^{-1 r}$

 4. $-\mathrm{i} \times 3$.$$
\begin{aligned}
& \mathrm{Re}= \\
& \mathrm{Im}=
\end{aligned}
$$

## $\mathbb{C} \boldsymbol{4}^{\mathbf{r}}$ remain lit in the status bar.

Generally, complex outputs follow real number formatting (cf. pp. 84ff). The number of displayable decimals, however, may be limited by screen space. If you want to see both parts of a complex number in higher precision, either press (CC), watch, and press (CC again or call SHOW. ${ }^{176}$ In calculations, press ENTERT for separating complex number input as you do in real domain.

## Example (with startup default settings):

$(\mathbf{1}+\mathbf{2 i}) \times(\mathbf{3 + 4 i})$ is entered and solved like this:

1 CC 2 ENTERT 1. $+\mathrm{i} \times 2$.

3 (CC 4
区
returning

Note that with a complex $\boldsymbol{x}$ closed and POLAR, ...

- +/ will change the signs of both the real and the imaginary part (as shown above),
- CPX conj will change the sign of the imaginary part only, and
- CPX Re§Im will swap real and imaginary parts.

[^83]SHOW will show both parts of a complex number with 34-digit precision each.

Also many transcendental functions will operate on complex numbers (e.g. $\sin , \cos , \tan , \ln , \mathrm{e}^{x}, \boldsymbol{y}^{x}, \sqrt{\boldsymbol{x}}$, etc.). Please check pp. 170f.

## Complex Numbers Used for 2D Vector Algebra

You can use complex domain for $2 D$ vector algebra as demonstrated below. The functions $|x|,+,-$, CROSS, DOT, and UNITV wait for you see the contents of $\underline{C P X}$ and the $I O I$.
Speed and Heading

| Vector |
| :--- |

Wind Vector 25 True Vector

Two complex examples more (from the HP-42S OM):


## Dot Product of Complex Numbers

The figure ... represents three 2D force vectors. Use complex numbers and add the three vectors. Then use the DOT (dot product) function to find the component of the resulting vector along the $175^{\circ}$ line.

## Solution:

## DISP FIX 2

## MODE DEG

SF X
170 CC 143 ENTERT
185 CC 62
$\oplus$
100 (CC) 261


Now, take the unit vector at $175^{\circ}$ :

## 1 (CC) 175 1. 女 175

and multiply

## CPX dot

he resulting vector sum has a component of approximately 79 N in the direction of $175^{\circ}$.

## Computing Moments.

To compute the moment of two vectors, use the CROSS (cross product) function. The cross product of two vectors is a third orthogonal vector. However, when two complex numbers are crossed, the WP43 simply returns a real number that is equal to the signed magnitude of the resulting moment vector.

Find the moment generated by the force acting through the lever in the illustration overleaf, where $\vec{M}=\vec{r} \times \vec{F}$


Note this sketch shows a $2 D$ situation, i.e. lever and force are both acting in the drawing plane.

Solution (continued with the settings of previous example):
Key in the radius vector and the force vector:

| 5 (CC) 50 ENTERT | $5.00 \Varangle 50.00^{\circ}$ |  |
| :---: | :---: | :---: |
| 300 (CC) 205 | 300. ¢ 205 |  |
| CPX) cross returns $y \times x$ |  | 633.93 |

The moment vector has a magnitude of 634 pounds times inches and, since the result is positive, the vector points up, perpendicular to the plane of this page. ${ }^{177}$

## Complex Numbers: Summary of Functions

Many of the functions operating on reals also work for complex numbers:

- General mathematics:
- Monadic functions:
$\sqrt{x}$ and $\left[x^{2}\right],[\sqrt[3]{x}]$ and $\left[x^{3}\right],\left[\sqrt{\left(1+x^{2}\right)}\right],\left[2^{x}\right]$ and $(l b x), 10^{x}$ and $[1 g)$,
 (tanh), and their inverses work as usual (note that (arcsin) etc. returning complex results will display them in radians but not tagged),

[^84]$\left[e^{x}-1\right]$ and $[\ln (1+x)]$ return more accurate results with $x \approx 0$,
+/. returns $\boldsymbol{x} \times(-1)$ (a.k.a. 'unary minus') for closed input and POLAR, while it turns $\boldsymbol{x}$ by $180^{\circ}$ for POLAR, and
$\left[(-1)^{x}\right]$ returns $\cos (\pi x)+i \sin (\pi x)$ for non-integer $\boldsymbol{x}$.

- Dyadic functions:
$\pm, \boxed{\square}, \boxed{x},\left(1, \sqrt[y^{x}]{ }\right.$ and $[\sqrt[x]{y}]$ work as usual, ${ }^{178}$
$\left(\log _{x} y\right)$ calculates the logarithm of $\boldsymbol{y}$ for base $\boldsymbol{x}$, [dot] and [cross] allow using complex numbers for $2 D$ vector computations, and (II) returns $(1 / x+1 / y)^{-1}$ for $x \times y \neq 0$ and 0 else.
- Isolating and manipulating parts of complex numbers:

Use CC for composing and cutting,
(Re) for isolating the real part of $\boldsymbol{x}$ and (Im) for its imaginary part,
( Re I Im ) for swapping its real and imaginary part,
for the magnitude of $\boldsymbol{x}$ and $\measuredangle$ for its phase (a.k.a. argument),
(FP) ( (IP)) for the fractional (or integer) part of $\boldsymbol{x}$ (preserving its sign); (UNITV) returns the unit vector of $\boldsymbol{x}$, and (conj) returns its complex conjugate.

- Rounding:
(RDP] $\boldsymbol{n}$ rounds $\boldsymbol{x}$ to $\boldsymbol{n}$ decimal places in FIX format (e.g. 0.023456789 RDP 5 will return 0.02346 , cf. p. 130),
(ROUND) rounds $\boldsymbol{x}$ using the current display format, ${ }^{179}$
(RSD] $\boldsymbol{n}$ rounds $\boldsymbol{x}$ to $\boldsymbol{n}$ significant digits, and

[^85]SHOW displays all 34 digits of each part of $\boldsymbol{x}$ until next keystroke.

- Advanced mathematics (see the ReM, App. I for comprehensive information about the functions following):
- Monadic functions:
(FIB) returns the extended Fibonacci number,
$\left[g_{d}\right]$ and $\left[\mathbf{g}_{d}{ }^{-1}\right]$ the Gudermann function and its inverse,
[sinc] returns $\sin (x) / x$ and $[\boldsymbol{\operatorname { s i n }} \boldsymbol{\operatorname { c o s } ]}$ returns $\sin (\pi x) / \pi x$ for $\boldsymbol{x} \neq 0$ and 1 for $\boldsymbol{x}=0$,
( $W_{p}$ ) returns the principal branch of Lambert's $W$, ( $W^{-1}$ ) returns $\boldsymbol{x}$ for given $W_{p}$,
$x!(=\Gamma(x+1))$ and $[\Gamma(x)]$ calculate the complex Gamma function, and $[\ln \Gamma]$ returns its natural logarithm.
- Dyadic functions:
(AGM) returns the arithmetic-geometric mean,
$\left[\mathrm{C}_{\mathbf{y x}}\right.$ ] and $\left[\mathrm{P}_{\mathbf{y x}}\right.$ ] calculate with complex Gamma, $[\beta(x, y)]$ returns Euler's Beta function, and $(\ln \beta)$ its natural logarithm.


## Vectors and Matrices: Introduction and Input

So far, we dealt with just one or two or (seldom) three numbers at once. Your WP43 can do more for you - e.g. manipulate a set of numbers in a column or a row or even in an array of $4,6,8,9,10,12$, or more numbers simultaneously. Such number columns or rows are called vectors and the arrays are called matrices by mathematicians. If you do not know of vectors and matrices yet, feel free to set them aside; your WP43 will serve you perfectly without them.

If you know of them, however, note the function set of your WP43 covers vector operations and also allows for adding, multiplying, inverting, and transposing matrices, as well as for editing and manipulating parts of such matrices. It also provides functions for computing determinants, eigenvalues and eigenvectors, and for solving systems of linear equations. Most of them are collected in MATX.

Generally, we talk of an $\boldsymbol{n} \times \boldsymbol{m}$ matrix if it features $\boldsymbol{n}$ (horizontal) rows and $\boldsymbol{m}$ (vertical) columns. A vector may be regarded as a special matrix featuring one column or one row only.

## MATX X.FN <br> 2 X

## Example:

A vector $\left[\begin{array}{c}4 \\ -5 \\ 6.7\end{array}\right]$ and a matrix $\left[\begin{array}{ccc}-1 & 12 & 7 \\ 25 & 0 & 3\end{array}\right]$ shall be entered subsequently.
The stack shall be clear at beginning.

## Enter DISP FIX 01

## 3 ENTERT 1 (MATX NEW

to initialize the $3 D$ column vector (i.e. a $3 \times 1$ matrix). See the new matrix in $\mathbf{X}$ and the top view of MATX displayed in the menu section:

| [ 0.0 |  |  |  | $\begin{array}{rr}  & 0.0 \\ .0 & 0.0]^{\top} \end{array}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| ENORM | $\mathrm{V}_{4}$ | STOEL | RCLEL | PUTM | GETM |
| dot | cross | UNITV | DIM | INDEX | EDITN |
| NEW | [M] ${ }^{-1}$ | \|M| | [M] ${ }^{\text {T}}$ | SIM EQ | EDIT |

For saving screen space, your WP43 displays each narrow column vector transposed (thus the superscript T trailing it), i.e. in one row instead of one column on the screen. The vector is initialized with all its components being zero. To enter the vector components, press EDIT (i.e. (F6) and the Matrix Editor will appear in the menu section:


Note the $1^{\text {st }}$ element of the vector is displayed inverted now, indicating the position of the edit cursor. This particular element is shown below in the format set (i.e. FIX 1 here). So we need two display rows for $\mathbf{X}$.

Now press 4

$$
\begin{array}{lll}
{\left[\begin{array}{lll}
0.0 & 0.0 & 0.0
\end{array}\right]^{\top}} \\
1 ; 1=4-
\end{array}
$$

Move the cursor to the next element ( $2^{\text {nd }}$ row, $1^{\text {st }}$ column): $\rightarrow$

$$
\begin{array}{ccc}
{\left[\begin{array}{ccc}
4.0 & 0.0 & 0.0
\end{array}\right]^{\top}} \\
2 ; 1=0.0
\end{array}
$$

Continue editing: 5 + $/ \rightarrow \mathbf{~} 6.7$

$$
\left.\begin{array}{lll}
{[4.0-5.0 \quad 0.0}
\end{array}\right]^{\top}
$$

EXIT

| $\left[\begin{array}{lll}4.0 & 5.0 & 6.7\end{array}\right]^{\text {T}}$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| ENORM | $V_{4}$ | STOEL | RCLEL | PUTM | GETM |
| dot | cross | UNITV | DIM | INDEX | EDITN |
| NEW | $[\mathrm{M}]^{-1}$ | \|M| | [M] ${ }^{\text {T}}$ | SIM EQ | EDIT |

Note [EXIT closed the last matrix element, left the Matrix Editor returning to the top view of MATX, closes input for the matrix, and shifts $x$ to the right.

Now, let's initialize the $2 \times 3$ matrix via
2 ENTERT 3 NEW and begin editing once again by

## EDIT



Three numeric rows are required for editing $x$ now.
The $3 \times 1$ matrix ( 3 rows, 1 column) in $\mathbf{Y}$ is the vector we entered just before; note any matrix is displayed in this short form (a.k.a. descriptor, using a $\times$ even for AULTX) in any stack register but $\mathbf{X}$.

Again, all elements of the new matrix comprise zero at the beginning. Its $1^{\text {st }}$ element is displayed inverted as the $1^{\text {st }}$ element of the vector was above. Matrix editing will continue in full analogy:
$1+$

$$
\left[\begin{array}{ccc}
-1.0 & 0.0 & 0.01 \\
0.0 & 0.0 & 0.0
\end{array}\right]
$$

$12 \rightarrow 7 \rightarrow$

$$
\left[\begin{array}{ccc}
-1.0 & 12.0 & 7.00 \\
-0.0 & 0.0 & 0.0 \\
2 ; 1=0.0 & 0
\end{array}\right]
$$

Entering the last $\rightarrow$ moved the cursor from the last $\left(3^{\text {rd }}\right)$ element of row 1 to the $1^{\text {st }}$ element of row 2 . So you can simply continue editing row-wise:

$$
25 \rightarrow \rightarrow 2 \quad\left[\begin{array}{rrr}
-1.0 & 12.0 & 7.0] \\
25.0 & 0.0 & 0.0
\end{array}\right]
$$

Now also this matrix is closed and ready for calculating. Assume you want to multiply it by $2 / 3$ and see three decimals in the result:

DSP 3
$\odot 2 \odot 3$

## [Jx $\perp$ llatrix] <br> [2×3 Matrix]

$$
0 \text { 2/3 }
$$

Press $\boldsymbol{x}$ and you will get all matrix elements multiplied by $2 / 3$ at once:
$\lfloor J \times \perp$ Ilatrix」
$\left[\begin{array}{lll}-0.667 & 8.000 & 4.6671 \\ 16.667 & 0.000 & 1.333\end{array}\right]$
You may store such matrices in any register or variable. So let's store our resulting matrix in $\mathbf{R 0 0}$ - just press STO $\mathbf{O} \mathbf{0}$ for this.

You can also create and fill a matrix directly in a variable (i.e. you do not have to create the matrix on the stack and store it afterwards).

## Example:

Create a square matrix $[M A]=\left[\begin{array}{rr}4 & -3 \\ -2 & 1\end{array}\right]$ and fill it directly.

2 ENTER $\boldsymbol{T}$ DIM $\boldsymbol{\alpha}$ MA ENTER $\mathbb{1}$ creates MA as a $2 \times 2$ matrix.

## EDITN VAR MA

[2×3 Matrix]

$$
\left[\begin{array}{ll}
0.000 & 0.000 \\
0.000 & 0.000
\end{array}\right]
$$

$1 ; 1=0.000$
$4 \rightarrow 3+2 \rightarrow+\square \rightarrow 1$
$[<x J$ riatrix]
[ $4.000-3.000]$
[-2.000 0.000]
$2 ; 2=1$ -

Now, press EXIT and you are done with MA - and the screen looks just as before again:
[J×ı I'Iatrix」


## Vectors and Matrices: Displaying and Editing Larger Objects

Whenever X contains a matrix, your WP43 will try to show it completely (i.e. display all its elements in the format you chose for reals). Objects in higher stack registers will be indicated in a single row (abbreviated if necessary) or will be shifted out of the display window.
If space does not suffice for showing the entire current matrix in the format chosen, your WP43 will switch to the small font automatically.

## Example (continued):

(DSP) 5

If font switching should not suffice for gaining enough screen space, your WP43 will automatically turn to abbreviated SCl 3 format for the elements of the respective matrix. This allows for showing arbitrary real matrices up to $5 \times 4$ entirely. ${ }^{180}$ If a real matrix exceeds five rows, its $5^{\text {th }}$ row is displayed filled with ellipses (...); if it exceeds four columns, its $4^{\text {th }}$ column is shown filled with ellipses.

[^86]
## Example:

Assume a $6 \times 5$ matrix

$$
x=\left[\begin{array}{ccccc}
1.1493 & 2.6 & 18.725 & 3 & 9.2 \\
0.4 & 5.462 & -6 & 95.1 & 51.6 \\
-7.744 & -8.8 & 9.95 & 54.5 & 0.17 \\
74.66 & 0.229 & -0.0934 & 2 & -3.829 \\
33.9 & -79.4 & 3.436 & 9.08 & 4.256 \\
0.0488 & 7 & 5.98 & -0.68 & -22.492
\end{array}\right]
$$

was entered on the present stack and is in $\mathbf{X}$ now. Then the screen will look like this:
$\left.\begin{array}{r}{[2 \times 3} \\ \quad \text { Matr ix] }\end{array}\right]$

Editing such a large matrix will push also $y$ from the screen as long as the Matrix Editor stays open. You can browse the entire matrix regardless of its size always.

For matrices larger than 5 rows and/or 4 columns, the display may vary depending on the cursor position: ellipses may appear on top and bottom, left and right side. A view of $3 \times 3$ real matrix elements including the one selected by the cursor can be seen always at least - this selected element is also displayed below of the matrix in the format you have chosen for reals. Since the indices of this element are shown there as well you always know where you are.

## Example (continued):

Press (MATX EDIT and you will see:

The $1^{\text {st }}$ matrix element is


Now, go to the bottom row of this matrix by pressing $\downarrow$ five times and you will get:


Go to the very last element of this matrix by pressing $\rightarrow$ four times:

$$
\begin{aligned}
& {\left[\begin{array}{llll}
\ldots . . & \ldots & & \ldots \\
\ldots & 9.950 & 5.450 \times 10^{1} & 1.700 \times 10^{-1} \\
\ldots . . & -9.340 \times 10^{-2} & 2.000 & -3.829 \\
\ldots & 3.436 & 9.080 & 4.256 \\
\ldots & 5.980 & -6.800 \times 10^{-1} & -2.249 \times 10^{1}
\end{array}\right]} \\
& 6 ; 5=-22.49200
\end{aligned}
$$

Wherever you are within a matrix, you can replace or modify the currently selected element in three ways:

1. Let an arbitrary monadic function operate on the selected element. If you need any menus to reach a function, they will temporarily replace the Matrix Editor menu; exiting those menus will bring you back to the Matrix Editor menu.
2. Recall a number from any register or variable via RCL.
3. Simply key in a new number replacing the old one.

## Example (continued):

Replace the last matrix element by 17.435 .
 (note ENTERT puts the new number into the matrix but OLD stays usable).

You can also store (copy) the current matrix element in any destination using STO.

Repeatedly pushing the cursor in one direction (e.g. by $\rightarrow$ ) will jump from the $1^{\text {st }}, 2^{\text {nd }}$, etc. to the last row and then return to the $1^{\text {st }}$ row in default wrap mode. If grow is set instead, another $\rightarrow$ from the very last (i.e. bottom right) matrix element will add an entire new row to the matrix.

$$
\begin{aligned}
& \text { 2021-06-12 22:14 Сட } \Varangle \pi / \text { max grow } \bar{\uparrow} \\
& \begin{array}{l}
{\left[\begin{array}{cccc}
\ldots & \ldots & \ldots & \ldots . \\
7.465 \times 10^{1} & 2.290 \times 10^{-1} & -9.340 \times 10^{-2} & \ldots \\
3.390 \times 10^{1} & -7.940 \times 10^{1} & 3.436 & \ldots \\
4.880 \times 10^{-2} & 7.000 & 5.980 & \ldots \\
0.000 & 0.000 & 0.000 & \ldots .
\end{array}\right]} \\
7 ; 1=0.00000
\end{array}
\end{aligned}
$$

Here, we are done with this matrix for now. So press EXIT and you will see again:

Note the $1^{\text {st }}$ matrix element is not highlighted anymore since you left the Matrix Editor.


Thus, just entering (4) will display, due to automatic stack lift:
[<xJ IIatrix]
[7×5 Matrix]

So matrix editing is easy and straightforward. The IOI contains additional information, also about the further commands DELR, INSR, and $\mathrm{R} \leftrightarrows \mathrm{R}$ showing up in the Matrix Editor menu.

## Vectors and Matrices: Complex Stuff

Your WP43 supports also complex vectors and matrices, i.e. matrices comprising complex elements. They are created and initialized like real objects via NEW or DIM as explained above. Or you can recall a real matrix and edit it; if you enter one or more complex numbers for its elements it becomes a complex matrix - you can store it at the same or another place after editing.

Example (continuation of p. 177):
Create and store a complex matrix $\left[\begin{array}{cc}5+8 i & \pi i \\ -2 & 4-3 i\end{array}\right]$.

## Solution:

Remember we have created a $2 \times 2$ matrix just a few pages ago. So it is most easy to recall it for using it as a template:

## RCL VAR MA

DSP 2 since this will suffice for the process following.
EDIT

$$
\left[\begin{array}{ll}
4.00 & -3.00] \\
-2.00 & 1.00
\end{array}\right] \quad[2 \times 3 \text { Matrix] }
$$

We can now enter the new elements here as we have done before:


EXIT STO 01

Since we edited on the stack and stored the resulting new complex matrix in a new location, the old real matrix MA is not affected at all.

Compare pp. 164f for the input and formatting of complex numbers. Everything else works as it does for real matrices. You see complex matrices are no complex topic at all for you with your WP43.

## Vectors and Matrices: Calculating

As we have seen on $p$. 176, your WP43 can multiply a matrix by a plain number (a.k.a. a scalar); doing this, each element of said matrix is multiplied by said number. Additions, subtractions, and divisions work alike for a matrix $\boldsymbol{y}$ combined with a scalar $\boldsymbol{x}$. Vice versa, with a scalar $\boldsymbol{y}$ and a
matrix $\boldsymbol{x}$, additions, subtractions and multiplications will work the same way (divisions are special, see below). Also monadic functions will operate on each matrix element in your WP43, if applicable.

## Examples:

With an arbitrary matrix in $\mathbf{X}$, pressing...

- $\sqrt{\boldsymbol{x}}$ will extract the square root of each matrix element individually. If CPXRES is set, a real matrix $\boldsymbol{x}$ comprising at least one negative element will become complex this way.
- $\boldsymbol{x}^{2}$ will square each matrix element individually (instead, use ENTER $\boldsymbol{x}$ for squaring the matrix, if applicable).
- ||x| will calculate the absolute value of each matrix element (instead, use MATX ENORM for calculating the Euclidean norm of the matrix or take $|M|$ for getting its determinant).
- +1/ will change the sign of each matrix element.

You can also let the dyadic functions $\oplus, \boxed{\square}, \boldsymbol{x}$, or $(1)$ operate on two matrices or vectors (i.e. DT 8 and 9), if the rules of linear algebra are obeyed:

|  | $y$ | $\boldsymbol{x}$ | Operation | Resulting $\boldsymbol{x}$ |
| :---: | :---: | :---: | :---: | :---: |
| $\pm$ | [ $\boldsymbol{m} \times \boldsymbol{n}$ Matrix] | [ $\boldsymbol{m} \times \boldsymbol{n}$ Matrix] | $[y]+[x]$ | [ $\boldsymbol{m} \times \boldsymbol{n}$ Matrix] |
| $\square$ |  |  | $[y]-[x]$ |  |
| x |  | [ $\boldsymbol{n} \times \boldsymbol{p}$ Matrix] | $[y] \cdot[x]$ | [ $\boldsymbol{m} \times \boldsymbol{p}$ Matrix] |
| (1) |  | [ $\boldsymbol{n} \times \boldsymbol{n}$ Matrix] | $[y] \cdot[x]^{-1}$ | [ $\boldsymbol{m} \times \boldsymbol{n}$ Matrix] |

The $1^{\text {st }}$ row of this table reads as follows: For adding or subtracting two arbitrary matrices, both must be of identical size and shape, and the result will be of the same size and shape. Subsequent rows read in analogy. ${ }^{181}$

[^87]If either matrix is complex, the result will often be complex as well.

Example (continuation of p. 177):
Multiply the matrices in R00 and MA. Output format shall be FIX 3.
Solution (we omit the menu section in the following pictures):
DSP 3
RCL VAR MA (or RCL $\propto$ (M) A ENTERT), if you have defined many variables already - cf. p. 60)
[2×2 C Matrix]
$\left[\begin{array}{rr}4.000 & -3.000 \\ -2.000 & 1.000\end{array}\right]$

Note the ' $2 \times 2 \mathbb{C}$ Matrix' in $\mathbf{Y}$ is the complex matrix we entered in previous chapter - the stack handles matrices as it handles other objects. Now let's recall R00:

RCL 0
[ $<\times<$ rlatrix]
$\left[\begin{array}{lll}-0.667 & 8.000 & 4.6671 \\ 16.667 & 0.000 & 1.333\end{array}\right]$

The ' $2 \times 2$ Matrix' in $\mathbf{Y}$ now is the one we have recalled from MA into $\mathbf{X}$ before recalling $\mathbf{R 0 0}$. We multiply $\boldsymbol{y}$ times $\boldsymbol{x}$ as usual by $\boldsymbol{X}$ resulting in
$[<\times<$ \& l'latrix]
$\left[\begin{array}{lll}-79.000 & 48.000 & 19.000\end{array}\right]$
$27.000-24.000-11.000]$

You see that arithmetic operations on matrices are almost as easy as on scalars using your WP43.

And your WP43 features further matrix operations: |M| for computing determinants, $[\mathrm{M}]^{-1}$ for inverting, $[\mathrm{M}]^{\top}$ for transposing, M.LU and M.QR for computing the LU and QR decompositions, and two norms (Euclid's ENORM and the row norm RNORM) - please look them up in the IOI.

[^88]
## Example:

We want to invert a $2 \times 2$ matrix $\quad[M]=\left[\begin{array}{ll}1 & 2 \\ 3 & 4\end{array}\right]$.

## Solution:

Just enter the matrix as usual
2 ENTER $\mathbb{M A T X}$ DIM $\alpha$ ENTERT creates $M$ as a $2 \times 2$ matrix.

## RCL VAR M

EDIT etc. until EXIT
[2×3 Matrix]
$\left[\begin{array}{ll}1.000 & 2.000 \\ 3.000 & 4.000\end{array}\right]$
$[M]^{-1}$
$\left[\begin{array}{cc}1<\times 3 & \text { rlatr } 1 \times 1 \\ -2.000 & 1.000 \\ 1.500 & -0.500\end{array}\right]$

Thus, the inverted matrix reads $[M]^{-1}=\left[\begin{array}{cc}-2 & 1 \\ 1.5 & -0,5\end{array}\right]$.

For two vectors of identical size, there are two special multiplications provided: DOT and CROSS. DOT will return the dot product, a scalar exactly what the table above says for $\boldsymbol{m}=\boldsymbol{p}=1$. CROSS works exclusively for two $2 D$ or $3 D$ vectors and will return their cross product.

## Example from the HP-27 OH:

The force $\vec{F}$ on a particle with charge $q$ which is moving with a velocity $\vec{v}$ through a magnetic field $\vec{B}$ is given by $\vec{F}=q \vec{v} \times \vec{B}$. Suppose a proton ( $q=$ $-e=1.6 \cdot 10^{-19}$ coulomb ) is moving with velocity $\vec{v}=$ (0.4 $2.8-1.2$ ) $10^{7} \mathrm{~m} / \mathrm{s}$. A uniform magnetic field surrounding the proton is of a strength $\vec{B}=\left(\begin{array}{lll}1.3 & -0.3 & 0.7\end{array}\right)$ tesla. Calculate the force on the proton. This can be written as

$$
\vec{F}=q \vec{v} \times \vec{B}=1.6 \cdot 10^{-19} \cdot 10^{7} \cdot\left(\begin{array}{lll}
0.4 & 2.8 & -1.2
\end{array}\right) \times\left(\begin{array}{ll}
1.3 & -0.3
\end{array}\right.
$$

## Solution (FIX 01 will do):

In cross products, vectors must be entered in proper sequence as written from left to right:

[^89]EDIT etc. until EXIT

STO $\propto$ ENTER $\mathbb{B}$ creates also $\mathbf{B}$ as a $3 \times 1$ matrix.
RCL VAR B
EDIT etc. until EXIT
cross returns

[E] 7
$1.6\left[E+19\right.$ returns $\quad\left[2.6 \times 10^{-12}-2.9 \times 10^{-12}-6.0 \times 10^{-12}\right]^{\top}$
... newtons, of course.
The total 'size' or absolute value of this force is
ENORM

## Compare the weight of a proton:

CONST (M)P F1
CONST G F6 X
recall the proton mass $\mathbf{m}_{\mathbf{p}}$.
recall the gravity acceleration $\mathbf{g}_{\oplus}$ and multiply.
$1.6 \times 10^{-26}$
So this is a force ratio of
$\square$

Thus, physicists deliberately neglect gravitational effects in such microscopic calculations as long as no higher precision is required.

If you just want to perform elementary vector operations in $2 D$, however, there are two simple alternatives (known also from earlier calculators):

1. Enter the Cartesian components of each vector in $\mathbf{X}$ and $\mathbf{Y}$ (if necessary, convert its polar components into Cartesian ones by before) and use [ $\Sigma \boldsymbol{+}$ ] for additions (cf. examples on pp. 142ff). Recall the result via (SUM) ; it may look like this, for example:

$$
\begin{aligned}
& \Sigma y= \\
& \Sigma x=
\end{aligned}
$$

2. Calculate with complex numbers. In complex domain, $\oplus, \square$, and also $2 D$ vector multiplications are possible since the commands DOT and CROSS are contained in CPX as well (cf. pp. 164ff).

## Vectors and Matrices: Solving Systems of Linear Equations

Your WP43 can also solve simultaneous linear equations (of the kind $[A] \cdot \vec{X}=\vec{B}$ ) for you. To deal with such problems, proceed as follows:

1. Specify the number of unknowns ${ }^{182}$ (e.g. 4) by entering MATX SIM EQ 04
Your WP43 automatically creates and dimensions three matrices (if necessary): Mat.A, Mat.B, and Mat.X. You will see a new menu showing up:
2. Press Mat A. The Matrix Editor will open and you can enter the 16 elements of the $4 \times 4$ coefficient matrix Mat.A (you learned on pp. 173ff how to do this). When done, leave the Matrix Editor by EXIT to return to the menu shown in step 1.
3. Press Mat $B$ and enter the elements of the $4 \times 1$ constant matrix Mat. $B$ the same way (this is a vector actually).
4. Press Mat $X$ to let your WP43 compute the $4 \times 1$ solution matrix Mat. $X$ (a vector again). You are done!

To work another problem with the same number of unknowns, return to step 2 or 3 . For a problem with a different number of unknowns, press EXIT for leaving the menu shown above, and start over with step 1.

It goes without saying this works for real and complex matrices.

[^90]Example from the HP-15C Advanced Functions Handbook:
Find the output voltage $V_{0}$ for the filter network shown. Input voltage shall be $\boldsymbol{V}=10 \mathrm{~V}$ at a frequency of $\boldsymbol{\omega}=15 \times 10^{3} \mathrm{rad} / \mathrm{s}$, and $\boldsymbol{R}_{1}=100 \Omega, \boldsymbol{R}_{2}=1 \mathrm{M} \Omega$,

$R_{3}=100 \mathrm{k} \Omega, C_{1}=250 \mathrm{nF}$,
$C_{2}=25 \mu \mathrm{~F}$, and $L=10 \mathrm{mH}$.

## Solution:

Describe the network using loop currents (via Kirchhoff equations):

$$
\left[\begin{array}{cccc}
\left(R_{1}+i \omega L-i / \omega C_{1}\right) & \left(i / \omega C_{1}\right) & 0 & 0 \\
\left(i / \omega C_{1}\right) & \left(R_{2}+i \omega L-i / \omega C_{1}\right) & \left(-R_{2}\right) & 0 \\
0 & \left(-R_{2}\right) & \left(R_{2}-i / \omega C_{2}+i \omega L\right) & (-i \omega L) \\
0 & 0 & (-i \omega L) & \left(R_{3}+i \omega L-i / \omega C_{2}\right)
\end{array}\right]\left[\begin{array}{l}
I_{1} \\
I_{2} \\
I_{3} \\
I_{4}
\end{array}\right]=\left[\begin{array}{l}
V \\
0 \\
0 \\
0
\end{array}\right]
$$

This system shall be solved for the four unknown loop currents $I_{1}$ through $I_{4}$. Then, $\boldsymbol{V}_{0}=\boldsymbol{R}_{3} I_{4}$. ENG 4 will suffice for the output.

With the values given above, this system of linear equations will look like:

$$
\left[\begin{array}{cccc}
100-i 116.67 & i 266.67 & 0 & 0 \\
i 266.67 & 10^{6}-i 116.67 & -10^{6} & 0 \\
0 & -10^{6} & 10^{6}+i 147.33 & -i 150 \\
0 & 0 & -i 150 & 10^{5}+i 147.33
\end{array}\right]\left[\begin{array}{c}
I_{1} \\
I_{2} \\
I_{3} \\
I_{4}
\end{array}\right]=\left[\begin{array}{c}
10 \\
0 \\
0 \\
0
\end{array}\right]
$$

## Start the solver via (MATX SIM EQ 04.

Press Mat A; enter the big complex matrix using the Matrix Editor as explained above.
Press Mat $\mathbf{B}$ and enter the voltage vector.
Finally, call Mat X and you will get the vector of the loop currents:

$$
\text { [ } \left.199.50 \times 10^{-6}+i \times 4.09664 \times 10^{-3} \ldots\right]^{\top}
$$

Its last $\left(4^{\text {th }}\right)$ element is the current we need. Separate it with:

## index Var Mat X

4 STO (1)
1 STO (J)
set the index pointers to $4^{\text {th }}$ row $\ldots$
... and 1st column.

# $53.446 \times 10^{-6}-\mathfrak{i} \times 2.2599 \times 10^{-6}$ 

100 E 3 multiply this current with $R_{3}$ and set POLAR
(SF X

$$
5.3494 \Varangle-2.4212^{\circ}
$$

The output voltage is $V_{0}=5.35 \mathrm{~V}$ with a little phase shift.

## Vectors and Matrices: Eigenvalues and Eigenvectors

An eigenvalue is a real or complex number $\lambda$ solving the matrix equation $[A] \cdot \vec{X}=\lambda \cdot \vec{X}$. ${ }^{183}$ Then, the vector $\vec{X}$ is called an eigenvector of $[A]$. Usually, there will be more than one $\lambda$ and a multitude of vectors $\vec{X}$ solving this problem. Thus, the simplest set of linearly independent vectors $\vec{X}$ is chosen to build the base of the eigenspace belonging to a particular eigenvalue found. And the simplest set of eigenvectors building a base of a space of the same dimension as $\vec{X}$ are called the eigenvectors of $[A]$. Your WP43 can solve such problems for you as well:

## Example 1:

What are the eigenvalues of the matrix $[M]=\left[\begin{array}{ll}2 & 1 \\ 6 & 1\end{array}\right]$ ?

## Solution:

We have got a $2 \times 2$ matrix named $\mathbf{M}$ already. We don't need its old contents anymore so we simply recall and edit it:


DISP FIX 0
MATX EDIT etc.
$\left.\begin{array}{ll}2.0 & 1.0 \\ 6.0 & 1.0\end{array}\right]$

## STO VAR M

The eigenvalues are the solutions of the characteristic polynomial of this problem:

[^91]$(2-\lambda)(1-\lambda)-6=0$
\[

M=\left[$$
\begin{array}{c}
{[2 \times 2 \text { Matrix] }} \\
{\left[\begin{array}{rr}
4.0 & 0.0 \\
0.0 & -1.0
\end{array}\right]}
\end{array}
$$\right.
\]

being the matrix with the eigenvalues as its diagonal elements. Note this resulting diagonal matrix is pushed on the stack.

Example 1 (continued):
Now, what are the eigenvalues of $[N]=\left[\begin{array}{cc}3 & 4 \\ -4 & 3\end{array}\right]$ ?
Solution:
(4) EDIT etc.

$$
M=\left[\begin{array}{cc}
2 \times 2 & \text { Matrix }
\end{array}\right]\left[\begin{array}{rr}
3.0 & 4.0 \\
-4.0 & 3.0
\end{array}\right]
$$

STO $\propto \mathbb{N}$ ENTERT
$\triangle$ EIGVAL returns

$$
\left.\left.\begin{array}{rr}
M=[2 \times 2 & \text { Matrix }] \\
N=[2 \times 2 & \text { Matrix }
\end{array}\right] \begin{array}{rl}
3.0+i \times 0.0] \\
{[3.0+i \times 4.0} & 0.0+i \times 0.0 \\
0.0+i \times 0.0 & 3.0-i \times 4.0
\end{array}\right]
$$

Note that we got complex eigenvalues here although $\mathbf{N}$ comprises only real elements.

Example 2:
What are the eigenvalues of $[Q]=\left[\begin{array}{ccc}0 & 0 & -2 \\ 1 & 2 & 1 \\ 1 & 0 & 3\end{array}\right]$ ?
Solution:
3 ENTER $\boldsymbol{T}$ MATX DIM $\propto$ Q ENTER $\boldsymbol{Q} \quad$ creates $\mathbf{Q}$ as a $3 \times 3$ matrix.
RCL VAR Q
EDIT etc.
[2×2 C Matrix]
$\left[\begin{array}{ccc}0.0 & 0.0 & -2.0 \\ 1.0 & 2.0 & 1.0 \\ 1.0 & 0.0 & 3.0\end{array}\right]$


Note one eigenvalue comes twice here.
Let's get the eigenvectors of $\mathbf{Q}$ now - they will be put out as row vectors of a matrix:
$x^{2} \geqslant \mathbf{y} \quad$ returns $\mathbf{Q}$ into $\mathbf{X}$


EIGVEC pushes this matrix on the stack


RCL (L) recalls $\mathbf{V}$.
$\Delta[M]^{-1} x^{2} \geqslant y \quad x$ returns

This looks very much like what was returned for the eigenvalues of $\mathbf{Q}$ above. Let's check:
(DSP) 4

- returns
$\left[\begin{array}{lll}0.0000 & 0.0000 & 0.000001 \\ 0.000 & 0.0000 & 0.0000 \\ 0.000 & 0.000 & 0.000\end{array}\right]$

So the result of $[\mathrm{V}]^{-1} \cdot[\mathrm{Q}] \cdot[\mathrm{V}]$ with $\mathbf{V}$ being the matrix of the eigenvectors of $\mathbf{Q}$ is exactly the diagonal matrix of the eigenvalues of $\mathbf{Q}$. This is proven to apply generally for any square matrix.

Your WP43 can compute eigenvalues and eigenvectors for matrices featuring rational elements as well:

Example 3:
What are the eigenvalues of $\left[\begin{array}{cccc}-38 & 43 / 7 & 63 / 2 & 1149 / 14 \\ -14 & 19 / 7 & 7 & 181 / 7 \\ -8 / 7 & -122 / 49 & 24 / 7 & 177 / 49 \\ -16 & 26 / 7 & 13 & 244 / 7\end{array}\right]$ ?

## Solution:

4 ENTERT (MATX NEW creates a $4 \times 4$ matrix.
EDIT $\mathbf{3 8}+$ + $\rightarrow$. $\mathbf{4 3 . 7} \rightarrow \mathbf{~ . 6 3 . 2 ~} \rightarrow \mathbf{. 1 1 4 9 . 1 4 ~} \rightarrow$ etc.
Note each matrix element can be entered as an integer or fraction but is converted to a real number following the current display settings as soon as said element is closed:
[3×3 Matrix]
$\left[\begin{array}{rrrr}-38.0000 & 6.1429 & 31.5000 & 82.0714 \\ -14.0000 & 2.7143 & 7.0000 & 25.8571 \\ -1.1429 & -2.4898 & 3.4286 & 3.6122 \\ -16.0000 & 3.7143 & 13.0000 & 34.8571\end{array}\right]$

DSP 0 (1) shall suffice here
$\left[\begin{array}{rrrr}-38.0 & 6.1 & 31.5 & 82.1 \\ -14.0 & 2.7 & 7.0 & 25.9 \\ -1.1 & -2.5 & 3.4 & 3.6 \\ -16.0 & 3.7 & 13.0 & 34.9\end{array}\right]$
MATX ©IIGAL returns $\left[\begin{array}{cccc}-5.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -2.4 \times 10^{-32} & 0.0 & 0.0 \\ 0.0 & 0.0 & 5.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 3.0\end{array}\right]$

Note the $2^{\text {nd }}$ eigenvalue is zero here.

RCL (L)
DSP 2
EIGVEC displays
$\left[\begin{array}{rrrr}4.00 & 1.25 & 5.00 & 1.33 \\ 3.00 & -0.50 & -4.00 & 0.67 \\ 1.00 & -1.00 & 5.00 & -1.00 \\ 1.00 & 1.00 & 1.00 & 1.00\end{array}\right]$

Generally, your WP43 solves characteristic polynomials numerically.

## Vectors and Matrices: Dealing with Statistical Data

You can enter $2 D$ data using an input matrix as well as keying them in point after point. How is this done?

Let's return to the application concerning measuring system analysis introduced on p .121 . Resume it with its step 4 - remember there were $\geq 30$ samples measured twice in a special way using the measuring system under investigation, resulting in a corresponding number of pairs of measured values recorded:
4. Create a $1 \times 2$ named matrix and open it for editing:

1 ENTER 2 MATX DIM $\alpha$ (M) A ENTER $\mathbf{x}$ creates MSA accordingly.


GROW allows the matrix to grow with data entered.
Now key in all recorded pairs of measured values: The $1^{\text {st }}$ value shall be $\boldsymbol{y}$, the $2^{\text {nd }} \boldsymbol{x}$ - so the keystroke sequence will be $\boldsymbol{m v 1} \rightarrow \boldsymbol{m v 2}$. Then, another $\rightarrow$ lets the matrix grow by one row for the next data point.
With all points entered in the matrix, eventually key in
EXIT

## STAT CL $\Sigma$

## RCL VAR MSA

$\mathbf{\Sigma +} \quad$ Calling $\Sigma+$ with such a matrix in $\mathbf{X}$ will accumulate all your data at once and display the very last point in $\mathbf{X}$ and $\mathbf{Y}$ (and save a copy of the input matrix in $\mathbf{L}$ ). ${ }^{184}$

[^92]
5. Call $\boldsymbol{\Delta}$ PLOT for plotting these points in a new window (see a typical plot here, cf. p. 122).
6. Press CENTRL to fit a straight line through all these points. Check if this line deviates significantly from $\boldsymbol{y}=\boldsymbol{x}$; coarsely (with some 95\% confidence), this holds if $a_{0} \pm$ $2 s\left(a_{0}\right)$ excludes 0 or $a_{1} \pm$ $2 s\left(a_{1}\right)$ excludes 1.
If it does then your measuring system may be running up still (wait some more time for equilibrium) and/or show an unstable zero (check and fix).
Then restart the procedure with a new set of measurements (step 2 on p. 121).
7. Else press $\mathbf{s}_{\mathbf{m i}}$ to push the precision of your measuring instrument under investigation (as measured under the boundary conditions prevailing during steps 2 and 3 of this test) on the stack. Then enter $\mathbf{3 0 ~} \mathbf{x}(1 / x \times$ and multiply with the width of the tolerance zone you want or have to control with this instrument.

If your result is $\geq 1$ then this measuring device may be used for controlling series production with this tolerance zone under these conditions (i.e. it is a capable instrument for this control job); else you shall look for a more precise instrument, better measuring conditions, or a wider tolerance.

## Vectors and Matrices: Summary of Functions

Assume $\mathbf{X}$ contains a matrix. Then there are functions operating on the entire matrix $\boldsymbol{x}$ and others operating on its elements $\boldsymbol{x}_{\boldsymbol{i j}}$ individually. Let's list the $1^{\text {st }}$ set first:

This matrix input method may also be used for data sets you want to plot for other reasons.

- General mathematics:
- Monadic functions operating on the entire matrix $\boldsymbol{x}$ :
(ENORM) returns the Euclidean norm of $\boldsymbol{x}$ (i.e. a real number),
(RNORM] computes the row norm of $\boldsymbol{x}$ (i.e. a real number),
(RSUM) computes the row sum of $\boldsymbol{x}$ (i.e. a vector),
[ $|M|$ 〕 returns the determinant of $\boldsymbol{x}$ (i.e. a real or complex number),
$\left[[M]^{\top}\right]$ returns the transpose matrix of $x$,
( $[M]^{-1}$ ) returns the inverse matrix of $x$,
[EIGVAL] returns the eigenvalues of $\boldsymbol{x}$, and [EIGVEC] its eigenvectors (cf. pp. 188ff), while [UNITV] returns the unit vector of $\boldsymbol{x}$ (see the ReM).
- Monadic functions operating on each element $\boldsymbol{x}_{i j}$ of $\boldsymbol{x}$ individually:
$\left[1 / x, \notin,\left[|x|, \boxed{(1) \sqrt{x}}\right.\right.$ and $x^{2},[\sqrt[3]{x})$ and $\left(x^{3}\right),\left[\sqrt{ }\left(1+x^{2}\right)\right],\left[2^{x}\right]$ and (lbx), $\boldsymbol{e}^{x}$ and $\left[\mathbf{I n}\right.$, $10^{x}$ and 19 , (sin), (cos), (tan), (sinh), (cosh), and (tanh) as well as their inverses work as explained for real and complex numbers above,
$\left[e^{x}-1\right]$ and $[\ln (1+x)]$ return more accurate results with $x_{i j} \approx 0$;
[sinc] returns a matrix containing $\frac{\sin \left(x_{i j}\right)}{x_{i j}}$ and (sincा] a matrix containing $\frac{\sin \left(\pi x_{i j}\right)}{\pi x_{i j}}$ for $x_{i j} \neq 0$ and 1 for $x_{i j}=0$,
$\left[(-1)^{\mathrm{x}}\right]$ returns $\cos \left(\pi x_{i j}\right)+i \sin \left(\pi x_{i j}\right)$ for non-integer $x_{i j}$.
(RDP) $\boldsymbol{n}$ rounds $x_{i j}$ to $\boldsymbol{n}$ decimal places in FIX format, (ROUND) rounds $\boldsymbol{x}_{i j}$ using the current display format, and (RSD) $\boldsymbol{n}$ rounds $\boldsymbol{x}_{\boldsymbol{i j}}$ to $\boldsymbol{n}$ significant digits;
(erf) and (erfc] return the error function and its complement for each $\boldsymbol{x}_{i j}$,
(FIB) the extended Fibonacci number for each $\boldsymbol{x}_{\boldsymbol{i} j}$,
calculates the factorial of $\boldsymbol{x}_{i j}$ or $\Gamma\left(x_{i j}+1\right)$, respectively, $[\Gamma(x))$ the Gamma function for each $\boldsymbol{x}_{i j}$, while ( $\ln \Gamma$ ) computes its logarithms, and


## [ $\zeta(x)$ ) returns Riemann's Zeta function for each $\boldsymbol{x}_{i j}$.

For complex matrices, (conj) returns a matrix with the complex conjugates of $\boldsymbol{x}_{i j}$. For real matrices,
(ceil) returns a matrix with the smallest integers $\geq x_{i j}$ and
[floor) with the greatest integers $\leq x_{i j}$,
(FP) returns a matrix with the fractional parts of $x_{i j}$ and
(IP) with their integer parts, preserving their signs,
(EXPT) returns a matrix with the exponents of each $x_{i j}$ and (MANT) with their mantissas, while (sign) replaces each $\boldsymbol{x}_{i j}$ by its signum returning 1 for $\boldsymbol{x}_{i j}>0$, -1 for $x_{i j}<0$, and 0 for $x_{i j}=0$ or non-numeric data.

- Dyadic functions operating on $\boldsymbol{x}$ and $\boldsymbol{y}$ :
$\oplus, \boxed{\square}, \boxed{x}$, and $\square$ work as explained on pp. 181ff,
[cross] represents the cross product operating on two real $2 D$ or $3 D$ vectors of identical size (cf. pp. 184f),
( $V_{4}$ ) returns the angle between two real $2 D$ or $3 D$ vectors of identical size, and
[dot] represents the dot product operating on two vectors or matrices of matching size (cf. pp. 184f).
- Dyadic functions operating on each element $\boldsymbol{y}_{i j}$ of $\boldsymbol{y}$ individually:
$\boldsymbol{y}^{x}$ raises $\boldsymbol{y}_{i j}$ to the power of $\boldsymbol{x}$, while
$x_{V}$ extracts the $x^{\text {th }}$ roots of $y_{i j}$.
$\left(\log _{\mathrm{x}} \mathrm{y}\right)$ calculates the logarithms of $\boldsymbol{y}_{i j}$ for base number $\boldsymbol{x}$.
- Isolating and manipulating bulk parts of a complex matrix $\boldsymbol{x}$ : Use ...
$(C X \rightarrow R E)$ for cutting $x$ in its two parts,
( $\mathrm{RE} \rightarrow \mathrm{CX}$ ) for composing $\boldsymbol{x}$ from its two parts,
(Re) for isolating its real part and (Im) for its imaginary part,
(|x|) for isolating its magnitude and ( $\Varangle$ ) for its phase, and
(Relim) for swapping its real and imaginary part.

The Matrix Editor was demonstrated on pp. 173ff. Turn to the ReM for additional information about all matrix operations provided.

If you look for general information about vectors and matrices, and further applications, please turn to textbooks covering linear algebra.

## Dates

For date calculations, choose one out of three date display modes (DDM) on your WP43: Y.MD, D.MY, and M.DY (these mode-setting commands are contained in the $2^{\text {nd }}$ view of CLK). ISO Y.MD is startup default.

Date input is decimal according to the DDM chosen and is terminated by (as shown on pp. 72f).

## Example:

## .d $\lg$



The $3^{\text {rd }}$ of November in 2022 is entered
2022.1103 .d in Y.MD,
3.112022 .d in D.MY, and
11.032022 .d in M.DY.

Alternatively, any real number may be converted into a date via CLK $x \rightarrow$ DATE, and any triple of reals or integers via $\rightarrow$ DATE (cf. p. 43). Input comprising more than the necessary digits for a date in the DDM
 selected will be rounded.

Vice versa, DATE $\rightarrow$ splits a date in three integers and pushes them on the stack as demonstrated on p. 43. Note that both DATE $\rightarrow$ and $\rightarrow$ DATE observe the DDM chosen. If you want to extract particular information from a date independent of current DDM, we recommend using one of the operations CLK DAY, MONTH, or YEAR.

Like in the status bar, a closed date input or a date output returned by a function is displayed as in the following example:

$$
\begin{aligned}
2021-11-03 & \text { in Y.MD, } \\
3.11 .2021 & \text { in D.MY, } \\
11 / 03 / 2021 & \text { in M.DY. }
\end{aligned}
$$

So you know the effective DDM immediately from looking at the date format in the status bar.

CLK WDAY takes a date from the stack - or a decimal input of e.g. 2013.0504 in Y.MD mode (equivalent to inputs of 4.052013 in D.MY or 5.042013 in M.DY) - and returns an integer indicating the position of this day in the corresponding week, temporarily headed by the name of this weekday:

## Saturday

6
Expect similar returns after CLK DATE .

Use $\mp$ to add an integer (or real), representing an integer number of days, to a date - and $\square$ to subtract such a number from a date. The result will be a date again.

And a date may be subtracted from another date, resulting in an integer difference of days. Compare the matrices on pp. 75 f .

## Example (with startup default settings):

The interval between $9^{\text {th }}$ of November in 1931 and the $26^{\text {th }}$ of March in 2022 amounts to 90 years and almost five months. How many days were these? And how many hours?

[^93]Calculation of weekdays for the past depends on the calendars used at that time - there may be different true results for different countries depending on the date the particular country introduced the Gregorian calendar. Officially, that calendar became effective in 1582-10-15 for the catholic world. Large parts of the world took their time and switched later. Store the proper date for your geographic area of interest using the command J/G (see the chapter Localizing Calculator Output above and check Wikipedia for dates applicable).
Dates before the year 8 A.D. may be indicated differently than they were experienced at the time due to the inconsistent application of the leap year rule before. We count on your understanding and hope this shortcoming will not affect too many calculations.
Note that 8 A.D. should be better written A.D. VIII instead - quite some false Latin is found in the English language. Nobody, however, counted years this way at that time around the Mediterranean Sea, it was the year DCCLXI A.V.C. in best case (actually, this notation was broadly introduced some XL - or even CD - years later). Also note the Julian calendar was introduced and became valid not earlier than DCCVIII A.V.C. before, months were organized differently. Julius Caesar was daggered in DCCIX A.V.C. so he did not live to see 'his' calendar for long.

Calendars may be a sensitive topic. Note there are also other calendars in use on this planet today, e.g. in the Muslim or Buddhist world.

## Solution:

1931.1109
$-$
24 x

2022-03-26
1931-11-09
33010 days.
792240 nours.

Using CLK $J \rightarrow D$ and $\mathrm{D} \rightarrow \mathrm{J}$, you can deal with Julian day numbers useful in astronomy (please see the $I O I$ ). And there is SETDAT, serving obvious purposes on your WP43. But that's it - these are all the legal operations on dates.

GAP, ALL, ENG, FIX, or SCI have no effect on dates.

## Times

There also is a special data type for time calculations on your WP43. Sexagesimal times are entered most easily using the format hhh.mmssfff terminated by h.ms - with hhh standing for hours, mm for minutes, ss for seconds, and fff for decimal fractions of seconds (these fractions as well as the hours may take more or less than three digits).

## h.ms $10^{x}$

Example (with startup default settings):
Enter 45 hours, 39 minutes, and 7.8642 seconds:

$$
45 \bigodot 39078642 \mathrm{~h} \cdot \mathrm{~ms}
$$

This is displayed with startup default TDISP 0 :
Choosing CLK $\triangle$ TDISP 3 will return
... instead, while TDISP (2) will return


The latter two formats also allow for compact returns when calling TIME. Note there is no display rounding for times.

The : (or a trailing s, see below) is the unambiguous indicator for a time displayed on your WP43. There may be leading zeros in the minutes and
seconds sections and a settable number of digits after the $2^{\text {nd }}$ : You can choose 12- or 24-hour display for time of day.

Example (continued):
Call TIME in the evening and you might get

## 21:47

(CF SYS.FL TDM24 will return
9:47pm

When time of day is returned by a function, it will be displayed according to your choice - internally, however, it is stored as standard 24-hour time for further calculations.

Add and subtract sexagesimal time intervals simply using $\oplus$ and $\square$. Multiply or divide such intervals by any integer, rational, or real number the result will stay a time. If you add any integer, rational or real number to a time, it will be taken as decimal hours and automatically converted to a time before adding. This applies to subtractions in analogy. If you divide by a time, it will be converted to decimal hours before. Compare the matrices on pp. 75 f .

## Example (with startup default settings):

To meet your date at 5:25 p.m. at Stanford, you need 15 ' from your office to get your car out of the parking garage, 1.5 hours for the ride, and 12' for walking from the parking lot to lecture hall. Being careful, you count in another quarter of an hour for a possible traffic jam on the expressway. When do you have to leave your office? If Stanford is 57 miles away, what time do you need for a mile on average?

## Solution:

CLK $\triangle$ TDISP 2


You shall leave at 3:13 p.m. the latest.

This result looks like a 12h-time here even with startup default settings - your WP43 cannot know better based on the input given.

If a result of your time calculations will become less than the display limit you set via TDISP (so no non-zero digit of the resulting time can be shown within the limit set) the result will be displayed in a fixed format like SCI 4 or ENG 4, depending on ALLENG. With ALLENG set, the following three resulting times will be displayed like this, for example:

| Time: | 5 min 8.4321 s | 6.14032 s | 0.0934215 s |
| :--- | :--- | :--- | :--- |
| TDISP 1 or 2 | $0: 05$ | 6.1403 s | $93.421 \times 10^{-3} \mathrm{~s}$ |
| TDISP 3 | $0: 05: 08$ | $0: 00: 06$ | $93.421 \times 10^{-3} \mathrm{~s}$ |
| TDISP 4 | $0: 05: 08.4$ | $0: 00: 06.1$ | $93.421 \times 10^{-3} \mathrm{~s}$ |
| TDISP 5 | $0: 05: 08.43$ | $0: 00: 06.14$ | $0: 00: 00.09$ |
| TDISP 6 | $0: 05: 08.432$ | $0: 00: 06.140$ | $0: 00: 00.093$ |
| TDISP 0 | $0: 05: 08.4321$ | $0: 00: 06.14032$ | $0: 00: 00.0934215$ |

You can convert such a (closed) sexagesimal time to decimal hours using d, e.g. for further calculations; and you can reconvert (closed) decimal hours to a sexagesimal time by pressing h.ms.

For converting a bigger number of seconds (e.g. 1234) into HMS format do as follows: 1234 ENTERT 3600 ( 1 h.ms returning 0:20:34.

There is only one more dedicated time command - SETTIM, serving obvious purposes on the calculator. ${ }^{186}$ The commands GAP, ALL, ENG, FIX, or SCI have no effect on times.

See OMS 5 for the TIMER.

[^94]
## Proposal for Configuration Setting

Based on what was discussed above, the following settings (deviating from startup default) may be most beneficial for your scientific or engineering calculations in real and complex domain:

- Set SSIZE8 and SPCRES.
- Set SCI 5 or ENG 5 (whatever you prefer), DENMAX 1000, and RANGE 99.
- Choose your regional preferences (cf. pp. 83f).
- Select your favorite formats for dates and times.
- If you want to allow for complex results, set CPXRES.
- Do the settings of CPXj and MULTX comply with your preferences? If not, change them.

Feel free to deviate from these suggestions for whatever reason. Having made your choices, call STOCFG to save them, preferably in a user variable.

## Alpha Input Mode: Introduction and Virtual Keyboard

This mode is designed for text entry, e.g. for keying in messages, prompts, and answers. It is entered via typically. Within AIM, ...

1. primary function of most keys will be appending the letter printed in grey bottom right of the respective key to $\boldsymbol{x}$ - see the virtual keyboard overleaf;
2. the menu Mya will pop up immediately in the menu section (unless another menu is open), comprising your favorite special characters or groups of them; ${ }^{187}$
3. $\square$ will lead to homophonic Greek letters. ${ }^{188}$

Upper and lower case are set by $\Delta$ and $\nabla$, respectively, also for the letters in Mya and CATALOG'CHARS'aINTL (see pp. 81 and 205f).

Wherever a default primary function is not primary anymore in AIM but continues being meaningful, it is reached via . E.g. $\square$ is required for appending a digit to $\boldsymbol{x}$. And $\square$ is also a shortcut to some special symbols, like $\square+$ calling $\pm$.

[^95]Two extra prefixes operate exclusively in AIM: $\square+\mathbf{R \downarrow}$ makes the next directly keyboard-accessible input character a subscript if provided, while $\square+E$ makes it a superscript. See the yellow arrows printed to the right of these keys, above the grey letters I and M.
 as reminders). Look up their contents in ReMS 2. Outside of AIM, you can call a.FN FBR to browse the entire character set provided.

## Alpha Input Mode：Entering Simple Text and More

Your WP43 features a large font mainly for numeric output and a small font for alphanumeric text strings．See here all＇small＇characters directly evocable through the virtual AIM keyboard shown on previous page：

## ABCDEFGHIJKLMNOPQRSTUVWXYZ

 abcdefghijklmnopqrstuvwxyz $\alpha \beta \gamma \delta \varepsilon \zeta \eta \vartheta ⿺ 𠃊 \lambda \mu \nu \xi \circ \pi \rho \sigma \tau \cup \varphi \chi \psi \omega$
0123456789＋－xor．／．，？ and the subscripts ABCDEFGHIJKLMNOPRRSTUVWXYZ
 as well as the corresponding superscripts ${ }^{\mathbf{A}} \ldots \mathbf{Z}^{\mathbf{a}} \ldots$ and $^{\mathbf{0}} \ldots{ }^{\mathbf{9}+}$ The 26 plain Latin letters can be also found in CATALOG＇CHARS＇aINTL （together with 73 more，supporting international com－
 munication）．Shortcut to alNTL is（A）in AIM．This catalog will remain open until closed explicitly by EXIT． The 24 plain Greek letters（plus 11 accented ones）are also found in CATALOG＇CHARS＇A．．．气．See the ReM．

So you may，for example，easily store and display an actual modern Greek message like

Actually，we could have written the major part of this OM just using said small font．It covers at least 47 languages from Afrikaans to Zhōngwén（see the ReM），providing the means that your display messages or prompts can be most easily read and understood by more than $50 \%$ of all mankind．

[^96]
## Example:

You can even spell Dèng Xiăopíng de famous and evidently successful slogan in Pinyīn straight ahead: ${ }^{190}$

Bùguăn bái māo, hēi māo, dàizhù lăoshũ jiù shì hão mão:

Taking advantage of this character set, it is also absolutely easy spelling e.g. French, Spanish, or German prompts correctly «en français ", "en Español", or "auf Deutsch", as well as text strings in many more languages using extended Latin alphabets.
Your WP43 supports you in climbing the very first step of politeness and respect by allowing you to adapt the software you write to the language your customers speak - instead of hacking in everything in English or using merely the very meager plain Latin letter set.

Two more menus ( $\alpha$ MATH, called via $\square+\square$, and $\underline{\alpha \bullet \text {, called via }, ~+a, ~}$ $\square+\square$ in AIM) contain further symbols and punctuation marks (cf. p. 204 and see the ReM):

Pressing USER in AIM toggles USERa. Individual symbols may be assigned to particular locations on the keyboard or within menus in AIM only (see pp. 307ff for how to do this). Such user assignments will become accessible when USER is set (indicated by $\mathbb{U}$ and A or $\alpha$ being both lit in the status bar).
Open alpha input can be deleted character by character using $\boldsymbol{\square}$ (like numeric input can be deleted digit by digit). If there is no character left, input will be canceled (cf. p. 25).
AIM is terminated by ENTERT (duplicating string $x$ in Y ) or EXIT unless pressed in a menu.

[^97]Example (continued):
Pressing EXIT with Dèng Xiăoping's slogan keyed in will display it in $\mathbf{X}$ as shown above.
Pressing ENTERT instead will display the text twice - fully in $\mathbf{X}$ and abbreviated to one line in $\mathbf{Y}$, showing only its first 7 words trailed by an ellipsis here:

> Bùguãn bǎi māo, hēi māo, dàizhù lăoshũ ... Bưguãn bǎi māo, hēi māo, dãizhù lăoshü jiù shì hăo māo !

Text strings exceeding two display lines will unveil their full contents after SHOW only.

## Combining Text Strings and Numeric Data

Due to the data type concept of your WP43, adding numeric data to a text string is as simple as pressing $\oplus$.

## Example:

Assume the two lowest stack registers look like this:

The train will arrive at
23:55

So here is a text string in $\mathbf{Y}$ and a time in $\mathbf{X}$. Pressing $\oplus$ now will combine $\boldsymbol{x}$ and $\boldsymbol{y}$, returning

The train will arrive at 23:55

So, $\boldsymbol{x}$ was converted to a text string, taking into account its present display format, and was appended to $\boldsymbol{y}$ (cf. the matrix on p. 75).

Let's enter a $2^{\text {nd }}$ string now by pressing $\alpha$ and the necessary letters, starting with a blank:

The train will arrive at 23:55
sharp at Victoria station.* sharp at Victoria station.

Now we have got text strings in $\mathbf{X}$ and $\mathbf{Y}$, so pressing $\boldsymbol{\pm}$ will append $\boldsymbol{x}$ to $\boldsymbol{y}$, returning:

The train will arrive at 23:55 sharp at Victoria station.

An alphanumeric string like this may contain up to 196 symbols in total (about five lines of text on display). Once numeric data (like the time above) became part of such a string, they are fixed and will not vary even if format is changed. ${ }^{191}$ Easy, isn't it?

## Working with Alphanumeric Strings

Your WP43 provides some more commands for dealing with such strings. You find them all in $\alpha$.FN:


## $\alpha \alpha, \mathrm{FN}$ <br> $\sqrt{\mathrm{x}}$ F

$X \rightarrow \alpha$ destination converts a character code in $\mathbf{X}$ to the corresponding symbol and appends it to the destination; the character code is saved in $\mathbf{L}$.

If $\mathbf{X}$ contains a text string, the entire string is appended to the destination.

If $\mathbf{X}$ contains a matrix, $x \rightarrow \alpha$ uses each element in the matrix as a character code or text string. $x \rightarrow \alpha$ starts with the $1^{\text {st }}$ element $(1 ; 1)$ and continues following usual reading habits until reaching the end of the matrix.

[^98]$\alpha \rightarrow x$ source converts the leftmost symbol in the source string to the corresponding code, removes this symbol from said string, and pushes its code on the stack. If the source is empty, $\alpha \rightarrow x$ returns zero.
$\alpha$ RL source rotates the source string by $\boldsymbol{x}$ symbols to the left.
$\alpha$ RR source rotates the source string by $x$ symbols to the right.
$\alpha$ SL source shifts the source string by $x$ symbols to the left, deleting the first $\boldsymbol{x}$ symbols from the string.
$\alpha$ SR source shifts the source string by $x$ symbols to the right, deleting the last $x$ symbols from the string.
$\alpha$ LENG? source pushes the length of the source string on the stack.
$\alpha$ aPOS? source searches the string in the source for the target symbol or string given in $\mathbf{X}$. If a match is found, aPOS? returns in $\mathbf{X}$ the position number where the target was found starting in the source (counting the leftmost symbol as position 0); else it returns -1. The target is saved in $\mathbf{L}$.

## FBR

 shows all characters defined in both fonts provided.You can also compare text strings using commands found in TEST, e.g. to create a sorted list. String $\boldsymbol{A}$ is called "smaller" than string $\boldsymbol{B}$ if $\boldsymbol{A}$ precedes $\boldsymbol{B}$ in sorting.

Nevertheless, do not forget that your WP43 is mainly designed as a programmable calculator. Please turn overleaf to see what could be performed with such devices.

## HP-65 in space with Apollo-Soyuz

The American astronauts calculated crifical course-correction maneuvers on their HP-65 programmable hand-held during the rendezvous of the U.S. and Russian spacecraft.

Twenty-four minutes before the rendeavous in spece, when the Apollo and Soyuz wete 12 miles tport, the American astrensust corfected their course to place their sjacecraff inte the same orbit as the Russian cruft. Twelve mimutes later, they made a second poritioning maneuver jost priot to braking and ewasted in to liekher.
In both cases, the Apolle astronauts made the course-correction calculations on their HP-65. Had the on-board opmputer falled, the spacecraf not being in communication with grount statioms af the time, the HP-6S would have been the only way to maks all the critical calculations. Using comples programs of nearly 1000 steps written by NASA crientists and pro-recorded on marnetic proptam cards, the astronants made the calculations antomatically, quickly, and with ten-dieit accuracy.
The HP-65 aleo served as a backup for Apollo's on-board computer for two earlier manemvers. Its anvwert providnd a comfidence-boosting doublecheck on the coelliptic (\#5 mite) maneuver, and the terminal phase initiation ( $\mathbf{2 z}$ mile) maneuver. which placed Apollo on an intercept trajectory with the Rusilan craft.
Periodically throughout their joint mission, the Apollo astronauts also used the HP-65 to calculate

## HEWLETT hp PACKARD

Salos and sarvice from the ellices in 65 seuntries
how to peint a high-gain antenna precisely at an orbiting satellite to assure the best possible ground camminleatlons
Thefint fully programmable hand-held calculatot, the HP65 automatically steps through lengthy or repetitive calculations. This advanced instrument relieves the user of the need to remember and exe cute the correct sequence of krystrokes, using programs recorded 100 steps at a time on tiny magnetic cards. Each program consists of any combination of the calculator's $5 t \mathrm{kej}$-strole functlons with tranching. logical comparison, and conditional skip instructions.
The HP-65 is priced at $5795^{\circ}$. See it, and the rest of the HIP family of professional hand-helds at quality departitent itores of campus boolstores. Call n00-535-7022 (In Callfornia, 800-662-9362) for the name of the retailer nearest you.


An advertisement of 1975 (above) and one of the Nineteeneighties (overleaf). Compare the capabilities of your WP43. Imagine the opportunities.


## SECTION 3: PROGRAMMING

Your WP43 is a powerful keystroke-programmable calculator. If already this statement makes you smile with delight, this section is for you. Else we will bring a smile on your face by mentioning the following facts:

Your WP43 allows you to store a sequence of keystrokes like you would use them to solve a problem manually; this is to save you time on repetitive calculations (cf. pp. 22ff). Once you have written the keystroke procedure (or routine) for solving a particular problem and recorded it in your WP43, you need no longer devote attention to the individual keystrokes that make up the procedure. Instead, you can let your WP43 solve each similar problem for you. And because you can easily check the routine stored, you have more confidence in your final answer since you do not have to worry each time about whether or not you have pressed an incorrect key. Your WP43 performs the drudgery, leaving your mind free for more creative work.

And even better: you may use program memory for storing many routines. For telling your WP43 where a routine begins and ends, each one is confined by two steps: It starts with LBL (for LaBeL) and typically ends with RTN (for ReTurN) - cf. p. 23. These two steps separate it from the other routines you may add for other tasks. And LBL puts a label on your routine so you can find and call it when you want it to be executed.

You may structure program memory even more: Collect some routines and separate them by END from other routines or sets of routines. What is found between two END steps is called a program. Programs are the basic building blocks within program memory. Think of the beginning and end of the entire used program memory section containing implicit END steps. ${ }^{192}$ So even with program memory cleared, there will be at least one program within at any time.

Within routines, you may store any sequence of keystrokes (commands, operations, objects). Choose any operation featured - almost all of them are programmable. The commands in your routine may also access each and every register, variable, or flag provided - there are almost no limits.

[^99]You are the sole and undisputed master of the memory. ${ }^{193}$

Each such routine itself may contain one or more subroutines. Also subroutines start with LBL and typically end with RTN. Actually, subroutines may look exactly like routines: the only difference is that a subroutine is called from another routine, while a main routine is called from the keyboard. Thus we do not need differentiating these two kinds of routines further on.

Enough of theory - press RTN and switch to program entry mode (PEM) via (P/R. The display of your WP43 will change to something like this:


LBLRTN GTO FLAG 9 - XEQ

## EA SF

This screen shows the example you entered on pp. $23 f$ (the status bar on top will differ according to your time and settings). In the section of the screen used for numeric output so far, the first 6 steps of the $1^{\text {st }}$ program in program memory are shown. Label steps are 'outdented' for visual structuring. The position of the program pointer (the current step) is highlighted by inversion; the routine the program pointer is in is called the current routine; the corresponding program is the current program. In this example here, the top view of P.FN is shown in the menu section.

193 This freedom has a price: Take care that your routines do not interfere with each other in their quest for data storage space. It is good practice to record the global registers, variables, and user flags a particular routine uses, and to document their purposes and contents for later reference.
An alternative - using local registers and flags - will be explained further below.

On the other hand, switching to PEM the very first time after unpacking your WP43 (or after CLPALL or resetting it), the display will look like this instead most probably:

## Recording a New Routine

Whenever you want to enter a new routine, switch to PEM using (unless you are already in); then start with pressing GTO... These keystrokes will bring you to the very end of the used section of program memory, so you can start keying in your new routine right there without interfering with anything coded previously.

Start the new routine with LBL giving it a unique label (cf. p. 59). Then press the keys as you would do in manual problem solving (cf. pp. 22ff). Each new step will be inserted right after the current step. You find the following programming keys all bottom right on your keyboard as shown on previous page:

- LBL for LaBeLing a routine or a program step following,
- XEQ for eXECUting or calling a specific routine,
- RTN for ReTurNing to the caller of the current routine,
- GTO for unconditionally Going TO a specified label (i.e. positioning the program pointer to the respective LBL step),
- R/S for Running or Stopping the current routine,
- $\triangle$ and $\nabla$ (or $\overline{\underline{E}} \triangle$ and $\overline{\underline{E}}$ if there is a multi-view menu displayed) for browsing program steps,
- P.FN END for finishing the current program,
- $\mathbb{P} / R$ for toggling $\underline{P E M}$ and $\underline{R} U M$,
- EXIT for exiting PEM returning to RUM unless a menu is open, continued to the left by the menus for LOOPs, TESTs, and PARTs.


## Further programming commands are collected in P.FN.



## Example (from the HP-15C OH):

Mother's Kitchen, a canning company, wants to package a ready-to-eat spaghetti mix containing three different cylindrical cans: one of spaghetti sauce, one of grated cheese, and one of meatballs. Mother's needs to calculate the base areas, total surface areas, and volumes of the three different cans. It would also like to know, per package, the total base area, surface area, and volume.

## Solution:

The program to calculate this information uses these formulas and data:

$$
\begin{aligned}
& \text { base area }=\pi r^{2} \\
& \text { volume }=\text { base area } \times \text { height }=\pi r^{2} h \\
& \text { surface area }=2 \times \text { base area }+ \text { side area }=2 \pi r^{2}+2 \pi r h
\end{aligned}
$$

| $\boldsymbol{r}$ | $\boldsymbol{h}$ | Base Area | Volume | Surface Area |
| ---: | ---: | :---: | :---: | :---: |
| 2.5 | 8.0 | $?$ | $?$ | $?$ |
| 4.0 | 10.5 | $?$ | $?$ | $?$ |
| 4.5 | 4.0 | $?$ | $?$ | $?$ |

## Method:

1. Enter an $\boldsymbol{r}$ value into the calculator and save it for other calculations. Calculate the base area ( $\pi r^{2}$ ), store it for later use, and add the base area to a register which will hold the sum of all base areas.
2. Enter $\boldsymbol{h}$ and calculate the volume $\left(\pi r^{2} h\right)$. Add it to a register to hold the sum of all volumes.
3. Recall $\boldsymbol{r}$. Divide the volume by $\boldsymbol{r}$ and multiply by $\mathbf{2}$ to yield the side area.
4. Recall the base area, multiply by 2, and add to the side area to yield the surface area. Sum the surface areas in a register.

Do not enter the actual data while writing the program - just provide for their entry. These values will vary and so they will be entered before and / or during each program run.
Switch to PEM using (P/R and go to free memory using GTO®. Key in the
following program to solve the above problem: ${ }^{194}$

| LBL (K) ENTER $\mathbf{T}$ | LBL 'K' |  |
| :---: | :---: | :---: |
| STO $\alpha$ R ENTERT | STO 'r' | Store radius |
| EXP $\mathrm{x}^{2}$ | $\mathrm{x}^{2}$ |  |
| $\pi$ | $\pi$ |  |
| ( | $\times$ | Compute base |
| STO $\propto \triangle$ BASE ENTERT | STO 'bASE' |  |
| STO $\rightarrow \square \triangle$ S B ENTERT | STO+ ' $\Sigma$ B' | Sum of bases |
| VIEW VAR (BASE) | VIEW 'BASE' | Show base for 1 s |
| P.FN INPUT $\boldsymbol{\alpha}$ (H) ENTERT | INPUT 'h' | Ask for height input |
| RCL VAR (BASE) | RCL 'BASE' |  |
| - | $\times$ | Compute volume |
| STO $\propto \triangle$ VOLUME ENTER $\boldsymbol{1}$ | STO 'VOLUME' |  |
|  | STO+ ' $\Sigma \mathrm{V}$ ' | Sum of volumes |
| VIEW VAR (VOLUME) | VIEW 'VOLUME' | Show volume for 1 s |
| $\operatorname{RCL}$ VAR ( $r$ ) | RCL 'r' |  |
| (1) | 1 |  |
| (2) | 2 |  |
| - | $\times$ | Compute side |
| RCL VAR (BASE) | RCL 'BASE' |  |
| (2) | 2 |  |
| - | $\times$ |  |
| $\pm$ | + | Compute surface |
| STO $\pm \boxed{\alpha} \square \mathrm{S}^{(1)}$ ENTERT | STO+ ' $\mathbf{S S}$ ' | Sum of surfaces |
| RTN | RTN | End of routine |
| P/R |  | Leave PEM |

Now, let's run the program (...):

GTO K
CLR CLCVAR
makes K the current program.
clears all variables of the current program. By doing so, it creates the variables not defined yet, avoiding later errors caused by undefined variables.

DISP FIX 01

[^100]| 2.5 | 2.5 |  | $1^{\text {st }}$ can: radius ${ }^{195}$ |
| :---: | :---: | :---: | :---: |
| (EEQ (K) | BASE $=19.6$ |  |  |
|  | h? | 0 | ask for height input |
| 8 (R/S | volume $=157.1$ |  | show volume for 1 s |
|  |  | 164.9 | surface |
| 4 | 4- |  | $2^{\text {nd }}$ can: radius |
| (XEQ ( ${ }^{\text {K }}$ | BASE $=50.3$ |  |  |
|  | h? | 8 | ask for height input |
| 10.5 R/S | voLuME $=527.8$ |  | show volume for 1 s |
|  |  | 364.4 | surface |
| 4.5 | 4.5 |  | $3^{\text {rd }}$ can: radius |
| (XEQ (K) | BASE $=63.6$ |  |  |
|  | h? | 10.5 | ask for height input |
| 4 R/S | voLume $=254.5$ |  | show volume for 1 s |
|  |  | 240.3 | surface |
| (RCL) VAR (EB) |  | 133.5 | sum of bases |
| RCL) VAR (IV) |  | 939.3 | sum of volumes |
| RCL VAR (IS) |  | 769.7 | sum of surfaces |

If you want to run this routine again for another set of cans, remember to clear the variables $\mathbf{\Sigma B}, \mathbf{\Sigma V}$, and $\boldsymbol{\Sigma S}$ before. CLCVAR will do this for you most easily.

The preceding program illustrates the basic techniques of programming. It also shows how data can be manipulated in PEM and RUM by entering, storing, and recalling data (input and output) using ENTERT, STO, RCL,

[^101]store arithmetic, and programmed I/O.
See the next chapters and the $I O I$ for comprehensive information about all the commands used in this example. ${ }^{196}$

## Labels

As mentioned above, each routine or subroutine begins with a LBL step. Structuring program memory and jumping around within is eased by those labels. You may tag labels not only to the $1^{\text {st }}$ but to any step in your routine. Your WP43 allows for specifying a wide variety of alphanumeric labels as described overleaf.

Whenever a step like e.g. GTO labl is encountered (with labl representing an arbitrary label) with a program running or a single step executed in RUM, your WP43 will search this label using one of the two following methods:

1. If labl is plain numeric ( $00 \ldots 99$ ) or $\mathbf{A} \ldots$ E, it will be searched forward from the current position of the program pointer. When an END step is reached without finding labl so far, the quest will continue right after previous END (so the search will stay in the current program). This is the procedure for local labels. So, local labels are valid in the current program only and may hence be reused in another program.
2. If, however, labl is an alphanumeric label of $\leq 7$ characters of arbitrary case (automatically enclosed in ' like ' $M$ ' or 'Ab1'), searching will start at the end of program memory, go upwards, and cover the entire program memory independent of the current position of the program pointer. This is the procedure for global labels. ${ }^{197}$

So, global labels can be accessed from anywhere in memory while local labels can only be accessed from within their own program.

[^102]Addressing labels, on the other hand, follows the rules given below:


SLV ' $F 1 \mu$ ' solves the function given in the routine $\mathrm{F} 1 \mu$ (see pp. 262ff).
$\int f d \rightarrow \mathbf{T} \quad$ integrates the function given by PGMINT over the variable whose label is on stack register $\mathbf{T}$ (see pp. 269ff).

## XEQ $\rightarrow 44$ calls and executes the routine whose

label is found in R44.

[^103]Furthermore, GTO provides four special cases: see GTO $\uparrow$, GTO $\downarrow$, GTO., and GTO.. in next chapter. - And remember TAM is set also during label addressing, so the virtual keyboard of your WP43 will look like this:

You can access the local labels A - E directly as well as the global labels $\mathrm{G}-\mathrm{N}, \mathrm{R}, \mathrm{T}, \mathrm{V}$, and $\mathrm{X}-\mathrm{Z}$, all 19 also in lower case (take $\boldsymbol{\nabla} / \Delta$ to switch case); furthermore the registers A-D, I-L, T, and $\mathrm{X}-\mathrm{Z}$. This allows for calling 14 programs with just two keystrokes each, and for calling five subroutines in one program.

Note also the changed assignment of F2 compared to p. 60. Thus, instead of keying in a longer global label in AIM, pressing PROG and selecting the label from this
 submenu (comprising all global labels defined at execution time) may be easier and faster.

## Editing a Routine

Whenever you want to edit (correct, expand, modify, shorten, etc.) an existing routine, start with ensuring you are in RUM - then enter GTO labl . This will position the program pointer onto the corresponding LBL step (as explained on p. 218). Then switch to PEM using $\mathbb{P} / R$ and start browsing from this LBL step.

## Example:

Let's browse the program steps in our example routine. Press GTO $\mathbf{T},(\mathbb{P} / \mathrm{R}$, and then three times $\mathbf{\nabla}$ :

```
0001: LBL 'T'
0002: ENTER^
0003: x
0004: \pi
0005: x
0006: RTN
0007: END
```

Unless you are next to the very beginning or end of a program, the program pointer will always be placed in the middle of the display with three steps displayed above and three steps below of it.

Navigating in program memory, you may execute various actions. If you want to, e.g. ...

- continue browsing forward, press $\nabla$ (or $\overline{\underline{E}}$ 保 a multi-view menu is displayed); when reaching the END, browsing will start with the $1^{\text {st }}$ step of the same program again.
- browse backwards, press $\boldsymbol{\Delta}$ (or $\overline{\underline{E}}$ if a multi-view menu is displayed); when reaching program top, browsing will stop.
- delete a program step, go to said step (make it the current step by positioning the program pointer on it), then press $\boldsymbol{-}$. It will vanish (this cannot be undone!) and the pointer will move on the step before.
- insert something, go to the program step before, and then press the keys to be inserted after it.


## Example (continued):

Replace ENTER $x$ in our routine above by $x^{2}$ doing the same.

## Solution:

Press $\triangle$ The program pointer is on step 0003: $\mathbf{x}$ now.
This step vanished;
the program pointer is on step
This step vanished;
0002:
the program pointer is on step
0001: LBL 'T' now.

EXP $\mathbf{x}^{2}$ This command will be stored in step 2 while $\pi$ moved to step 3 .

- jump to a particular global label (without inserting a GTO in the current routine), press GTO $\propto$ label ENTERT or GTO. PROG (label); for a local label, press GTO $\square \boldsymbol{n n}$ instead; 1-letter labels from $\mathbf{A}$ to $\mathbf{E}$ can be accessed e.g. via GTO. (D).
- jump to the top (i.e. the $1^{\text {st }}$ step) of next program, press
- jump to the top of current program, press GTO $\Delta$. If the program pointer is there already, GTO $\Delta$ will bring it top of previous program.
- start writing a new routine, press GTO D. then LBL... (as experienced in Mother's Kitchen on pp. 215f).

That's almost all. When you are done, press (P/R or EXIT to leave PEM, returning to RUM.

Note your WP43 cannot know if all necessary preconditions for a certain program step will be fulfilled at the time of its execution. Thus, respective error messages can show up no earlier. Debugging may be required.

## Running a Routine from the Keyboard (also for Debugging)

Whenever you want to execute an existing routine, ensure you are in RUM. Then there are three alternatives:

1. Automatic execution of the current routine: Press RTN to return the program pointer to the $1^{\text {st }}$ step of the current routine. Then press R/S. This will run this routine, i.e. start executing the following steps automatically until an error happens, a STOP, a final RTN, or an END will be encountered ${ }^{202}$ where it will halt and display $\boldsymbol{x}$.
2. Automatic execution of a selected routine: Press XEQ and specify the global label of the program you want to execute (or press PROG and pick it from the menu). ${ }^{203}$ This will move the program pointer to the corresponding LBL (cf. p. 218) and start executing the following steps automatically until an error happens, a STOP, a final RTN, or an END

[^104]will be encountered ${ }^{202}$ where it will halt and display $\boldsymbol{x}$ (cf. pp. 22f).
3. Stepwise execution of a selected routine: Press GTO and specify the global label of the program you want to execute (or press PROG and pick it from the menu). This will move the program pointer to the corresponding LBL (cf. p. 218) and wait. Each following program step will then be executed one at a time as you press $\boldsymbol{\nabla}$ or $\overline{\underline{E}}$ : pressing $\nabla$ or $\bar{E} \overline{\text { V }}$ will display the current step at left end of the top numeric row, releasing it will execute it. ${ }^{204}$ Where your routine asks for an input (see pp. 232ff), close this input with $\nabla$ or 五 $\bar{\nabla}$. Reaching the end of the current routine, $\nabla$ or $\overline{\underline{E} \nabla}$ will return to its $1^{\text {st }}$ step. Following this procedure, you will go through the routine as in normal execution but significantly slower - and you may call RBR or STATUS or perform additional checks after each program step. ${ }^{205}$ This procedure is especially useful for debugging.

If an error occurs while a routine is running, it stops immediately at the step generating said error and throws the corresponding error message (see the ReM, App. C, for a list of all error messages provided). Press any key to clear this temporary information. To view the program step leading to said error thrown, press

## Subroutines: Running a Routine from another Routine

The command XEQ is programmable as well. Whenever a running routine encounters an XEQ, it will search for the associated label as described on p. 218, go to it, and continue program execution with the step after this LBL until it encounters a RTN. This will return the program pointer to the step right after said XEQ where execution will continue. Compare the picture overleaf where routine $\mathbf{A}$ calls routine $\mathbf{B}$.

You can also nest subroutines - your WP43 can remember as many pending subroutine return addresses as RAM allows. Note every (sub-)

[^105]Main program
(top level)


End of program
routine may allocate its own local registers, up to 98 each (see p. 247).

But all return addresses will be lost for the current program if you alter the program pointer while this program is stopped - pressing R/S, EV, or $\nabla$, however, will not cause a loss of return addresses.

If you need any of your subroutines elsewhere in another routine then you can call it again at

## Main program

 (top level)$\left.\begin{array}{|cc|}\hline \text { LBL A } \\ \vdots \\ \text { XEQ } & \text { B } \\ \text { SIN } \\ \vdots \\ \text { RTN }\end{array}\right)$

End of program
no expense of memory. If you want to call a particular subroutine from another program than the one it is defined in, the label of said subroutine must be global.

## Automatic Binary Testing and Conditional Branching

So far, we were talking about linear programs, running straight from their beginning (LBL) to end (final RTN or END). Your WP43 can do more for you: like other keystroke-programmable calculators before, it features a set of binary tests checking various calculator states. Most of the binary tests provided are collected in TEST. There are also two tests in BIT, and
 8 tests of flags are stored in FLAG. Names of binary test commands contain a '?', most times as their last symbol. Generally, binary tests will return true or false as temporary information at left end of the $\mathbf{Z}$ numeric row if called from the keyboard. If called in a running routine
automatically instead, they will execute the next program step if the test is true at execution time, else skip that step. So the general rule reads "do if true". Think of the step following the test containing a GTO and you see how conditional branching comes into play.

## Example:

```
0020: x\leq? Y If this test is true (i.e. if }x\mathrm{ is not greater than }y\mathrm{ )
0021: GTO 'Join'
0022: x<y
:-
0032: LBL 'Join'
0033: ln x
...
```

Most binary tests operate on $\boldsymbol{x}$. Six tests check its data type:

- REAL? tests if $\mathbf{X}$ contains a real object (DT 2 or 8 ) and executes the next program step if true, else skips it.
- CPX? tests if $\mathbf{X}$ contains a complex object (DT 3 or 9 ) ...
- MATR? tests if $\mathbf{X}$ contains a matrix (DT 8 or 9 ) ...
- STRI? tests if $\mathbf{X}$ contains a text string (DT 7) ...
- SPEC? tests if $\boldsymbol{x}$ is special (i.e. or 'Not a Number') ...
- NaN? tests if $\boldsymbol{x}$ is 'Not a Number' ...


## Five check its numeric content:

- INT? tests if $\boldsymbol{x}$ is integer (note INT? will return true if FP? returns false, i.e. for $\boldsymbol{x}$ being of $D T 1,10$, or 3 with $\operatorname{FP}(\boldsymbol{x})=0$ ) ...
- EVEN? tests if $\boldsymbol{x}$ is integer and even ...
- ODD? tests if $\boldsymbol{x}$ is integer and odd...
- FP? tests if $x$ has a nonzero fractional part (cf. INT?) ...
- PRIME? tests if the absolute value of the integer part of $\boldsymbol{x}$ is a prime number and executes the next program step if true, else skips it.

Seven tests compare its numeric content with 0,1 , or the content of a source specified (let's call it $\boldsymbol{s}$, cf. also pp. 61 and 63):

- $\mathbf{x}<$ ? tests if $\boldsymbol{x}$ is less than $\boldsymbol{s}$ and executes the next program step if true, else skips it.
- $x \leq ?, x=$ ?, $x \neq ?, x \geq$ ?, and $x>$ ? work in analogy to $x<?$.
- $x \approx$ ? tests if the rounded values of $x$ and $s$ are equal and executes the next program step if true, else skips it.


## Four check its internal structure:

- BC? (or BS?) tests if $\mathbf{X}$ contains a short integer, then checks its bit specified and executes the next program step if said bit is clear (or set), else skips it.
- LEAP? tests if $\mathbf{X}$ contains a date, then extracts the year and tests for a leap year ...
- M.SQR? tests if $\mathbf{X}$ contains a matrix, then checks if it is square ...

Eight flag tests operate on the flag specified:

- FC? (or FS?) tests this flag and executes the next program step if said flag is clear (or set), else skips it.
- FC?F (FS?F) works as FC? (FS?) but flips this flag after testing (i.e. clears it if it was set or sets it if it was clear).
- FC?C (FS?C) works as FC? (FS?) but clears this flag after testing.
- FC?S (FS?S) works as FC? (FS?) but sets this flag after testing.


Finally, there are four special tests:

- LBL? tests for the existence of the label specified, anywhere in program memory.
- TOP? will return true if the program pointer is in the top level routine (cf. sketch on p. 223).
- ENTRY? checks the (internal) entry flag. This is set if:
- a character is entered in AIM or
- a command is accepted for entry (be it via ENTERT, a function key, or R/S with a partial command line).

ENTRY? is useful e.g. after PAUSE.

| $\square$ | $12$ | $13$ | $\overline{14}$ | $\frac{\square}{15}$ | $\square$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| (1/x | EXP | TRI) | (1n) | $e^{x}$ | ( $\times$ |
| 21 | 22 | 23 | 24 | 25 | 26 |
| STO | RCL | Rt | CC | (1) | 9] |
| 31 | 32 | 33 | 34 | 35 | 36 |
| $\begin{gathered} \text { ENTER } \\ 41 \end{gathered}$ |  | $\begin{gathered} x^{2} y \\ 42 \end{gathered}$ | $\begin{aligned} & +/, / \\ & 43 \end{aligned}$ | E | 4 |
|  |  | 44 |  | 45 |
| (1) | 7 |  | 8 |  | 9 | (XEQ |
| 51 | 52 | 53 |  | 54 | 55 |
| X | 4 | 5 |  | 6 | 4 |
| 61 | 62 | 63 |  | 64 | 65 |
| - | 1 | 2 |  | (3) | V |
| 71 | 72 | 73 |  | 74 | 75 |
| $\pm$ | 0 | $\square$ |  | R/S | EXIT |
| 81 | 82 | 83 |  | 84 | 85 |

- KEY? tests if a key was pressed while a routine was running or paused. If no key was pressed in that interval, the next program step after KEY? will be executed (exception!); else it will be skipped and the code of said key will be stored in the address specified. Key codes reflect the rows and columns on the keyboard (see the picture and p. 235 for an application).

See the IOI for more information about all binary tests provided, also beyond those mentioned above.

Tose There are further commands also featuring a trailing '?' but returning numbers (e.g. WSIZE?) or codes (e.g. KTYP?) instead of true or false - you will find these commands in INFO. Turn to the ReM for information about them.

As mentioned above, routines end with RTN (typically) and programs with END. Executing a program, both RTN and END work in a very similar way and show only subtle differences: An RTN immediately after a binary test returning false will be skipped - an END will not.

## Loops and Counters

The commands DSE, DSL, DSZ, ISE, ISG, and ISZ (all contained in LOOP) are for controlling loops in routines. Each of them Decrements or Increments a counter in a register or variable as specified, tests the result, and executes or skips the following program step. See the picture below illustrating ISG (Increment and Skip if Greater), DSE (Decre-

## LOOP TEST

 4 Z ment and Skip if Equal), ISE (Increment and Skip if Equal),, and DSL (Decrement and Skip if Less). ${ }^{206}$ The control variable contains the start value of a counter ccccc, its 3-digit final value fff, and a 2-digit increment/decrement ii. If $\boldsymbol{i j}$ is entered as zero, it defaults to 1. (The commands ISZ and DSZ simply skip if Zero - anything between -1 and +1 is assessed as 0 here.)

With GTO placed in the skipped step pointing to a label upstream in the same routine you can create loops running until the specified condition is met - see the examples below. E.g. if the loop counter should go from 5 to 1 , use 5 STO K ... DSE (K) on the other hand, if it should go from 1 to 5 , use 1.005 STO K ... ISG K).

Without such an exit condition you can deliberately create an infinite loop. Note such loops are allowed by the operating system of your WP43. Such a loop will run until you interrupt it manually by EXIT or R/S, or until battery voltage falls below the limit (your WP43 will not turn off automatically with a program running!)

[^106]Infinite loops created inadvertently are difficult to recognize in a running program unless outputs are implemented in the loop - just 䨘 will stay lit in the status bar as long as the program is running and your WP43 will accept no keystrokes except EXIT or R/S - it may look like caught in a hung state. Thus, implementing some outputs is recommended in regular intervals especially in programs not completely checked yet.

Let's demonstrate loops, use of flags, tests, and also indirect addressing:

## Example 1:

The so-called 'Cullen numbers' are $C_{n}=n 2^{n}+1$. It is proven that no $C_{n}$ is prime for $n \leq 1000$ but very few cases. One is $C_{1}$. What else do you find?

## Solution:

Reset the program pointer to the start of free program memory by GTO. $\square$. Switch to programming via $(P / R)$ and key in:


Now, load the loop counter: 1.999 STO (1) (999 is the maximum final count possible) and start the program by pressing XEQ PROG ( $\mathbf{C n}$ ). It will display 1 as it should. Press R/S. What will be shown next? And thereafter?

Example 2 (following an idea of Gene Wright):
Write a little routine to store random numbers in R25 through R39.

## Solution:

Enter GTO $๑(P / R$ and key in:

| LBL X | LBL ' X ' |  |
| :---: | :---: | :---: |
| PROB $\triangle$ RAN\# | RAN\# | Get a new random number and.. |
| STO $\rightarrow$ 2 4 | STO $\rightarrow 24$ | store it where r24 is pointing to. |
| LOOP ISG 24 | ISG 24 | Increment the pointer. |
| GTO X | GTO ' X ' | If $\boldsymbol{r} 24 \leq 39$ then return to label $\mathbf{X}$ |
| RTN | RTN | else finish this routine. |

## P/R

Initialize the loop counter via 25.039 STO (2) (4) and start the program by pressing XEQ X. It will stop with the last random number in display. Check the target registers using RBR.

Example 2 (continued):
Now, write a routine to sort these 15 stored numbers so the smallest number moves to the register with the smallest address.

## Solution:

We will use the so-called 'bubble sort' algorithm. Re-initialize the loop counter via 25.039 (STO (2)4] ( $r 24$ was modified by program X above). Enter

## GTO 9

P/R

| LBL (Y) | LBL ' Y ' |  |
| :---: | :---: | :---: |
| P.FN LOCR 2 ENTERT | LocR 02 | Allocate 2 local registers. |
| LBL (A) | LBL A |  |
| RCL 24 | RCL 24 | Put the start pointer $\boldsymbol{r} 24$. |
| STO O0 | STO . 00 | into local register 00; |


| LOOP INC X | INC X | increment this pointer and |
| :---: | :---: | :---: |
| STO OTD | STO . 01 | store it in local register 01. |
| CF 100 | CF . 00 | Clear local flag 00. |
| LBL (C) | LBL C |  |
| RCL $\rightarrow 00$ | RCL $\rightarrow$. 00 | Recall the contents of the registers |
| $\underline{\mathrm{RCL}} \rightarrow 0 \mathrm{O}$ | RCL $\rightarrow$. 01 | where $r .00$ and r. 01 are pointing to. |
| (TEST) $\mathrm{x}<$ ? Y | $\mathrm{x}<$ ? Y | Is $x<y$ ? |
| GTO B | GTO B | Then go to local label B |
| LBL (D) | LBL D | else. |
| LOOP ISG ©00 | ISG . 00 | increment r. 00 and.. |
| ISG 901 | ISG . 01 | if $r .00 \leq 39$ increment $r .01$ and... |
| GTO (C) | GTO C | if $(r .00>39$ or $r .01 \leq 39)$ then return to |
| FLAG FS? ©0 | FS? . 00 | else check local flag 00: |
| GTO A | GTO A | if set, return to local label A, |
| RTN | RTN | else stop this routine. |
| LBL B | LBL B |  |
| SF 00 | SF . 00 | Set local flag 00. |
| STO $\rightarrow$ O0 | STO $\rightarrow$. 00 | Store the smaller value ... |
| $x^{2} \times 2$ | $\times$ \% | where r. 00 and the greater ... |
| STO $\rightarrow$ OT0 1 | STO $\rightarrow$. 01 | where $r .01$ is pointing to. |
| $x^{2} \geqslant y$ | $x$ \% | Restore the stack and. |
| GTO D | GTO D | return to local label D. |
| P.FN END | END |  |
| P/R |  |  |

Start the program by pressing XEQ Y. Then check the target registers using RBR. You will find the smallest value in R25, a greater one in R26, etc., up to the greatest in R39.
Note this program allocates two local registers for its exclusive use (R. 00 and R.01). Furthermore, it uses one local flag and four local labels.

The following sorting program is even shorter (kudos to Jean-Marc Baillard). Start it by $\mathbf{2 5 . 0 3 9}$ XEQ $\mathbf{Z}$. Finding out how it works is left as an exercise for the reader.

```
LBL 'z'
    sign
LBL A
    RCL L
    RCL L
    RCL ->L
LBL B
    RCL }->\textrm{Y
    x>? Y
    GTO C
    x%
    RCL L
    +
LBL C
    R
    ISG Y
    GTO B
    x* 
    STO ->Z
    ISG L
    GTO A
END
```

Cf. HP-42S OM, pp. 152 - 154.

## Programmed User Interaction and Dialogues

A number of commands are provided for controlling the interaction of programs with you. A program shall output a result to you at least, and it may also ask for your input. In the IOI, the behavior of those I/O commands is described if they are entered from the keyboard. Executed by a program, however, they will work differently.

When you start a program by XEQ or R/S, the letter P will be lit in the status bar. While in RUM each command executed may change the display immediately, only INPUT, PAUSE, STOP, and VIEW will update the screen with a program running. VIEW will display for 1 s , the others will display until the next of these four commands is encountered or the program stops. For programmed I/O, see the following examples:

- Take VIEW for displaying intermediate results. Specify any register or variable you want as source of information - also $\mathbf{X}$ is a valid parameter of VIEW. The name of the source will label the output.

Frequent display updates will slow down program execution.

- Use


## VIEW $x y z$

PAUSE $n$ for showing output for a defined time interval of $\frac{n}{10} S$.

- If you have a printer connected, you may send your program output thereto as well. Turn to pp. 246ff for printing.
- Ask ('prompt') for numeric input employing

| VIEW $\mathbf{x y z}$ | Update display showing the register or variable $\mathbf{x y z}$ |
| :--- | :--- |
| STOP | ... and wait for user reaction, finished by R/S. |
| STO $\mathbf{x y z}$ | Store what the user entered. |

A stop sign will be displayed in the status bar when the program runs on a STOP. Whatever you will key in will be put into $x y z$ when you continue program execution by $R / \mathbf{S}$.

More elegant is using INPUT for this task:

```
INPUT xyz does the same in just one step.
```

- Prompt for alphanumeric (string) input using the following steps:

| SF ALPHA | sets AIM for upcoming input. <br> RCL $\boldsymbol{x y z}$ |
| :--- | :--- |
| displays the register with the message string.  <br> STOP waits for your input. Whatever you key in now is <br> appended to $\boldsymbol{x}$ when you continue by pressing R/S  |  |
| CF ALPHA returns to the numeric mode previously set. <br> STO $\boldsymbol{y Z X}$ stores $\boldsymbol{x}$ to wherever you like. |  |

Again, more elegant is using INPUT for this task:

| SF ALPHA | sets AIM for upcoming input. |
| :--- | :--- |
| INPUT $x y z$ |  |
| CF ALPHA | returns to the numeric mode previously set. |

- If you need to enter values for several variables then the following way is most efficient (although it may look lengthy here):

LBL 'Var.In'
MVAR 'xy1'
MVAR 'xy2'
MVAR 'xy3'
VarMNU 'Var.In'
STOP
EXITall
RCL 'xyz'
we will need this label for VARMNU later.
creates a menu for the variables defined immediately after 'Var.In' and shows it. stops for user interaction. exits the menu when program continues. recalls what you need first (it may have been entered in any order).

The label called 'Var.In' here should be located close to the program top. It may be followed by up to 18 MVAR steps defining your variables required. When the program encounters the step VarMNU, it will set up a menu for these variables and display it. In our example, this would look like

> | $x y 1$ | $x y 2$ | $x y 3$ |
| :--- | :--- | :--- |

As long as this menu is displayed and you want to...

- write a new value into one of these variables, key in the value or calculate it, then press the softkey corresponding to your target variable, and $\boldsymbol{x}$ will be stored in it.
- recall the present value of one of these variables, press RCL and then the corresponding softkey.
- view the present value of one of these variables, press VIEW and then the corresponding softkey. ${ }^{207}$
- exit this menu, press EXIT.
- continue program execution, press R/S.

If you continue by pressing a softkey, the name of the corresponding variable is stored into $\mathbf{K}$. Your program can use this information to determine which key was pressed. Else, if you continue by pressing ( $/ \mathbf{S}$, $\boldsymbol{k}$ is not altered.

[^107]- Directly react on particular keys pressed. The key codes returned by KEY? (cf. p. 227) allow for 'real time' response to user input from the keyboard. KEY? takes a register argument ( $\mathbf{X}$ is allowed but will not lift the stack) and stores the key most recently pressed during program execution or pause in the register specified. ${ }^{208}$ Although the keyboard is active during program execution it is desirable to display a message and suspend the routine by PAUSE while waiting for user input. Since PAUSE will be terminated early by any key press, simply use PAUSE 99 in a loop to wait 10 s for input. And since KEY? acts as a test as well, a typical user input loop may well look like the following example:


## LBL 'User.in'

RCL $x y z$
PAUSE 99
KEY? 00
GTO 'User.in'
LBL? $\rightarrow 00$
XEQ $\rightarrow 00$
GTO 'User.in'

Displays the register with the message string. Waits 9.9 s for user input unless a key is pressed. Tests for user input and puts the key code in R00; if there was no input then return to the beginning; else if a label corresponding to the key code exists ... then call it, .. else return to the beginning.

Instead of the dumb waiting loop, the routine can do some computations in between and update the display before the next call to PAUSE.

To be even more versatile, you can use KTYP? to return the type of the key pressed if its row / column code is given (see the IOP).

If you decide not to handle the key in your program you may feed it back to the main processing loop of your WP43 with the command PUTK nn. It will cause the program to halt, and the key will be handled as if pressed after the stop. This is especially useful if you want to allow numeric input while waiting for some special keys like the arrows. After execution of the PUTK command you are responsible for letting the routine continue its work by pressing R/S.

See the IOI for more about the commands mentioned in this chapter.

[^108]
## Solving Differential Equations

The following examples use the programmability of your WP43 for solving differential equations frequently appearing in various disciplines:

A simple ecological model ${ }^{209}$ of interacting populations consists of rabbits with an infinite food supply and foxes that prey on them. The system can be approximated by a pair of nonlinear, first-order differential equations:

$$
\frac{d r}{d t}=2 r-\alpha r f \quad \text { and } \quad \frac{d f}{d t}=-f+\alpha r f
$$

where $r$ is the number of rabbits, $f$ is the number of foxes, and $\alpha$ is a positive constant to show how frequently rabbits and foxes meet. When $\alpha=0$ there are no encounters; the rabbits keep breeding and the foxes starve. For a specific $\alpha$, the probability of encounter is proportional to the product of the numbers of foxes and rabbits. A reasonable choice for $\alpha$ is 0.01 .
One way to solve this problem numerically is a simple Euler method, solving equations of the form:

$$
x_{n+1}=x_{n}+\Delta t g\left(x_{n}\right)
$$

using a $\ldots$ small increment of time $\Delta t$. The equations become:

$$
r_{n+1}=r_{n}+\Delta t\left(2 r_{n}-\alpha r_{n} f_{n}\right) \text { and } f_{n+1}=f_{n}+\Delta t\left(-f_{n}+\alpha r_{n} f_{n}\right)
$$

The features of the WP43 make it ideally suited for problems of this type.

```
LBL 'Foxes'
    STO 'n.fox' initialisation
    R
    STO 'n.rabb'
    FIX 5
LBL 01
    RCL 'n.fox'
    ENTER^
    ENTER^
    RCL 'n.rabb'
    RCL 'meet'
    x
    x number of meetings between rabbits and foxes
    STO 'n.meet'
    store it for later use in the other equation
```

[^109]```
x
RCL '\Deltat'
*
+
x<0.?
0
STO 'n.fox'
RCL 'n.rabb'
2
*
RCL 'n.meet'
RCL '\Deltat'
*
RCL 'n.rabb'
+
x<0.?
0
STO 'n.rabb'
IP
RCL 'n.fox'
IP
E 5
/
+
PAUSE 20
GTO 01
change in number of foxes
new number of rabbits
```

merge numbers of rabbits and foxes into 1 output number pause for 2 s for data recording end of time loop, return to its begin

## Example:

$r_{0}=300, f_{0}=150, \alpha=0.01, \Delta t=0.02$. Start with

### 0.01 STO meet 0.02 (STO) $\Delta t 300$ Entert 150 XEQ Foxes

The results can be plotted by hand on a graph of foxes versus rabbits. Note that the calculations give ... floating-point numbers, but the display is always truncated to an integer.

A plot of foxes versus rabbits (drawn overleaf) is a circular loop with a period of about five time units ( 250 iterations). The rabbit minimum is 14 , the maximum is 342 on the first loop. The fox minimum is 54 and the maximum is 478 on the first loop. $r_{0}=100, f_{0}=200$ is a constant solution.


The following method solves ordinary $2^{\text {nd }}$ order differential equations, a type frequently occurring in physics. ${ }^{210}$ In a first example, we will solve the equation of motion for the fall of a parachutist (or skydiver):

$$
\frac{d^{2} f}{d t^{2}}=g-b\left(\frac{d f}{d t}\right)^{2}
$$

with Earth's gravity acceleration and taking care of drag. Proceeding in small constant time steps $\Delta t$ again, the following set of equations controls the vertical motion of the skydiver:

$$
\begin{array}{r}
\left(\frac{d f}{d t}\right)_{1 / 2} \approx\left(\frac{d f}{d t}\right)_{0}+\left[g-b\left(\frac{d f}{d t}\right)_{0}^{2}\right] \frac{\Delta t}{2}=\left(\frac{d f}{d t}\right)_{0}+A_{0} \frac{\Delta t}{2} \\
f_{1} \approx f_{0}+\left(\frac{d f}{d t}\right)_{1 / 2} \Delta t \text { and }\left(\frac{d f}{d t}\right)_{3 / 2} \approx\left(\frac{d f}{d t}\right)_{1 / 2}+A_{1 / 2} \Delta t \\
f_{2} \approx f_{1}+(d f / d t)_{3 / 2} \Delta t \text { etc. }{ }^{211}
\end{array}
$$

Assume start height at time zero $(t=0)$ is 1000 m and vertical velocity is zero (i.e. $f(t=0)=f\left(t_{0}\right)=f_{0}=1000$ and $\left(\frac{d f}{d t}\right)_{0}=0$ ). Using named variables $\boldsymbol{\Delta t}, \boldsymbol{b}, \boldsymbol{t}, \boldsymbol{f}$, and $\boldsymbol{d f b y d t}$ (note a / must not appear in a name), the following routine will compute height and velocity of the parachutist as functions of time:

[^110]${ }^{211}$ Note that $\boldsymbol{A}$ must be $\geq 0$ always. Thus, this routine will work for velocities $<\sqrt{g / b}$ only. Furthermore, it will not work for abruptly decelerating fast initial movements (e.g. by opening a parachute).

| LBL 'PFall' |  |
| :---: | :---: |
| . 5 | initialize all variables used |
| STO ' $\Delta t$ ' |  |
| . 003 | assumed realistic drag value for a body (with closed |
| ST0 'b' | parachute) falling in air |
| 1000 | start height |
| STO 'f' |  |
| 0 | start time and velocity |
| STO 't' |  |
| STO 'dfbydt' | end of initialization |
| LBL 01 | begin of time loop |
| \# $\mathrm{g}_{\oplus}$ | recall $g_{\oplus}$ from CONST |
| RCL 'b' |  |
| RCL 'dfbydt' $x^{2}$ |  |
| $\times$ | $b \times(d f / d t)^{2}$ |
| - | $\boldsymbol{g}-\boldsymbol{b} \times(\boldsymbol{d} / \mathrm{d} \boldsymbol{t})^{2}=\mathbf{A}$ |
| RCL 't' |  |
| $x>0 . ?$ | check time - it will be zero in $1^{\text {st }}$ run |
| GTO 02 | from $2^{\text {nd }}$ run on go to local label 02 |
| DROP $\downarrow$ | $1{ }^{\text {st }}$ run only: forget $t$ |
| 2 | $1{ }^{\text {st }}$ run only: |
| 1 | $1^{\text {st }}$ run only: $\boldsymbol{A} / 2$ |
| GTO 03 | $1^{\text {st }}$ run only: go to common part |
| LBL 02 | from $2^{\text {nd }}$ run on: |
| DROP $\downarrow$ | from $2^{\text {nd }}$ run on: forget $t$ |
| LBL 03 | common part of time loop resumes here again |
| RCLx ' $\Delta t^{\prime}$ | $A \times \Delta t$ ( $/ 2$ in $1^{\text {st }}$ run) |
| STO+ 'dfbydt' | calculate the new df/dt |
| RCL ' $\Delta$ t' |  |
| STO+ 't' | calculate the new time |
| RCLx 'dfbydt' | $d f / d t \times \Delta t$ |
| STO- 'f' | calculate the new $\boldsymbol{f}$ |
| VIEW 't' | display new time for plotting next data points |
| STOP | -- |
| VIEW 'f' | display new height |
| STOP | ---- |
| VIEW 'dfbydt' | display new velocity |
| STOP | ---------- |
| GTO 01 | end of time loop |
| END |  |

Now, leave PEM and start program execution via XEQ PROG (PFall). Plotting the points calculated will result in a diagram like this. Height decreases follow-
 ing a parabola over time at beginning but becomes linear later. Note vertical velocity does not increase much after some 12 s here, approaching some $57 \mathrm{~m} / \mathrm{s}$ while skydiving with closed parachute. For comparison: the velocity limit with an open parachute ( $b \approx$ 0.3 ) will be less than $6 \mathrm{~m} / \mathrm{s}$, so the vertical velocity at touchdown will be like falling from a wall 1.65 m high.

In a $2^{\text {nd }}$ example, we will look at a horizontal harmonic oscillator. As long as it is oscillating freely, its equation of motion is simply

$$
m \frac{d^{2} f}{d t^{2}}=F=-r f
$$

with its spring rate $r$ and its mass $\boldsymbol{m}$. Adding an external stimulating force and velocity-dependent damping, the equation will change to

$$
m \frac{d^{2} f}{d t^{2}}=-r f-b \frac{d f}{d t}+s \sin \omega t \Leftrightarrow \frac{d^{2} f}{d t^{2}}=-\alpha f-\beta \frac{d f}{d t}+\gamma \sin \omega t=A
$$

with $\alpha=r / m, \beta=b / m$, and $\gamma=s / m$. Then the following set of equations controls the oscillator motion for small constant time steps $\Delta t$ :

$$
\begin{aligned}
\left(\frac{d f}{d t}\right)_{1 / 2} \approx\left(\frac{d f}{d t}\right)_{0}+\left[-\alpha f_{0}-\beta\left(\frac{d f}{d t}\right)_{0}+\gamma \sin \omega t_{0}\right] \frac{\Delta t}{2} & =\left(\frac{d f}{d t}\right)_{0}+A_{0} \frac{\Delta t}{2} \\
f_{1} \approx f_{0}+\left(\frac{d f}{d t}\right)_{1 / 2} \Delta t \text { and }\left(\frac{d f}{d t}\right)_{3 / 2} & \approx\left(\frac{d f}{d t}\right)_{1 / 2}+A_{1 / 2} \Delta t \\
f_{2} & \approx f_{1}+\left(\frac{d f}{d t}\right)_{3 / 2} \Delta t
\end{aligned}
$$

Now, all you have to do is rewriting the $1^{\text {st }}$ part of previous program:

| LBL 'Osci' |  |
| :---: | :---: |
| . 2 | initialize all variables used |
| STO ' $\Delta$ t' |  |
| 1 | assumed relative spring rate |
| STO ' $\alpha$ ' |  |
| . 5 | assumed relative damping |
| STO ' $\beta$ ' |  |
| 1 | size factor for stimulation |
| STO ' $\gamma$ ' |  |
| RAD |  |
| . 3 | frequency of stimulation |
| STO ' $\omega$ ' |  |
| 1.5 | start position |
| ST0 'f' |  |
| 0 | start time and velocity |
| STO 't' |  |
| STO 'dfbydt' | end of initialization |
| LBL 01 begin of time loop |  |
| RCL ' t ' |  |
| RCL ' $\omega$ ' |  |
| $\times$ |  |
| sin | $\sin (\boldsymbol{\omega} \boldsymbol{t})$ |
| RCL ' $\gamma$ ' |  |
| x | $\boldsymbol{V} \sin (\boldsymbol{\omega} \boldsymbol{t})$ |
| RCL ' $\beta$ ' |  |
| RCL 'dfbydt' |  |
| $\times$ |  |
| - | $\boldsymbol{V} \sin (\omega t)-\boldsymbol{\beta} d \boldsymbol{f} / \mathbf{d} \boldsymbol{t}$ |
| RCL ' $\alpha$ ' ${ }^{\text {' }}$ |  |
| RCL ' f ' |  |
| $\times$ |  |
| - | $\boldsymbol{V} \sin (\boldsymbol{\omega} t)-\boldsymbol{\beta} d \boldsymbol{f} / \boldsymbol{d t}-\boldsymbol{\alpha} \boldsymbol{f}=\boldsymbol{A}$ |
| RCL 't' |  |
| $x>0 . ?$ | check time - it will be zero in $1^{\text {st }}$ run |

The remaining code can be taken over from the $1^{\text {st }}$ example as is (feel free to delete the output of $d f / d t$ if you are not interested in it). Play with the parameters and observe the results.
Very similar equations apply to vertical oscillators, actually to almost all vibrating, swinging, or bouncing objects and parts. Enjoy!

Consult the ReM, App. I, for more about solving differential equations, also in $2 D$.

## The Programmable Menu (MENU)

Your WP43 features a programmable menu you can use to cause program branching. By this, you can create menu-driven programs. The command MENU selects this programmable menu. The menu will be displayed when the program stops. You can define each item in this menu so that when this key is pressed, a particular GTO or XEQ instruction will be executed. You can even re-define $\boldsymbol{\Delta}, \boldsymbol{\nabla}$, and EXITT. ${ }^{212}$

## To define a softkey in the programmable menu:

1. Store a text string of up to 7 characters in register $\mathbf{X}$. This text will appear in the menu space for the softkey specified (if free space does not suffice, only the first characters of $\boldsymbol{x}$ will be displayed). $\mathbf{X}$ is neither used nor dropped when defining $\boldsymbol{\Delta}, \boldsymbol{\nabla}$, or EXIT.
2. Call KEYG (i.e. on key, go to) or KEYX (i.e. on key, execute). You find them in P.FN.
3. The menu view changes. The black fields indicate you shall specify which softkey you want to define:

 or EXIT.

Alternatively, enter the respective key number, 1 through 21 (the unshifted leftmost softkey carries \#1, the -shifted leftmost \#13, $\Delta$ \#19, etc.).

Let's assume KEYG here and the -shifted leftmost softkey being pressed, for example.

[^111]4. The command echo changes:
5. Specify a program label using one out of the following four methods (the menu view changes to the one shown on p .220 ):
a. Key in a single-letter local label out of $\mathbf{A}-\mathbf{E}$.
b. Key in a two-digit local label,
c. Select a global label by pressing the corresponding softkey in PROG.
d. Key in a global label character by character using $\boldsymbol{\alpha}$ and AIM.

Unless you have just defined $\boldsymbol{\Delta}, \boldsymbol{\nabla}$, or EXIT, $\boldsymbol{x}$ will be dropped. Repeat this procedure for each softkey in the programmable menu you want to define. A new definition replaces any previous definition that may exist for this softkey.

## To display the programmable menu:

Execute the MENU function, e.g. by entering P.FN P.FN2 MENU .

## To clear all softkey definitions in the programmable menu:

Press CLR CLMENU (clear the programmable menu).

## Basic Kinds of Program Steps

You have seen various program steps so far. Each step takes a single place in program memory, and each step is numbered automatically. Basically, the contents of these steps fall into five categories - one program step may contain...

- a global or local label (like LBL 'Satell' or LBL 02 above) or
- an entire command (like - or $y^{\mathbf{x}}$ or STOx $\rightarrow$ 'Prd2') or
- a fixed text string or alphanumeric constant (like This is the iteration loop output: ; the text will be displayed enclosed in '; it
will be automatically stored in $\mathbf{X}$ without these delimiters) ${ }^{213}$ or
- a fixed number (a.k.a. a numeric constant, either keyed in like $-1.902 \times 10^{-16}, 1234516,1.7+i \times 0.4,-23^{4} / 5$, or recalled from CONST like \# $\lambda_{c}$; the respective value will be automatically stored in $\mathbf{X}$ ) or
- a fixed date (like 2022-09-10) or time (like 21:45:17.654); also here, the respective value will be automatically stored in $\mathbf{X}$.

Since each constant takes one step in a routine, there is no necessity for separating them by ENTER $\uparrow$.

## Example: <br> Think of calculating $12.3+45.67$ in a routine.

## Then pressing 12.3 ENTERT $45.67 \oplus$ will result in a program snippet

## 12.3

45.678
+
which will do for returning 57.978. The missing ENTER $\uparrow$ saves one byte of program space and makes the routine a tiny bit faster. It may not be really crucial here but you should know.

Constant vectors and matrices cannot be entered directly in a program; instead, you can store them in registers or variables and manipulate these stored items (as described in OMS 2) in routines as well.

Each program step requires at least one byte of memory (see the ReM, $A p p . B$ ). Use MEM? to learn about free space remaining in the pool. We think you will hardly ever run out of program space. ${ }^{214}$

You can use almost all operations provided on your WP 34 S to solve repetitive and iterative problems of almost any kind you can imagine with a speed exceeding the possibilities of earlier HP pocket calculators by far.

[^112]Keystroke programs may be more difficult to read than routines written in a high level language since inline comments and indenting are not supported. Loops in particular need some habituation.

Let's look at an exemplary PASCAL structure and its WP43 equivalent doing the same:


Once you have learned the basics (like "do if true, skip if false", sometimes reverting the logic of a test, and checking the counter at the end of the loop), you will enjoy the freedom you have.

## Deleting Programs

To delete some steps of a program, do as explained on pp. 220f. Repeat as often as necessary.
To delete an entire program quickly, press CLR CLP , then PROG to choose the program, or press $\alpha$ to input the program name, or alternatively ENTERT to delete the current program. Note that CLP will remove the entire program from memory without further ado, not just the current routine. And CLP cannot be undone! The space freed by CLP will be returned to the pool of free space your WP43 offers you.
To delete all programs stored in RAM, press CLR CLPall, think, and confirm your intention. Thereafter, program memory will be completely wiped out, i.e. you may have up to some 45000 bytes for new programs then (depending on the space your data in registers and variables occupy). Note that also CLPALL, since confirmed, cannot be undone.

## Flash Memory (FM)

In addition to the RAM provided, your WP43 also allows you accessing its large FM. The major part of it is allocated to a so-called FAT disk of 6 MB (see a sketch in next chapter). You can use it for voltage-fail-safe storage of your programs and data, e.g. for backups.
Use SAVE for creating an on-board backup of your entire work (your calculator configuration, programs, variables, register contents, flag status, etc.). ${ }^{215}$ The different flavors of LOAD (provided in I/O) are for recalling it:

- LOADE recalls the statistics matrices STATS and histo as well as the summation registers,
- LOADR the contents of all numbered registers,
- LOADV all user-defined variables,
- LOADSS recovers the saved system settings of your WP43 (incl. its configuration),

[^113]- LOADP all programs, and
- LOAD recalls the entire backup at once (incl. the lettered registers).

SAVEST saves the system state of your WP43 in a named file on FAT disk. Actually, such a system state encompasses the same as a backup but you may store various system states under different filenames. ${ }^{216}$
LOADST loads the system state of your choice from the respective named file into your WP43.

Turn to the $I O /$ for more information about these nine commands.
While $F M$ is a safe place, hence ideal for long-living data, it shall not be used for frequently changing temporary storage like in programmed loops. ${ }^{217}$ Conversely, registers and standard user program memory residing in RAM are designed for data changing with high frequency - but they will not hold any data when the batteries are removed for longer than xxx seconds. So both RAM and FM have their specific advantages and disadvantages you should take into account for optimum benefit and longevity of your WP43.

## Input and Output of Data and Programs

Looking at I/O from your WP43 calculator, this matter appears as pictured schematically below (memories not to scale).

Print orders will be sent via the $I R$ line. For the different flavors of LOAD, SAVE, and SAVEST cf. previous chapter; DUMP, SNAP, READP, and WRITEP are covered below.

[^114]

## Data exchange with your computer:

There are two ways you can take care of data exchange:

1. With a USB cable connecting your WP43 with your computer, you can 'activate' the FAT disk (FD, a.k.a. USB disk) in FM via ActUSB. Then said disk will appear in the address space of your computer, and you can use file management applications there (e.g. the Explorer on a Windows $P C$ ) to copy or move arbitrary files to and from this FD. Having done this, EXIT ENTERT will return the FD to your WP43 exclusively, so you can continue calculating.
2. You can transmit programs from your WP43 to FD, store snapshots or backups, or dump specific results (e.g. matrices or very long integers) in files there. You may then employ method 1 and deal with these files on your computer, enjoying more editing comfort than a little calculator can offer. We will elaborate on this way in the following:

I/O DUMP dumps $\boldsymbol{x}$ with its full precision on FD in the directory SAVFILES in a file named regx-yyyymmdd-hhmmsscc.tsv (according to date and time of the dump; see two examples below). This is the way to permanently output all digits of very long integers ( $>10^{44}$ ). Note that you can SHOW long integers up to $10^{296}$ easily but only their top 44 digits will stay on display after next keystroke.

|  | regx-20230302-15101624 | 02.03.2023 15:10 | TSV-Datei | 1 KB |
| :---: | :---: | :---: | :---: | :---: |
|  | regx-20230302-15102094 | 02.03.2023 15:10 | TSV-Datei | 1 KB |

Use a text editor for dealing with .tsv-files.
SNAP stores a snapshot (a.k.a. screenshot, hardcopy) of the entire calculator screen on FD in the directory SCREENS in a graphic file named yyyymmdd-hhmmsscc.bmp (according to date and time of the snapshot). See the $I O I$ for more.

> I/O WRITEP writes your current program in a named .p43 file on FD in the directory PROGRAMS.

I/O READP reads a named .p43 file from FD into program memory, so you can execute or edit this program.

What works on the Simulator shall work on your physical WP43 and vice versa as well - both platforms share the same set of functions. ${ }^{218}$ So you can WRITE the current Program from your WP43 in a file on FD, connect your WP43 to your computer, READ the Program file from FD into program memory of the Simulator, execute, test, debug, and edit. When finished, WRITEP from your computer on FD again, disconnect, READP, and work with the refined file in program memory of your WP43. In detail:

## Printing:

If you have got a thermoprinter HP82440 A or B, you can use it to get immediate hardcopies of your results. There are fifteen print commands

[^115]provided，all stored in 量（a．k．a．PRINT，accessed via $\square+0$ ）．They do the following：

且MODE $n$ sets the print mode（see the $I O I$ ）．
且WIDTH returns the number of print columns that $\boldsymbol{x}$ would take in the print mode set．
具TAB $n$ positions the printhead to column $\boldsymbol{n}$ ．
国ADV prints the current contents of the print buffer and a linefeed．
皿DLAY $n$ sets a delay of $\boldsymbol{n}$ ticks to be used with each linefeed on the printer．
国\＃ $\boldsymbol{n}$ sends a single byte $\boldsymbol{n}$ to the printer，e．g．a control code．
国CHAR $n$ sends one character with code $\boldsymbol{n}$ to the printer．
且x prints $\boldsymbol{x}$ ．
国r $r$ prints the content of the register or variable specified．
国 $\Sigma \quad$ prints the contents of all summation registers．
国STK prints the contents of the entire stack as allocated．
且REGS prints the contents of a set of GP registers．
国USER prints all user－defined variable names and global program labels in alphabetic order．
䦔PROG lists the current program．
国LCD prints a hardcopy of the entire screen．
We recommend consulting the $I O I$ for details of these commands．

## Local Data

After some time with your WP43 you will have a number of routines stored， and keeping track of their resource requirements may become challenging．Most modern programming languages take care of this by declaring local variables，i．e．memory space allocated from general data memory and accessible for the current routine only；when the routine is finished，this memory is released．On your WP43，mainly registers are
used for data storage - so we offer local registers to you allocated to your routines exclusively.

> Example:
> Let's assume you create a routine labeled P1 and need five registers (in addition to the stack) for your computations therein. Then all you have to do is just enter PEM, go into the routine P1, and key in
P.FN LOCR 5 ENTERT
specifying that you want five local registers. After this step, you can access these new registers by using local addresses $.00-.04$ throughout P1.

Now, if you call another routine P2 from P1, also P2 may contain a step LOCR, requesting local registers once again. These will also carry local register addresses .00 etc., but the local register .00 of $\mathbf{P 2}$ will be physically different from the local register .00 of P1, so no interference will occur. As soon as RTN or END is executed, the local registers of the respective routine will be released and the memory they took will be returned to the pool of free space.

Local data holding allows for recursive programs, since every time such a routine is called again it will allocate a new set of local registers and user flags being different from the ones it got before. See the commands LOCR, LOCR?, and POPLR in the IOI.

You may allocate up to 98 local registers per routine (as far as memory allows). In addition, you get 32 local user flags as soon as you request at least one local register.

Since you are free to nest subroutines (cf. p. 223) as well, and each subroutine may allocate its own local registers, the total number of registers allocated may become quite large. The total number of accessible registers at one time, however, will never exceed 211 (cf. p. 58). Look up App. B of the ReM for more information, also about limitations applying to local data.

## SECTION 4: ADVANCED PROBLEM SOLVING

Your WP43 provides some powerful commands for computing programmable sums and products, definite integrals, $1^{\text {st }}$ and $2^{\text {nd }}$ derivatives as well as for solving equations. All these are contained in $\frac{\text { ADV EQN }}{1} \begin{array}{r}\text { W }\end{array}$ ADV or EQN. Pressing ADV in RUM results in this view:


The commands $\Sigma_{\mathrm{n}}, \Pi_{\mathrm{n}}$, SLVQ, SOLVE, $\int, \mathrm{f}^{\prime}(\mathrm{x})$, and $\mathrm{f}^{\prime \prime}(\mathrm{x})$ are explained below in this order; all of them are programmable. Interactive integrating, deriving, and solving equations can be reached through (EQN). See below for details and examples.

## Programmable Sums

The command $\Sigma_{n}$ is called with a loop control number in $\mathbf{X}$ and a label trailing the command. Said loop control number follows the format ccccc.fffii (as known from DSE etc. mentioned above).
In its heart, $\Sigma_{n}$ then works like this:

1. $\Sigma_{\mathrm{n}}$ sets the sum to 0 initially.
2. $\Sigma_{n}$ fills all stack registers with $\operatorname{ccccc}$ and calls the routine specified by label. That routine returns a summand in $\mathbf{X}$.
3. $\Sigma_{\mathrm{n}}$ adds this summand to said sum.
4. $\Sigma_{n}$ decrements $\operatorname{ccccc}$ by ii (or by 1 if ii $=0$ ); if $\operatorname{ccccc} \geq f f f$ then $\Sigma_{\mathrm{n}}$ goes back to step 2, else it returns the final sum in $\mathbf{X}$.

## Example:

$\sum_{k=0}^{100} \sqrt{k}=$ ?

## Solution:

1. Write a little program for the internal calculation of the summands:

LBL ' $\Sigma v^{\prime}$
$\sqrt{x}$
RTN
2. Enter

100
ADV $\boldsymbol{\Sigma}_{\mathbf{n}} \propto \square$ (S ENTERT
(or pick $\Sigma \mathrm{V}$ from PROG, cf. p. 220)
and get $\quad 671.4629$ returned if FIX 4 is set.
$\Sigma$ deliberately sums from the last term to the $1^{\text {st }}$, on the assumption that summations will often be of convergent series so this summing sequence should generally increase accuracy of result.

## Programmable Products

The command $\Pi_{\mathrm{n}}$ is called with a loop control number in $\mathbf{X}$ and a label trailing the command (as for $\Sigma_{\mathrm{n}}$ above).
In its heart, $\Pi_{n}$ works almost as $\Sigma_{n}$ :

1. $\Pi_{n}$ sets the product to 1 initially.
2. $\Pi_{n}$ fills all stack registers with $\operatorname{ccccc}$ and calls the routine specified by the label. That routine returns a factor in $\mathbf{X}$.
3. $\Pi_{n}$ multiplies this factor with said product.
4. $\Pi_{n}$ decrements $\operatorname{ccccc}$ by ii (or by 1 if $i i=0$ ); if $\operatorname{ccccc} \geq f f f$ then $\Pi_{\mathrm{n}}$ goes back to step 2, else it returns the final product in $\mathbf{X}$.

## Example:

$\prod_{k=1}^{30} \frac{1}{\sqrt{k}}=$ ?

## Solution:

1. Write a little program for the internal calculation of the factors:

LBL 'PROD'
$\sqrt{x}$
$1 / \mathrm{x}$
RTN
2. Instead of calling PROD via ADV $\boldsymbol{\Pi}_{\boldsymbol{n}}$ you can call it from another program and measure the time it takes. Thus write a second little program:

```
LBL 'P2'
    TICKS
    STO OO
    30.001
    # n 'PROD'
    TICKS
    RCL- 00
    RTN
```

XEQ PROG (P2) and get $6.1400 \times 10^{-17}$ and 2 ticks ( 0.2 s ) returned.

## Solving Quadratic Equations

The command SLVQ finds the real and complex roots of a quadratic equation $a x^{2}+b x+c=0$ with its real parameters on the input stack [c, b, a, ...] :

- If $r:=b^{2}-4 a c \geq 0$ then SLVQ will return the 2 real roots $(-b \pm \sqrt{r}) / 2 a$ in $\mathbf{Y}$ and $\mathbf{X}$. If called in a routine then the step after SLVQ will be executed.
- Else, SLVQ will return the $1^{\text {st }}$ complex root in $\mathbf{X}$ and the $2^{\text {nd }}$ in $\mathbf{Y}$ (the complex conjugate of the $\left.1^{\text {st }}\right)$. If called in a routine then the step after SLVQ will be skipped.

So, actually, SLVQ tests for real roots at its very end. In either case, it returns $\boldsymbol{r}$ in $\mathbf{Z}$. Higher
 stack registers are kept unchanged. L will contain equation parameter $\mathbf{c}$.

## Example:

Find the roots of $4 x^{2}-3 x-2=0$.
Solution (with FIX 4 chosen):
4 ENTERT 3 + $1 /$ ENTERT 2 + $/-$
(ADV SLVQ returns $x=1.1754, y=-0.4254, z=41.0000$. Since $z$ is positive, $\boldsymbol{x}$ and $\boldsymbol{y}$ are the two real roots of this equation here.

Check: Store $\boldsymbol{x}$ in $\mathbf{J}$ and $\boldsymbol{y}$ in $\mathbf{K}$. Then enter


Remember your WP43 calculates with 34 digits precision, so any result within $\pm 3 \cdot 10^{-33}$ is equal to zero in this matter.

## Arbitrary Algebraic Equations

The menu EQN lets you create, store, select, edit, and delete arbitrary algebraic equations. You may use each such equation for

- solving it interactively for any variable it comprises,
- plotting,
- integrating or differentiating.

The total number of equations and variables usable is limited only by the amount of free memory available.

## Example:

Press EQN. If there is no equation in memory yet, your WP43 will return:

## NEW

Press NEW to enter a new equation. You will get immediately:

with $A I M$ turned on (cf. pp. 203ff). Key in your equation, e.g.
height $=\mathbf{h} \downarrow(0) \mathbf{v} \mathbb{R} \downarrow$ time $-\mathbf{g}_{\oplus}(1) \times$ time ^ (2) ${ }^{219}$
for the height of e.g. a ball thrown vertically upwards with velocity $v_{0}$ starting at height $\boldsymbol{h}_{0}$ (e.g. from the Eiffel Tower or a Zeppelin). Press ENTERT for closing the Equation Editor and see: ${ }^{220}$


| height $=\mathrm{h}_{0}+\mathrm{v}_{0} \cdot$ time $-\mathrm{g}_{\oplus} / 2 \cdot$ time $^{2}$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| NEW | EDIT | $f^{\prime \prime}$ | $\mathrm{f}^{\prime}$ | ¢f | Solver |

You will get such a view whenever one or more equations are stored. The equation displayed (in the blue row) is called the current equation. ${ }^{221}$
Pressing EDIT opens the Equation Editor for this equation:


You may modify this equation at any position by moving the edit cursor to the location behind the symbol(s) to be changed and pressing $\boldsymbol{\uplus}$ as required followed by the new symbol(s) to be inserted here.

For labeling this equation, move the cursor left to its very begin using $\leftarrow$, and key in up to 7 characters (cf. p. 59) plus a colon:

## (F) R E E $\boldsymbol{\Delta}(\mathbb{F} \mid(\mathrm{A}):$

freeFal: :height $=h_{0}+v_{0} \cdot$ time $-g_{\oplus} / 2 \cdot$ time $^{\wedge} 2$

|  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $\leftarrow$ | () | $\wedge$ | $:$ | $=$ | $\rightarrow$ |

[^116](If an equation becomes wider than the screen, ellipses will be displayed at its end(s); then use $\leftarrow$ and $\rightarrow$ for scrolling.)

ENTERT

| freeFal: | $\text { height }=h_{0}+v_{0} \cdot \text { time }-g_{\oplus} / 2 \cdot \text { time }^{2}$ |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| NEW | EDIT | f' | f' | Jf |  |

ENTERT terminates the Equation Editor and stores the modifications you made. Note editing an equation clears all its variables.

Note you are free to create and name the variables in your equations to your liking; but remember variable names must be unique - do not reuse names of predefined items for your variables (cf. p. 59). ${ }^{222}$

Furthermore, you may employ any constant defined in your WP43 by just calling its name (see pp. 287ff). You must not, however, reuse any name of such a constant for any of your variables - else your WP43 may misunderstand your intentions.

Evaluation of equations follows PFEMDAS (i.e. parentheses first, then function, exponentiation, multiplication, division, addition, subtraction). Unary minus is interpreted as a multiplication times -1. And the number of pending operations as well as the number of parameters in each such algebraic equation must not exceed nine.

## The Solver

## Interactively Solving Arbitrary Algebraic Equations

The built-in Solver application of your WP43 is a special root finder that enables you to solve an equation for any of its variables. It allows for solving for an arbitrary unknown as well as for finding the root(s) of an arbitrary equation. ${ }^{223}$

[^117]

Press EQN, make the equation you want to solve the current equation (see previous chapter), and press Solver. Your WP43 will check this equation for syntax errors (missing operators, misspelled functions, illegal variable names, etc.). It will then return a menu of all applicable variables, like the one in our example:


Your WP43 recognized $\boldsymbol{g}_{\oplus}$ being defined in CONST (see pp. 287ff). It will detect other constants as well, if applicable. Now you can enter values for each known variable by pressing the respective softkey, e.g.
$\mathbf{0}$ height $50 h_{\mathbf{0}} \mathbf{1 5} \mathrm{v}_{\mathbf{0}} \quad$ corresponding to a target height 0 , a start point on a 50 m platform, and a velocity of $15 \mathrm{~m} / \mathrm{s}$ upwards at time zero. Only one unknown variable remains.
(Optionally, enter one or two initial guesses for the unknown like

## 5 time 10 time.)

Then press [54 for the unknown
time (now without any numeric input heading - the Solver takes what is in $\mathbf{X}$ and $\mathbf{Y}$ as initial guesses for time), and your WP43 will solve the equation for this variable and return its value in $\mathbf{X}$ (for FIX 1 set):
time $=5.1$
freeFal: height $=h_{0}+v_{0} \cdot$ time $-g_{\oplus} / 2 \cdot$ time $^{2}$
i.e. $5.1 s$ until the ball hits the ground. ${ }^{224}$

If you want to recall your inputs or results later again, use (RCL for the respective variables.

[^118]Another example (from the HP-27S OM):
Carbon-14 Dating. Wood on the outer surface of a giant sequoia tree exchanges carbon with its environment. The radioactivity of this wood is 15.3 counts per minute per gram of carbon. A sample of wood from the center of the tree yields 10.9 counts per minute per gram of carbon. The rate constant for the radioactive form of carbon, ${ }^{14} \mathrm{C}$, is $1.20 \times 10^{-4}$. How old is the tree? What is the half-life of ${ }^{14} \mathrm{C}$ ?

Solution (assuming you continue directly after previous example):
Exit the old current equation and enter a new one for radioactive decay:


Functions like $\boldsymbol{\operatorname { s i n }}, \mathbf{a r c c o s}, \mathbf{l g}, \mathbf{l b}, \boldsymbol{t a n h}$, or sinc shall be called by their names as listed in the 101 . Hit $\square \sqrt{\boldsymbol{X}}$ to enter $\sqrt{ }\left(\mathbf{l}_{m}\right)$, $|\overline{x \mid}|$ to insert $|=|$ and $\boldsymbol{e}^{\boldsymbol{x}}$ to get $\mathbf{e}_{\mathbf{E}}{ }^{\wedge}\left({ }^{m}\right)$; type the other function names, trailed by ().
Specify the arguments within the parentheses (or bars) like above. This applies to dyadic operations in analogy - their arguments shall be separated by colons (like COMB(x:y)).

ENTERT


Solver

$1.2 \mathrm{E}++/ 4$ rate 15.3 no 10.9 n time

This is the computed age of this tree in years.
Now, calculate the half-life of ${ }^{14} \mathrm{C}$; this is the time needed for half the material present to decay:

## 2 no 1 n time

```
time = 5776.2
```

Good guess! The half-life of ${ }^{14} \mathrm{C}$ is known to be $(5730 \pm 40)$ years. Since the rate constant was given with $1 \%$ accuracy in the problem above, your calculatory result for the half-life cannot be any better, i.e. ( $5776 \pm 58$ ) years; this is consistent with the 'official' value.

## One more example:

Find the roots of $7 x^{3}+5 x^{2}-3 x-2=0$.

## Solution:

1. Enter the equation as demonstrated above.
2. Make this equation the current equation and press Solver

You will see:

| 5776.2 |  |  |  |
| :--- | ---: | :---: | :---: |
| poly: $0=7 \cdot x^{3}+5 \cdot x^{2}-3 \cdot x-2$ |  |  |  |
| $x$ | Draw Calc |  |  |

3. Optionally enter one or two initial guesses for the unknown like
$0 \times 1 \times$.
Press Draw for drawing the equation between these two values. Counting the (minor) ticks you can estimate this function having a root at about 0.64.

Let's see - press EXIT and you will get the previous screen again.

4. Set the display format to FIX 4, then leave DISP via EXIT. Press (F1 for the unknown $\mathbf{x}$ twice (now without any numeric input heading), and your WP43 will solve the equation for this variable and return its value in $\mathbf{X}$ :

5. If you want to crosscheck you can press

for the function at this location, confirming the result of the Solver.
6. There may be up to two more roots - you guess they are on the left side:
a. Enter two new initial guesses for the unknown like $\mathbf{- 1} \mathbf{x} \mathbf{. 7} \mathbf{x}$; check the function with Draw; 2022-01-04 23:39 use Z00M to get a convenient tick size. You can read from the plot there will be one more root close to -0.55 and another one at -0.8 .
b. Press EXIT to close the diagram.

c. Overwrite the guesses with $\mathbf{- 1} \times \mathbf{- . 6 \times} \times$, then press $\mathbf{x}$ once more, and you will get:
$\square x=-0.8064 \square$
d. Enter two new initial guesses for the unknown like $-\mathbf{7} \mathbf{x} \mathbf{- . 4} \mathbf{x}$. Press $\mathbf{x}$ once more and you will get:
$x=-0.5510$

Feel free to look into Section 5 of the HP-27S OM for more about interactive solving of algebraic equations.

## Interactively Solving Expressions Stored in Programs

Instead of operating on an equation as described in previous chapters, your WP43 can also solve an expression $\boldsymbol{f}$ stored in a program. Then,

1. Write a program for $\boldsymbol{f}$.
2. Press ADV SOLVE and select the program you want solved.
3. Enter values for all known variables of $\boldsymbol{f}$.
4. Let your WP43 compute the unknown variable.
5. Leave the Solver.

We will go through this step by step:

1. Write a program for $\boldsymbol{f}$ :

- It shall begin with a global label.
- It must define all variables required for calculating $\boldsymbol{f}$.
- It shall be as efficient as possible since it is going to be executed many times.

For interactive solving, we recommend proceeding as follows for this program: From its $2^{\text {nd }}$ step on, menu variables shall be declared using MVAR instructions (cf. p. 234) covering all variables of $\boldsymbol{f}$; no trailing VARMNU here! The subsequent body of the routine shall then evaluate $\boldsymbol{f}$ recalling these variables. For a Solver routine, the original expression shall be rewritten in a way that $\boldsymbol{f}=0$ is fulfilled.

## Example (with FIX 01 chosen):

## Let's return to the equation we dealt with in the last two chapters:

$$
\text { height }=h_{0}+v_{0} \cdot \text { time }-g_{\oplus} / 2 \cdot \text { time }^{2}
$$

This is easily rewritten:

$$
h_{0}+v_{0} \cdot \text { time }-g_{\oplus} / 2 \cdot \text { time }^{2}-\text { height }=0
$$

So the required program might look like this:

```
LBL 'FreeF'
    MVAR 'height'
    MVAR ' }\mp@subsup{h}{0}{\prime
    MVAR ' }v
```

    MVAR 'time' (no trailing VARMNU here!)
    ```
# g
-2
/
RCLx 'time'
RCL+ 'v}\mp@subsup{v}{0}{\prime
RCLx 'time'
RCL+ 'h}\mp@subsup{h}{0}{\prime
RCL- 'height'
RTN
take this out of CONST.
now we have got f}\mathrm{ .
```

2. Press ADV:

| PGMSLV |  | $\mathrm{f}^{\prime \prime}(\mathrm{x})$ | PGMINT |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| SOLVE | SLVQ | $\mathrm{f}^{\prime}(\mathrm{x})$ | $\mathrm{n}_{\mathrm{n}}$ | $\Sigma_{n}$ | ¢fdx |

Choose SOLVE :


Press PROG and pick the proper program for $\boldsymbol{f}$ (here FreeF). You will get the corresponding menu of variables, i.e. here:
height $\quad h_{0} \quad v_{0} \quad$ time
3. Enter values for all known variables of $\boldsymbol{f}$ and (optionally) one or two guesses for the unknown.

In our example, we may just take the values we know from above:
0 height $50 h_{0} \quad 15 \mathrm{v}_{0} \quad 5$ time 10 time
4. Let your WP43 compute the unknown variable.

Press time once more but without a heading numeric entry. Your WP43 will return time $=5.1$ as you have expected (cf. p. 258).
5. Leave the Solver.

Pressing EXIT will return to the top view of ADV as known from step 2.

## Using the Solver in a Program

For using the Solver in a programs, it has to be told what you did tell it interactively so far. Calling ADV in PEM, you will need a command you did not use above so far:


PGMSLV is for specifying the program calculating $\boldsymbol{f}$. This step must be found in your program before SOLVE is called.

Furthermore, define the necessary variables in advance and load the known values, e.g. using STO. Eventually, the unknown variable must be specified calling SOLVE.

## Example:

Let's return to the equation we dealt with in the last chapters. So the required program for $f$ might look like this (like previous program but without the MVAR steps if the variables required were defined before):


The program one level above could contain a section looking like this:

```
...
```

PGMSLV 'FreeFp' SOLVE 'time' VIEW 'time'
specify the function to be solved.
solve for time.
display the solution.

Before starting this program (let's call it L), fill the variables of the equation to be solved, e.g. with the start values known from above:
0 STO VAR height 50 STO VAR $h_{0}$ 15 STO VAR $\mathrm{v}_{0}$

Optionally fill the unknown with a $1^{\text {st }}$ guess, e.g. with 5 as we specified above (a $2^{\text {nd }}$ guess will be taken from $\mathbf{X}$ ):

## 5 STO VAR time

Alternatively, you can put these STO steps in $\mathbf{L}$ at any place before SOLVE is called. Or put in MVAR steps for all required variables as we did for FreeF above. Or mix both methods.

Call $L$ via XEQ (L) and you will get time $=5.1$ (cf. p. 258).

Eventually turn to Part 3, Section 12 of the HP-42S OM. See the HP-34C OHPG (Section 8 and App. A) or the HP-15C OH (Section 13 and App. D) for more information about automatic root finding and some caveats.

## Integration

## Numeric Integration of Arbitrary Algebraic Equations



Solution:

You can compute definite integrals numerically using the command $\int$ on your WP43.

## Example:

Let's compute the Bessel function of ${ }^{\text {st }}$ kind and order zero. This function can be written

$$
J_{0}(x)=\frac{1}{\pi} \int_{0}^{\pi} \cos [x \sin (t)] d t
$$

This is calculated in radians, thus enter MODE RAD and press EQN :

| 5.1 |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| freeFal: height $=\mathrm{v}_{0} \cdot$ time $-\mathrm{g}_{\oplus} / 2 \cdot$ time $^{2}$ |  |  |  |  |  |
| DELETE |  |  |  |  |  |
| NEW | EDIT | $f^{\prime \prime}$ | $\mathrm{f}^{\prime}$ | \f | Solver |

Above function is not in the equation list yet. So, press NEW and start entering the integrand:

## (ITB $\boldsymbol{T}$ (E)STS:



Continue with COS ( ) $x \times$ STN ( ) T


Close and store this function by pressing ENTERT. The menu will return to the previous one:


Then press 【f . Your WP43 will check the current equation (cf. pp. 255ff) for syntax errors (missing operators, misspelled functions, illegal variable names, etc.). ${ }^{225}$ It will then return a menu of all applicable variables:


You can enter a value for each variable you already know (i.e. integration constants) by pressing the respective softkey now, e.g. $\mathbf{2 \times}$. (For recalling such an integration constant, just press RCL VAR before the respective softkey.)

Then select the variable of integration by simply pressing $\mathbf{t}$ here (there must not be any numeric input heading $\mathbf{t}$ !). The menu will change:


Even your WP43 cannot compute an integral exactly, it approximates its value numerically. The accuracy of this approximation depends

[^119]on the accuracy of the integrand's function itself as calculated by your program. This is affected by round-off error in the calculator and also by the accuracies of the integration constants specified.
ACC is a real number that defines the relative error of the integration. With ACC = 0.001, for example, you can be sure that
$$
\left|\frac{v_{T}-v_{C}}{v_{C}}\right| \leq 0.001
$$
(with $\boldsymbol{v}_{\boldsymbol{T}}$ being the true value and $\boldsymbol{v}_{\boldsymbol{C}}$ the computed value of the integrand) at any point between $\downarrow$ Lim and $\uparrow$ Lim.

We want to see the result accurate to three decimals. Thus we enter .001 ACC for the accuracy of computation,
0 *Lim for the lower integration limit,
(IT) $\uparrow$ Lim for the upper integration limit,
and start integrating by pressing $\int$. Your WP43 will return:


Do not forget to divide this result by $\pi$ to get the correct value for :
DISP FIX (3) (1) ${ }^{226}$


Enter other values for $\boldsymbol{x}$ and integrate again to get $J_{0}(x)$ at other locations. ${ }^{227}$

[^120]
## Interactively Integrating Expressions Stored in Programs

Instead of operating on an 'equation' as described in previous chapter, your WP43 can also integrate an expression $\boldsymbol{f}$ stored in a program. Then,

1. Write a program for $\boldsymbol{f}$.
2. Press ADV 【fdx.
3. Enter values for all known variables (integration constants) of $\boldsymbol{f}$, for ACC, and for the integration limits. Select the variable of integration.
4. Let your WP43 compute the definite integral specified. ${ }^{228}$

We will go through this step by step:

1. Write a program for $\boldsymbol{f}$ :

- It shall begin with a global label.
- It shall define all variables required for calculating $f$.
- It shall be as efficient as possible since it is going to be executed many times.

We recommend proceeding as follows: From the $2^{\text {nd }}$ step of this program on, menu variables shall be declared using MVAR instructions (cf. p. 234) covering all variables of $\boldsymbol{f}$. The subsequent body of the routine shall then evaluate $f$ recalling these variables.

## Example:

Let's return to the integrand we dealt with in the last chapter. Then the required program for $\boldsymbol{f}$ might look like this:

LBL 'IBessI'
MVAR ' $x$ '
MVAR ' $t$ '
RCL ' t '
sin
RCLx ' $x$ '
cos now we have got $f$.
RTN

[^121]2. Press ADV:


## Choose $\int \mathrm{fdx}$ :



Press PROG and pick the proper program for $\boldsymbol{f}$ (here IBessI). You will get the corresponding menu of variables, here:

3. Enter values for all known variables (integration constants) of $\boldsymbol{f}$ and select the variable of integration.

In our example, we may take the value we know from above: $\mathbf{2 x} \mathbf{t}$. So $t$ will be the variable of integration. The menu will change now:

| ACC | *Lim | 个Lim | $\int$ |
| :--- | :--- | :--- | :--- |

We enter (like in previous chapter) . 001 ACC 0 $\downarrow$ Lim $\pi$ tim
4. Let your WP43 compute the definite integral specified.

Press $\int$ to integrate with all the parameters as chosen, and your WP43 will return $\int \approx 0.704$ as you might have expected (cf. previous chapter).
Divide by $\pi$ to get the value for $J_{0}(x)$ as above.

## Using the Integrator in a Program

For using the Integrator in a programs, it has to be told what you sent to it interactively so far. Calling ADV in PEM, you will need a command you did not use above so far:


PGMINT is for specifying the program calculating $\boldsymbol{f}$. This step must be found in your program before the integration is called. You shall define the necessary variables in advance. Use STO to load them with the known values before integrating. Then call the menu $\int \mathrm{fdx}$ :

| STO AC | STO $\downarrow \mathrm{L}$ | STO AL |
| :--- | :--- | :--- |

The integration limits as well as the requested accuracy shall be stored before integrating. Eventually, the variable of integration must be specified calling $\int$.

## Example:

Let's return to the integrand we dealt with in the last two chapters. So the required program for $\boldsymbol{f}$ might look like this:

```
LBL 'IBessP'
    RCL 't'
    sin
    RCLx 'x'
    cos now we have got f
    RTN
```

The program one level above could contain a section looking like this:
LBL 'IntP'

```
PGMINT 'IBessP' specifying the function to be integrated.
    0
    STO '$Lim'
    \pi
    STO 'fLim'
    0 . 0 0 1
    STO 'ACC'
    \intfd 't' integrate over time.
    VIEW X
    display the solution.
```

Before starting this program, load the variables staying constant under integration, e.g. with the start values known from above:

## 2 STO VAR $x$

XEQ PROG (IntP) and you will see $\int \approx 0.704$ as you may have expected (cf. previous chapter). Divide by $\pi$ to get $J_{0}(x)$ as above.

Eventually turn to Part 3, Section 13 of the HP-42S OM. See the HP-34C OHPG (Section 9 and App. B) or the HP-15C OH (Section 14 and App. E) for more information about automatic integration and some caveats.

## Differentiating

## Differentiating Arbitrary Algebraic Equations

There are two commands provided returning the values of the first two derivatives of the function $\boldsymbol{f}(\boldsymbol{x})$ at position $\boldsymbol{x}$. This function $\boldsymbol{f}(\boldsymbol{x})$ can be specified in an equation.
$f^{\prime}(x)$ returns the $1^{\text {st }}$ derivative. For computing it, ...

1. $\mathrm{f}^{\prime}(\mathrm{x})$ will first look for a user routine labeled ' $\mathbf{\delta x}$ ' (or ' $\boldsymbol{\delta} \mathbf{X}$ ', ' $\Delta \mathbf{x}$ ', or ' $\Delta \mathbf{X}$ ', in this order), returning a fixed step size $\boldsymbol{d x}$ in $\mathbf{X}$. If that routine is not defined, $d x=0.1$ is set for default.
2. Then, $\mathrm{f}^{\prime}(\mathrm{x})$ fills all stack registers with $\boldsymbol{x}$ and calls $\boldsymbol{f}(\boldsymbol{x})$. It will evaluate $\boldsymbol{f}(\boldsymbol{x})$ at ten points equally spaced in the interval $\boldsymbol{x} \pm 5 d \boldsymbol{x}$ (if you expect any irregularities within this interval, change $d x$ to exclude them).
3. On return, the $1^{\text {st }}$ derivative will be in stack register $\mathbf{X}$, while $\mathbf{Y}, \mathbf{Z}$, and $\mathbf{T}$ will be clear and the position $\boldsymbol{x}$ will be in $\mathbf{L}$.

## Example (with SCI 3 set):

Take the equation poly: again (used on pp. 260f for solving). Instead of checking two function values left and right of the root you could check the slope at the root just once.

## Solution:

You have got poly in EQN already. For each of the three roots found, calculate the root first, then the $1^{\text {st }}$ derivative of $\boldsymbol{g}$ at that point:

1. Press [EQN, make poly the current equation, and press Solver. You will see then:

2. Find the $1^{\text {st }}$ (leftmost) root as shown above:

$$
-2 \times-1 \times \times
$$



Press EXIT :


Press $\mathbf{f}^{\prime}$ :

$$
\begin{array}{ll}
f^{\prime}= \\
\text { poly: } 0=7 \cdot x^{3}+5 \cdot x^{2}-3 \cdot x-2 & 2.591
\end{array}
$$

| $f^{\prime}=$ | 2591 |
| :---: | :---: |
| poly: 0 $=7 \cdot x^{3}+5 \cdot x^{2}-3 \cdot x-2$ |  |
| $\times$ | f'here |

Note that f' returned the value of the $1^{\text {st }}$ derivative at this location immediately since $g$ features only one variable; else f' would have needed your input via the softkeys displayed and pressing $f^{\prime}$ here thereafter. So the slope of $g$ at $\boldsymbol{x}=\mathbf{- 0 . 8 0 6 4}$ is 2.591. Get the slopes at the two other root positions the same way:

EXIT returns to the top view of EQN as above.

## 3. Solver

Find the $2^{\text {nd }}$ root of $g:-7 \times \mathbf{0} \times \times$

$$
x=
$$

$-5.510 \times 10^{-1} \square$
EXIT returns to the top view of EQN as above.

## f' returns

$$
f^{\prime}=
$$

EXIT returns to the top view of EQN as above.

## 4. Solver

Find the $3^{\text {rd }}$ (rightmost) root of $\mathbf{g}: \mathbf{0} \times \mathbf{1 \times x}$
$\square=$
$6.431 \times 10^{-1}$
(EXIT) returns to the top view of EQN as above.

So the slope of $g$ at $x=-0.8064$ is 2.591, at $x=-0.5510$ it is $\mathbf{- 2 . 1 3 4}$, and at $\boldsymbol{x}=\mathbf{0 . 6 4 3 1}$ it is $\mathbf{1 2 . 1 1}$; so the sequence of slopes is positive, negative, and positive as expected.
$f$ " $(x)$ works in full analogy, computing the $2^{\text {nd }}$ derivative of the function.

## Interactively Differentiating Expressions Stored in Programs

Instead of operating on an equation as described in previous chapter, your WP43 can also derive an expression $f(\boldsymbol{x})$ stored in a program. Then,

1. Write a program for $\boldsymbol{f}(\boldsymbol{x})$. It must begin with a global label. For interactive derivation, we recommend proceeding as follows: ${ }^{229}$
From the $2^{\text {nd }}$ step of this program on, menu variables shall be declared using MVAR instructions (cf. p. 234) covering all variables of $\boldsymbol{f}(\boldsymbol{x})$. The subsequent body of the routine shall then evaluate $f(x)$ recalling these variables.
2. Optionally, write another program labeled ' $\delta x$ ' (see p. 271).
3. Enter values for all known variables (derivation constants) of $\boldsymbol{f}(\boldsymbol{x})$. Put the location where you want to know he derivative into $\mathbf{X}$.
4. Press ADV:

5. Press $f^{\prime}(x)$ or $f^{\prime \prime}(x)$. You will get:

6. Choose PROG to pick the label of the program containing the function $\boldsymbol{f}(\boldsymbol{x})$ (or enter its label directly as described on p. 219).
7. Let your WP43 compute the $1^{\text {st }}$ or $2^{\text {nd }}$ derivative at location $\boldsymbol{x}$.
[^122]
## Computing Derivatives in a Program

For computing derivatives in programs, proceed as demonstrated in previous chapter. Just remember you should omit the MVAR instructions in your program calculating $\boldsymbol{f}(\boldsymbol{x})$; instead, define the necessary variables in advance and load them with the known values using STO.

Call ADV in PEM:


Press $f^{\prime}(x)$ (or $f^{\prime \prime}(x)$ ): you will be asked for the label of your program calculating $f(x)$ - type it or pick it from the list as explained in steps 5 and 6 in previous chapter. Your WP43 will compute the requested derivative for you in this program step.

## Nesting Advanced Operations

You can nest SLV, $\int, f^{\prime}(x), f^{\prime \prime}(x), \Sigma$, and $\Pi$ in routines to any depth as far as memory allows and your patience and power last.

## Example:

Light is observed to be diffracted when passing through very small holes, an effect most obvious when using laser light. Its intensity behind a circular hole is

$$
I(r)=I_{0} \times\left(\frac{J_{1}(2 \pi r)}{\pi r}\right)^{2}
$$

at distance $r$ from the center of the beam, with

$$
J_{1}(x)=\frac{1}{\pi} \int_{0}^{\pi} \cos [t-x \sin (t)] d t
$$

being the Bessel function of the $1^{\text {st }}$ kind and order 1 (cf. p. 265). Find the first three roots of the intensity (i.e. the radii where no light will be observed).

## Solution:

1. Write a little program for the integrand $f(t)=\cos [t-x \sin (t)]$ :
```
LBL 'J1'
    RCL ' t '
    ENTER \(\uparrow\)
    \(\sin\)
    \(\sin (t)\)
    RCLx ' \(x\) '
    \(x \sin (t)\)
    -
    \(\cos\)
END
\(t-x \sin (t)\)
\(\cos [t-x \sin (t)]\)
```

2. Write a $2^{\text {nd }}$ little program for calculating the intensity $I(r)$. Actually, to determine its roots you just need solving $J_{1}(2 \pi r)$. And ADV called in PEM displays:

| PGMSLV |  | f"(x) |  |  | PGMINT |
| :---: | :---: | :---: | :---: | :---: | :---: |
| SOLVE | SLVQ | $\mathrm{f}^{\prime}(\mathrm{x})$ | $\mathrm{n}_{\mathrm{n}}$ | $\Sigma_{n}$ | $\int \mathrm{fdx}$ |

LBL 'II'
MVAR ' $r$ '
$1 . \times 10^{-5}$
STO 'ACC' cf. p. 267.
CLX
STO ' ZLIM '
$\pi$
STO '个LIM'
RCL× ' $r$ '
STO+ X
Tr

STO ' $x$ '
PGMINT 'J1'
$\int f d$ 't'
specifies the program of the integrand (see the menu). computes $\pi \times J_{1}(2 \pi r)$ and returns it in $\mathbf{X} .{ }^{230}$ END

## 3. Enter DISP SCI 3 MODE RAD

 ADV SOLVE|  |  |
| :--- | :--- | :--- | :--- | :--- | :--- |

4. Enter PROG (II). You will see
${ }^{230}$ Note that $I(x)=0 \Leftrightarrow J_{1}(x) / x=0 \Leftrightarrow J_{1}(x)=0$ for all $x \neq 0$.

# 5. Enter $.1 \begin{array}{llll}\mathbf{r} & \mathbf{r} & \mathbf{r}\end{array}$. You will get $6.098 \times 10^{-1}$ after some time. ${ }^{231}$ <br> 6. Enter $\mathbf{1} \mathbf{r} 1.5 \mathbf{r} \quad \mathbf{r}$ and you will get 1.117 <br> 7. Enter 1.5 r 2 r $\quad$ r and you will get 1.619 

Solving nested advanced operations may require very many calculations to be executed. We recommend connecting your WP43 to an USB outlet for supplying external power when dealing with such problems.

You will find further instructions and examples in HP-42S RPN Scientific Programming Examples and Techniques. Despite its title, this also comprises significant material about the Solver, Integrator, matrices, and statistics. Watch the differences between HP-42S and WP43 though.

[^123]The algorithm finds this root directly.

## SECTION 5: TWO BROWSERS, TWO APPLICATIONS, AND TWO SPECIAL MENUS

There are two browsers featured in your WP43 for quick and easy checking memory, registers, and flags (RBR and STATUS, see below). And there are two very useful applications: a TIMER (or stopwatch, see pp. 280f) and the "Time Value of Money" (TVM, see pp. 282ff). Furthermore, two special menus will ease your path in science and engineering and particular regions of this planet (see pp. 287ff).

## The Browsers RBR and STATUS

These two browsers may be called in all modes except alpha input. They are not programmable. Some special keys and special rules apply within these browsers as explained below.

(RBR browses all currently allocated registers, showing their contents. See here its virtual keyboard. It looks similar to the one in TAM (cf. pp. 60ff) but differs.


The $1^{\text {st }}$ view you will see after calling RBR covers registers $\mathbf{X} \ldots$ I (contents will deviate on your screen - note individual numbers are shown in the display format currently selected, while text strings may be abbreviated and matrices will be. Fractions are displayed with their decimal value):

| 2022-08-05 | $23: 15$ | $\mathbb{R L} 4^{\circ}$ |
| :--- | ---: | ---: |$\quad$ /max | 64:2 |
| :--- |
| I: |
| L: |

(4) goes up the stack, continuing with the remaining lettered registers, then with R00, R01, etc. as shown below. For R00 ... R99, every fifth register is displayed overlined to guide the eye:

After R99, X will be shown again.

$\nabla$ browses the registers going down from $\mathbf{R 9 9}$ (if starting with the screen above) to R00; then continues with K, J, down to $\mathbf{X}$. After $\mathbf{X}, \mathbf{R 9 9}$ will be shown again.
$\square$ turns to local registers if any are allocated, starting with $\mathbf{R} \mathbf{. 0 0}$. Then, $\triangle$ and $\nabla$ browse local registers up and down until another $\square$ turns to named variables (incl. system variables), and one more $\square$ returns to the $1^{\text {st }}$ screen of RBR as shown above. Else (i.e. if no local registers are allocated) $\square$ directly turns to named variables.

R/S toggles display to show the register contents or the space allocated for them (see the ReM, App. B).
(A) $\ldots$ (D) or (I) $\ldots$ (D) browses immediately to the corresponding register.

00 ... 99 browses immediately to the corresponding (global or local) register. If no such local register is allocated, it returns to the $1^{\text {st }} \mathrm{global}$ register. If pressed while named variables are displayed, it jumps to the corresponding global register.
(RCL in RUM recalls the register displayed in the lowest row and leaves RBR; in PEM, it enters a corresponding step RCL ... and leaves RBR.

SNAP dumps a snapshot of the present screen on the FAT disk. Note you can skip pressing $\square$ here.

EXIT leaves RBR.


FLAG STATUS shows the amount of free memory available and the user accessible flags set. ${ }^{232}$ See here its virtual keyboard. Local user flags will only be displayed if local registers are allocated.

Some global settings and system flags set are displayed in the bottom rows (covering


[^124]only what is not shown in the status bar):
$\Delta$ and $\nabla$ toggle between views if more flags are set.

SNAP dumps a snapshot of the present screen on the FAT disk. Note you can skip pressing here.

EXIT leaves STATUS.

## The Timer Application

Your WP43 provides a timer. See its virtual keyboard here.

Press (o start it.


Then the top numeric row will be replaced by 0:00:00.0 [00] , unless the timer was running before already (then the accumulated run time will replace zero here).

Then,

R/S starts or stops the timer without changing its value. ${ }^{233}$
0.1 s toggles displaying tenths of seconds (startup default is 'display').

RESET or resets the timer to zero without changing its status (running or stopped). It deletes the total time if applicable (see $\square$ ).

Note this is not the global RESET command, it just shares its label.
$\boldsymbol{\Sigma +}$ adds the present timer value (converted to a real number $\boldsymbol{y}$ ) and the data point number $(x=1,2,3, \ldots)$ to STATS.

00 ... 90 sets the current register address (CRA, startup default is $00)$. The CRA is displayed between rectangular brackets like

> 27:31:55.6 [08]

You shall enter both digits always. ${ }^{234}$ Since the CRA may be incremented automatically, set it so there will be sufficient unused registers following for your storage.

ENTERT stores the present timer value in the CRA without changing the timer status or value. Then increments the CRA by one.
( $\boldsymbol{\nabla}$ increments or decrements the CRA by one, respectively. $\square$ allows for overwriting the last time stored.
combines ENTERT and an immediately subsequent $\boldsymbol{T}$ in one keystroke; and the total time counted since the last explicit press of $\boldsymbol{\square}$ or RESET will then be displayed like

|  | $2: 04: 15^{\top}$ | $0: 02: 29$ |
| :---: | :---: | :---: |
| or $26: 02: 31.7^{\top}$ | $10: 00: 49.6$ | $[11]$ |

and updated from then on. $-\square$ allows for recording lap times

[^125]in a series of registers. Note the total time is volatile - it will vanish and be reset when $\boldsymbol{\Psi}$ or RESET is pressed.
$\pm$ combines the functionalities of $\Sigma \boldsymbol{\Sigma}$, $\operatorname{ENTERT}$, and $\boldsymbol{\Psi}$ in one keystroke. This allows for recording lap times and total time for later offline analysis, e.g. for computing the arithmetic mean and standard deviation of lap times after leaving TIMER.
(RCL $n n$ recalls $r n n$ and adds it to the time recorded without changing the timer status. Note the total time is not affected by this addition.

SNAP dumps a snapshot of the present screen on the FAT disk. Note you can skip pressing here.

EXIT leaves the application. The time string in the top numeric row will vanish. Unless stopped, however, the timer will continue incrementing in background (indicated by in the status bar) until ...
a) you return to TIMER and stop it explicitly by R/S or ...
b) you turn off your WP43 manually ${ }^{235}$ or ...
c) your WP43 shuts down due to low battery (supply power via USB).

The time counted can be indicated up to 99:59:59.9 - then the display will overflow. This applies to the total time as well.

For subtracting split times you have to leave this application.
TIMER is not programmable.

## The Time Value of Money (TVM)

TVM is a well proven financial application (thus found in FIN) computing e.g. the future value (FV) of

## $\Delta \%$ FIN

+/- L

1. a repeated investment or
2. a regular down-payment for a credit
${ }^{235}$ Note your WP43 will not turn off automatically with the timer running in foreground. If it is running in background, however, it will continue incrementing even if your WP43 is turned off automatically.
based on its present value $(P V)$, its interest rate per annum in percent (i\%/a), the required payment per period (PMT), the number of periods per annum (per/a), and the total number of payment periods $\left(\boldsymbol{n}_{\text {PER }}\right){ }^{236}$ This kind of financial problems will often occur also to technical people, so we included TVM in your WP43.

For your information, the general formula for such problems reads

$$
F V=-P V(1+i)^{n_{P E R}}+(1+i p) \frac{P M T}{i}\left[1-(1+i)^{n_{P E R}}\right]
$$

with the deduced parameter $i=\frac{I \% / \text { period }}{100}=\frac{I \% / \text { year }}{100} / \frac{\text { periods }}{\text { year }}$ and the binary switch value $p$ : If payments occur at the...

- end of each period then $p=0$ (choose End in TVM).
- beginning of each period then $p=1$ (choose Begin in TVM).

For the notation of the TVM formula, cf. f. 41 on p. 48.

## TVM uses the cash flow convention that cash outlays (payments) are input as negative, and cash incomes are entered as positive. ${ }^{237}$

The present value $P V$ always occurs at the beginning of the $1^{\text {st }}$ period. It can also be an initial cash flow or a discounted value of a series of future cash flows.

The future value $F V$ is always meant to occur at the end of the $\boldsymbol{n}_{\text {PER }}{ }^{\text {th }}$ period. It can also be a final cash flow or a compounded value of a series of cash flows.

[^126]${ }^{237}$ This convention was introduced with the HP-92.


Example for calculating the number of periods: ${ }^{238}$
A potential development site currently appraised at $\$ 380000$ appreciates at $30 \%$ per year. If this rate continues, how many years will it be before this land is worth $\$ 750000$ ?

Solution:
DISP FIX 2

This will suffice for all financial problems here. Then all you have to do is keying in the known parameters and boundary conditions:

## FIN TVM

| Begin |  |  |  |  | End |
| :---: | :---: | :---: | :---: | :---: | :---: |
| RCL $\mathrm{n}_{\mathrm{p}}$ | i\%/a | per/a | PV | PMT | RCL FV |
| $\mathrm{n}_{\text {PER }}$ | i\%/a | per/a | PV | PMT | FV |


| Begin |  | see top of previous page. |
| :---: | :---: | :---: |
| -380000 PV | -380 000.00 | present value paid, |
| 30 i\%/a | 30.00 | \% interest rate per year, |
| 0 PMT | 0.00 | no payments, ${ }^{239}$ |
| 1 per/a | 1.00 | default. |
| 750000 FV | 750000.00 | Now, how long does it take to reach this future value? |
| $\mathrm{n}_{\text {PER }}$ | 2.59 | years. |

[^127]${ }^{239}$ Attention: Such a calculation might return wrong results for $\mathrm{PMT} \neq 0$.


Solution:

| -6000 PV | -6000.00 | present value paid, |
| :--- | ---: | :--- |
| $\mathbf{4}$ per/a | 4.00 | quarters, |
| $\mathbf{1 0 0 0 0 ~ F V}$ | 10000.00 | future value, |
| $\mathbf{8}$ ENTERT $\mathbf{4} \boldsymbol{x}$ n $_{\text {PER }}$ | 32.00 | periods. Now, we need... |
| i\%/a | 6.44 | $\%$ interest rate per year to achieve this |




#### Abstract

Example for finding the present value of a compounded amount: In 5 years when your son starts college, you will need $\$ 20000$. You deposit a lump sum in a certificate account that earns 6\% compounded daily. How much do you need to deposit today to reach that goal? (Continue ...)


## Solution:

| $\mathbf{2 0 0 0 0 ~ F V}$ | 20000.00 | future value, |
| :--- | ---: | :--- |
| $\mathbf{6} \mathrm{i} \% / \mathrm{a}$ | 6.00 | $\%$ interest rate per year, |
| $\mathbf{3 6 5}$ per/a | 365.00 | days per year, |
| $\mathbf{5}$ ENTERT $\mathbf{3 6 5} \times \mathrm{n}_{\text {PER }}$ | 1825.00 | periods. Now, we need... |
| PV | -14816.73 | to be deposited. |

[^128]
## Example for finding the future value of a compounded amount:

FV (?) The local trading post manager


$\$ 1000$ (PV)
opened up a savings operation 5 years ago, offering 6\% compounded daily. Gold miner Yellowstone Sam deposited $\$ 1000$ at that time, and now wants to know his present balance and the total accrued interest after all this time. (Continue ...)

## Solution:

| $\mathbf{- 1 0 0 0}$ PV | -1000.00 | original deposit, |
| :--- | ---: | :--- |
| $\mathbf{6}$ i\%/a | 6.00 | $\%$ interest rate per year, |
| $\mathbf{3 6 5}$ per/a | 365.00 | days per year, |
| $\mathbf{5}$ ENTERT $\mathbf{3 6 5} \boldsymbol{x}$ neER | 1825.00 | periods. Now, Sam has... |
| FV | 1349.83 | present balance meaning... |
| PV $\oplus$ | 349.83 | accrued interest. |

Note that the irrational adding of PV here is due to the cash flow convention of p. 282 (cf. the $2^{\text {nd }}$ paragraph of f. 140 on pp. 133f).

Nominal interest rate converted to effective rate:

Example for finding the effective annual interest rate:
What is the effective annual rate of interest if the annual nominal rate of $12 \%$ is compounded quarterly? (Continue ...)

## Solution:

| $\mathbf{- 1 0 0}$ PV | -100.00 | base value, |
| :--- | ---: | :--- | :--- |
| $\mathbf{1 2}$ i\%/a | 12.00 | \% nominal rate per year, |
| $\mathbf{4}$ per/a | 4.00 | quarters per year, |
| $\mathbf{4} \mathrm{n}_{\text {PER }}$ | 4.00 | compound periods; |
| FV | 112.55 |  |
| PV $\oplus$ | 12.55 | \% effective interest rate. |

Turn to App. 3 (on pp. 329ff) for more applications of TVM (annuities etc.).

## The Catalog of Constants

Your WP43 contains 80 physical, astronomical, and mathematical constants, sorted alphabetically in CONST. Pressing CONST you will get:


Besides by browsing with $\Delta$ and $\nabla$, you can reach the constants most easily using the alphabetical access method demonstrated in ReMS 2.

Names of astronomical and mathematical constants are printed on colored background in the table starting below. The unit of each physical and astronomical constant is listed here as well. Find the numeric values of the constants and their uncertainties as well as all unit symbols used here explained in ReMS 2.

| Name | Unit | Remarks |
| :---: | :---: | :---: |
| a | d | Gregorian year |
| $a_{0}$ | m | Bohr radius |
| $a_{\text {moon }}$ | m | Semi-major axis of the Moon's orbit around Earth. |
| $\boldsymbol{a}_{\oplus}$ | m | Semi-major axis of the Earth orbit around the Sun. Within its uncertainty, $\mathbf{a}_{\oplus}$ equals 1 AU (astronomic unit). |
| c | $\mathrm{m} / \mathrm{s}$ | Speed of light in vacuum |
| $C_{1}$ | $\mathrm{m}^{2} \mathrm{~W}$ | $1^{\text {st }}$ radiation constant |
| $\mathrm{C}_{2}$ | m K |  |
| e | C | Elementary charge |
| $e_{E}$ | 1 | Euler's e |
| F | C/mol | Faraday constant |
| $\mathrm{F}_{\alpha}$ | 1 | Feigenbaum's $\alpha$ and $\delta$ |
| $\mathrm{F}_{\delta}$ |  |  |


| Name | Unit | Remarks |
| :---: | :---: | :---: |
| G | $\mathrm{m}^{3} / \mathrm{kg} \mathrm{~s}^{2}$ | Newtonian constant of gravitation; also known as $\gamma$ from other authors. See also $\mathbf{G M}_{\oplus}$ below. |
| $\mathrm{G}_{0}$ | $1 / \Omega$ | Conductance quantum |
| $\mathrm{G}_{\mathrm{c}}$ | 1 | Catalan's constant |
| $\mathrm{g}_{\mathrm{e}}$ | 1 | Landé's electron g-factor |
| $\mathrm{GM}_{\oplus}$ | $\mathrm{m}^{3} / \mathrm{s}^{2}$ | Geocentric gravitational constant (according to the Earth model WGS84 - see the ReM) |
| $g_{\oplus}$ | $\mathrm{m} / \mathrm{s}^{2}$ | Standard Earth gravity acceleration |
| h | J s | Planck constant, a.k.a. elementary quantum of action |
| ћ | J s | So-called 'Dirac constant', actually just $h / 2 \pi$ |
| k | J/K | Boltzmann constant |
| K」 | Hz/V | Josephson constant |
| $l_{p}$ | m | Planck length |
| $\mathrm{m}_{\mathrm{e}}$ | kg | Electron mass |
| M Moon | kg | Mass of the Earth's Moon |
| $\mathrm{m}_{\mathrm{n}}$ | kg | Neutron mass |
| $\mathrm{m}_{\mathrm{p}}$ | kg | Proton mass |
| $M_{p}$ | kg | Planck mass |
| $\mathrm{m}_{\mathrm{p}} / \mathrm{m}_{\mathrm{e}}$ | 1 | Proton to electron mass ratio |
| $\mathrm{m}_{u}$ | kg | Atomic mass constant |
| $m_{u} c^{2}$ | J | Energy equivalent of atomic mass constant |
| $\mathrm{m}_{\mathrm{p}}$ | kg | Muon mass |
| $M_{\odot}$ | kg | Mass of the Sun |
| $M_{\oplus}$ | kg | Mass of Earth. See also $\mathrm{GM}_{\oplus}$ above. |
| $\mathrm{N}_{\text {A }}$ | 1/mol | Avogadro's number |


| Name | Unit | Remarks |
| :---: | :---: | :---: |
| NaN |  | NaN means "Not a Number", i.e. e.g. $0 / 0$ or $\ln (x)$ for $x<0$ or $\tan \left(90^{\circ}\right)$ unless in complex domain. <br> NaN covers poles as well as regions where a function result is not defined at all. Note that infinities, on the other hand, are considered numeric in your WP43 (see the end of this table). Non-numeric results will lead to an error message thrown unless SPCRES (or flag (D) is set; NaN allows that functions written by you can return it instead. |
| $\mathrm{P}_{0}$ | Pa | Standard atmospheric pressure |
| R | J/mol K | Molar gas constant |
| $\mathrm{r}_{\mathrm{e}}$ | m | Classical electron radius |
| $\mathrm{R}_{\mathrm{K}}$ | $\Omega$ | Von Klitzing constant |
| $\mathrm{R}_{\text {Moon }}$ | m | Mean radius of the Moon |
| $\mathrm{R}_{\infty}$ | 1/m | Rydberg constant |
| $\mathbf{R}_{\odot}$ | m | Mean radius of the Sun |
| $\mathbf{R}_{\oplus}$ |  | Mean radius of the Earth |
| Sa | m | Semi-major axis |
| Sb |  | Semi-minor axis according to the Earth |
| $\mathrm{Se}^{2}$ | 1 | $1^{\text {st }}$ eccentricity squared <br> model WGS84 (see |
| $S e^{\prime 2}$ |  | $2^{\text {nd }}$ eccentricity squared $\quad$ the ReM) |
| $S f^{-1}$ |  | Flattening parameter |
| $\mathrm{T}_{0}$ | K | $=0^{\circ} \mathrm{C}$, standard temperature |
| $t_{p}$ | S | Planck time |
| $\mathrm{T}_{\mathrm{p}}$ | K | Planck temperature |
| $V_{m}$ | $\mathrm{m}^{3} / \mathrm{mol}$ | Molar volume of an ideal gas at standard conditions |
| $\mathrm{Z}_{0}$ | $\Omega$ | Characteristic impedance of vacuum |
| $\alpha$ | 1 | Fine-structure constant |


| Name | Unit | Remarks |
| :---: | :---: | :---: |
| $\gamma$ | $\mathrm{m}^{3} / \mathrm{kg} \mathrm{s}^{2}$ | Newtonian constant of gravitation; also known as G from other authors. See also $\mathrm{GM}_{\oplus}$ above. |
| $\gamma_{\text {EM }}$ | 1 | Euler-Mascheroni constant |
| $\gamma_{p}$ | ${ }^{\mathrm{Hz}} / \mathrm{T}$ | Proton gyromagnetic ratio |
| $\Delta v_{\text {cs }}$ | Hz | Hyperfine transition frequency of ${ }^{133} \mathrm{Cs}$ |
| $\varepsilon_{0}$ | F/m | Electric constant or vacuum permittivity |
| $\lambda_{c}$ | m | Compton wavelengths of the electron, neutron, and proton |
| $\lambda_{c n}$ |  |  |
| $\lambda_{\text {cp }}$ |  |  |
| $\mu_{0}$ | H/m | Magnetic constant or vacuum permeability |
| $\mu_{B}$ | ${ }^{\mathrm{J}} / \mathrm{T}$ | Bohr magneton |
| $\mu_{\text {e }}$ |  | Electron magnetic moment |
| $\mu_{\mathrm{e}} / \mu_{\mathrm{B}}$ | 1 | Ratio of electron magnetic moment to Bohr's magneton |
| $\mu_{n}$ | $\mathrm{J}^{\text {/ }}$ | Neutron and proton magnetic moment |
| $\mu_{p}$ |  |  |
| $\mu_{u}$ |  | Nuclear magneton |
| $\mu_{\mu}$ |  | Muon magnetic moment |
| $\sigma_{B}$ | $\mathrm{W} / \mathrm{m}^{2} \mathrm{~K}^{4}$ | Stefan-Boltzmann constant |
| Ф | 1 |  |
| $\Phi_{0}$ | Wb | Magnetic flux quantum |
| $\omega$ | $\mathrm{rad} / \mathrm{s}$ | Angular velocity of Earth according to the Earth mode WGS84 (see the ReM) |
| $-\infty$ $\infty$ | 1 | Both these infinities are counted as special numeric values in your WP43. If SPCRES (or flag (D) is set then infinities resulting from calculations can be transferred, else they will lead to error messages thrown (cf. $\mathbf{N a N}$ ). |

Employ the constants stored here for further useful equivalences, e.g.:

- express joules in electron-volts $\left(1 \mathrm{eV} \approx 1.602 \times 10^{-19} \mathrm{~A} \mathrm{SV} \Rightarrow 1 \mathrm{~J} \approx\right.$ $\left.6.24 \times 10^{18} \mathrm{eV}=6.24 \times 10^{9} \mathrm{GeV}\right)$,
- calculate the wavelength from the frequency of electromagnetic radiation via $\lambda=c / v$ (so 100 THz correspond to ca. $3 \mu \mathrm{~m}$ ),
- determine the energy of electromagnetic radiation from its frequency via $E=h v \quad$ (so $1 \mathrm{THz} \times h=6.63 \times 10^{-22} \mathrm{~J}=4.14 \times 10^{-3} \mathrm{eV}$ ). Thus, 1 eV corresponds to 241.8 THz (or a wavelength of $1.24 \mu \mathrm{~m}$ ).


## Another example:

Remember Albert Einstein's most popular formula $E=m c^{2}$ ? So if you want to see the energy equivalent (in electron-volts) of one of the small particle masses given in kg above, multiply its mass by

$$
c^{2} / e \approx 5.610 \times 10^{35} \mathrm{~m}^{2} / \mathrm{As}^{3}
$$

and you are done: $m_{e} c^{2} / e=510999 \mathrm{eV}=511 \mathrm{keV}$, for instance, $m_{p}$ corresponds to 938.3 MeV , etc.

One more final example:
Assume American advanced scientists will succeed in producing a tiny bit of anti-matter in one of their high-tech laboratories one day - let's say $0.1 \mu \mathrm{~g}$ of anti-hydrogen, carefully stored isolated in ultra-high vacuum (UHV).
Although in future, most probably US power transmission lines will still look like they do today since this is a well-tried American standard. Thus, under slightly extreme weather conditions, ${ }^{241}$ an accidental blackout may easily happen for some days - the electric vacuum pumps will run down and stop working, and a subsequent vacuum breakdown will let atmospheric gas leak into the shiny vacuum vessel where it will interact with the precious anti-matter and annihilate immediately. How much energy is going to be released then?

## Solution:

You only need the same tiny amount of (common) matter, so $0.2 \mu \mathrm{~g}$ will annihilate in total within the vessel. $1 \mu \mathrm{~g}=10^{-6} \mathrm{~g}=10^{-9} \mathrm{~kg}$. Thus enter:

```
DISP ENG 3
```


### 0.000

[^129]| . 2 E + 9 | $0.2 \times 10^{-9}$ |
| :---: | :---: |
| CONST c | $299.8 \times 10^{6}$ |
| ENTERT $\boldsymbol{x}$ | $89.88 \times 10^{15}$ |
| - | $17.98 \times 10^{6}$ |

... resulting in 18 MJ set free. The odds are frightening high this lab will need no cleaning in next weeks. ${ }^{242}$
On the other hand, $0.1 \mu \mathrm{~g}$ of anti-matter require e.g. $N_{A} / 10^{7}$ atoms of anti-hydrogen (with $N_{A}$ being Avogadro's number). This means $6 \times 10^{16}$ atoms or $3 \times 10^{16}$ molecules of this gas (i.e. 30000 million millions of anti-hydrogen molecules). Luckily, this amount is far from being produced in any lab for the time being. ${ }^{243}$

## Unit Conversions

Your WP43 features 14 angular conversions stored in $\underset{\rightarrow}{ }$ (as shown on p. 140) and 112 unit conversions in $\underline{U} \rightarrow$. The latter menu mainly provides means to convert local to common units and vice versa. ${ }^{244}$


Note also the constant $T_{0}$ found in CONST may be useful for converting centigrade temperatures to kelvin. It is not repeated in $\underline{U} \rightarrow$ since it is only added or subtracted in temperature conversions.

[^130]
$\underline{U} \rightarrow$ is structured like a tree. Press $U \rightarrow$ and you will see this view:

containing the labels of the branches for conversions of Energy, Power, Force \& pressure, mass, length $(\mathbf{x})$, Area, and Volume units. The entire structure of $\underline{U \rightarrow}$ is shown on the next two pages (with the menu rows printed top down following common reading habits). Some conversion labels require more than 6 characters due to long unit names - then extra high menu rows will be displayed:

|  | F1 | F2 | F3 | F4 | F5 | F6 | Remarks |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\underline{\mathrm{U}}$ : | E: | P: | year $\rightarrow$ s | F\&p: | m: | X: | submenu headers, units of temperature, time, torque, and ratios - the latter two in an extrahigh menu row |
|  | ${ }^{\circ} \mathrm{C} \rightarrow{ }^{\circ} \mathrm{F}$ | ${ }^{\circ} \mathrm{F} \rightarrow{ }^{\circ} \mathrm{C}$ | $s \rightarrow$ year |  | V: | A: |  |
|  | power <br> ratio <br> $\rightarrow \mathrm{dB}$ | $\mathrm{dB} \rightarrow$ power ratio | $\underset{\mathrm{lbf} \cdot \mathrm{ft}}{\stackrel{\mathrm{Nm}}{\mathrm{l}}}$ | $\text { lbf•ft } \rightarrow$ $\mathrm{Nm}$ | field <br> ratio <br> $\rightarrow \mathrm{dB}$ | $\mathrm{dB} \rightarrow$ field ratio |  |
| E: | cal $\rightarrow$ J | $\mathrm{J} \rightarrow \mathrm{cal}$ | Btu $\rightarrow$ J | $J \rightarrow$ Btu | Wh $\rightarrow$ J | $J \rightarrow W h$ | units of energy |
| P: | $h p_{E} \rightarrow W$ | $W \rightarrow h p_{E}$ | $\mathrm{hP}_{\mathrm{uk}} \rightarrow \mathrm{W}$ | $W \rightarrow h p_{u k}$ | $\mathrm{hp}_{\mathrm{M}} \rightarrow \mathrm{W}$ | $W \rightarrow h p_{M}$ | units of power |
| F\&p: | $\mathrm{lbf} \rightarrow \mathrm{N}$ | $\mathrm{N} \rightarrow \mathrm{lbf}$ | bar $\rightarrow \mathrm{Pa}$ | Pa $\rightarrow$ bar | psi $\rightarrow$ Pa | Pa $\rightarrow$ psi | units of force and pressure |
|  | $\underset{\rightarrow}{\substack{\mathrm{in} . \mathrm{Hg} \\ \rightarrow \mathrm{~Pa}}}$ | $\begin{aligned} & \hline \mathrm{Pa} \vec{~} \\ & \text { in. } \mathrm{Hg} \end{aligned}$ | $\begin{aligned} & \text { torr } \\ & \rightarrow \mathrm{Pa} \end{aligned}$ | $\mathrm{Pa} \rightarrow$ torr | atm $\rightarrow \mathrm{Pa}$ | Pa $\rightarrow$ atm |  |
|  |  |  | $\underset{\rightarrow \mathrm{Pa}}{\underset{\mathrm{mmHg}}{ }}$ | $\mathrm{Pa} \rightarrow$ mmHg |  |  |  |

conversions while working within $S I$. Thus, most of the material appearing in $\underline{U \rightarrow}$ will look quirky to obsolete for the overwhelming majority of mankind.
Some old units die hard, however, in some corners of this world (English is spoken in all of those). Thus, $\underline{U \rightarrow}$ will also help when you get caught in a time loop and happen to be thrown back into such an obstinate environment. $\underline{\rightarrow}$ may also give you a slight idea of the mess we had in the world of measuring before going metric following the French Revolution over 220 years ago. For symmetry reasons, we included also 9 (refined) traditional Chinese units in $\underline{\underline{U}}$.
Without Imperial, US-American, and Chinese units, $\underline{\underline{U}}$ would feature 18 entries only.

|  | F1 | F2 | F3 | F4 | F5 | F6 | Remarks |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| m: | lb. $\rightarrow$ kg | $\mathrm{kg} \rightarrow \mathrm{lb}$. | cwt $\rightarrow$ kg | kg $\rightarrow$ cwt | oz $\rightarrow$ g | $g \rightarrow 0 z$ | units of mass |
|  | stone <br> $\rightarrow \mathrm{kg}$ | $\mathrm{kg} \rightarrow$ stone | $\begin{array}{c\|} \hline \text { short } \\ \text { cwt } \rightarrow \mathrm{kg} \\ \hline \end{array}$ | $\mathrm{kg} \rightarrow$ sh.cwt | $\begin{gathered} \text { tr.oz } \\ \rightarrow \mathrm{g} \\ \hline \end{gathered}$ | $\underset{\text { tr.oz }}{\mathrm{g} \rightarrow}$ |  |
|  | ton $\rightarrow$ kg | $\mathrm{kg} \rightarrow$ ton | short ton$\rightarrow \mathrm{kg}$ | $\mathrm{kg} \rightarrow$ short ton | carat $\rightarrow \mathrm{g}$ |  |  |
|  | liäng $\rightarrow \mathrm{kg}$ | $\begin{aligned} & \mathrm{kg} \rightarrow \\ & \text { liäng } \end{aligned}$ |  |  | jin $\rightarrow$ kg | $\mathrm{kg} \rightarrow \mathrm{jin}$ |  |
| $\underline{x}$ : | $a u \rightarrow m$ | $\mathrm{m} \rightarrow \mathrm{au}$ | l.y. $\rightarrow$ m | $\mathrm{m} \rightarrow$ l.y. | $\mathrm{pc} \rightarrow \mathrm{m}$ | $\mathrm{m} \rightarrow \mathrm{pc}$ | units of length |
|  | $\mathrm{mi} . \rightarrow \mathrm{m}$ | $\mathrm{m} \rightarrow \mathrm{mi}$. | $\mathrm{nmi} \rightarrow \mathrm{m}$ | $\mathrm{m} \rightarrow \mathrm{nmi}$ | $f t . \rightarrow m$ | $m \rightarrow f t$. |  |
|  | in. $\rightarrow$ mm | mm $\rightarrow$ in. |  |  | yd. $\rightarrow \mathrm{m}$ | $\mathrm{m} \rightarrow \mathrm{yd}$. |  |
|  | $\underline{l i} \rightarrow$ m | $\mathrm{m} \rightarrow \mathrm{li}$ | yǐn $\rightarrow$ m | $m \rightarrow$ yin | zhàng $\rightarrow \mathrm{m}$ | $\mathrm{m} \rightarrow$ zhàng |  |
|  | chǐ $\rightarrow$ m | $\mathrm{m} \rightarrow \mathrm{chi}$ | cùn $\rightarrow$ m | $\mathrm{m} \rightarrow$ cùn | fēn $\rightarrow m$ | $m \rightarrow$ fēn |  |
|  | point <br> $\rightarrow$ mm | $\mathrm{mm} \rightarrow$ point | fathom $\rightarrow$ m | m $\rightarrow$ <br> fathom | survey foot $\rightarrow$ m | m $\rightarrow$ survey foot |  |
|  | mi. $\rightarrow$ km | $\mathrm{km} \rightarrow \mathrm{mi}$. | nmi $\rightarrow$ km | $\mathrm{km} \rightarrow \mathrm{nmi}$ |  |  |  |
| A: | $\begin{aligned} & \text { acre } \\ & \rightarrow \text { ha } \end{aligned}$ | $\text { ha } \rightarrow$ <br> acre | $\mathrm{ha} \rightarrow \mathrm{m}^{2}$ | $\mathrm{m}^{2} \rightarrow \mathrm{ha}$ | $\begin{gathered} \text { acre }_{\text {us }} \\ \rightarrow \text { ha } \end{gathered}$ | ha $\rightarrow$ acre $_{\text {us }}$ | units of area |
|  | mǔ $\rightarrow \mathrm{m}^{2}$ | $\mathrm{m}^{2} \rightarrow \mathrm{mu}$ |  |  |  |  |  |
| V : | $\mathrm{gal}_{\text {UK }} \rightarrow$ l | $l \rightarrow \mathrm{gal}_{\mathrm{UK}}$ | qt. $\rightarrow$ l | $l \rightarrow q t$. | $\mathrm{gal}_{\text {Us }} \rightarrow$ l | $l \rightarrow \mathrm{gal}_{\text {us }}$ | units of volume |
|  | $\mathrm{floz}_{\mathrm{UK}}$ $\rightarrow \mathrm{ml}$ | $\begin{gathered} \mathrm{ml} \rightarrow \\ \mathrm{floz}_{\mathrm{UK}} \end{gathered}$ | barrel $\rightarrow \mathrm{m}^{3}$ | $\mathrm{m}^{3} \rightarrow$ <br> barrel | floz $_{\text {US }}$ $\rightarrow \mathrm{ml}$ | $\begin{aligned} & \mathrm{ml} \rightarrow \\ & \mathrm{floz}_{\mathrm{US}} \end{aligned}$ |  |

In ReMS 2, you will find detailed explanations of the units used in these conversions.

You may combine conversions as you like. Units are matched carefully. ${ }^{245}$ (DISP) ENG [2 will do for all examples here:

[^131]
## Example 1:

For filling your tires with a maximum pressure of 30 psi the following will help you at gas stations in Europe and beyond:
30
Pa $\rightarrow$ bar returns $\quad$ psi $\rightarrow \mathrm{Pa} \quad 207 . \times 10^{3} \mathrm{~Pa}$.

Now you can set the filler and will not blow your tires.

## Example 2:

Your friend tells you she has got 10 cubic feet of debris on her veranda after flooding (the dams in the Mississippi delta turned out being of less use than once thought). What does this mean in real units?

| $1 \\|$ xi ft. $\rightarrow$ m | returns | $305 . \times 10^{-3}$ |
| :---: | :---: | :---: |
| $3 y^{x}$ |  | $28.3 \times 10^{-3}$ |
| $10 \times$ |  | $283 . \times 10^{-3}$ |

This is less than a third of a cubic meter. OK, some work - but manageable.

## Example 3:

A network switch is specified for $3320 \mathrm{Btu} / \mathrm{h}$. What?!?

$$
3320 \bigcup \text { E: Btu } \rightarrow \mathrm{J} \text { returns } \quad 3.50 \times 10^{6} \mathrm{~J} / \mathrm{h} .
$$

Since $1 J=c_{c} W h \Leftrightarrow 1 J / h=c_{c} W$ applies, you can use
$J \rightarrow$ Wh for converting and get 973. W.

This is almost 1 kW . Now you know what will be going on there.

## Example 4:

In OMS 2, there was an example ending with a box featuring a volume of 19 11/16 cubic inches. So, what does this volume mean in real units instead? And how much water can such a box contain in areas where people are condemned to deal with Imperial units nowadays still?

| $1 \triangle$ x: $\triangle \mathrm{in} . \rightarrow \mathrm{mm}$ | returns | 25.4 |
| :---: | :---: | :---: |
| $3 y^{x}$ |  | $16.4 \times 10^{3}$ |
| $19 \odot 11 \odot 16$ x |  | $323 . \times 10^{3}$ |

Since $1 \mathrm{~mm}^{3}=10^{-3} \mathrm{~cm}^{3}$ and $1 \mathrm{~cm}^{3}=1 \mathrm{ml}$,
1000 (1)
returns
323. $\mathrm{ml} \approx 1 / 3$ liter.

And to help those enduring life on the British Imperial islands or ex-territories, you must (!) ask them for their location first. Then choose either $\left(\mathrm{U} \rightarrow \mathbf{V}: \mathbf{m l} \rightarrow \mathrm{floz}_{\mathbf{u k}}\right.$ or $\mathbf{m l} \rightarrow \mathrm{floz}_{\mathbf{u s}}$, and give them the respective result, i.e. 11.4 or 10.9 , for what it is worth.

## Example 5:

A celestial object moves with a velocity of 0.1 parsec per year. What does this mean in standard units? What does it mean in relation to the velocity of light? And how does this translate for air pilots?

| . $1 \cup$ X: $\mathrm{pc} \rightarrow \mathrm{m}$ | returns | $3.09 \times 1015$ |
| :---: | :---: | :---: |
| EXIT | returns to the top view of $\underline{U}$. |  |
| 1 year $\rightarrow$ s (1) | returns | $97.8 \times 10^{6} \mathrm{~m}$ |
| CONST c 1 | returns | $326 . \times 10^{-3}=$ |

Since $1 \mathrm{~h}=60 \times 60 \mathrm{~s}=3600 \mathrm{~s} \Leftrightarrow 1 \mathrm{~m} / \mathrm{s}=3600 \mathrm{~m} / \mathrm{h}$.

Thus, RCL (L) $\boldsymbol{x}$
3600 X
... corresponding to $U \rightarrow \mathbf{X}: m \rightarrow n m i$
$97.8 \times 10^{6} \mathrm{~m} / \mathrm{s}$. $352 . \times 10^{9} \mathrm{~m} / \mathrm{h}$.
$190 . \times 10^{6} \mathrm{nmi} / \mathrm{h}$ or 190 mega-knots. ${ }^{246}$

Supported by your WP43, you will find further easy ways to produce whatever conversion you may need.

In emergencies of a particular kind, it may be helpful knowing that ...

- becquerel ( Bq ) equals hertz in your Geiger-Müller counter,
- $\operatorname{gray}(\mathrm{Gy})$ is the unit for deposited or absorbed energy (see the ReM),
- and sievert $(\mathrm{Sv})$ is gray times a radiation dependent dose conversion factor ( $\geq 1$ ) for the damage caused in biological material including human bodies (remember also the example on pp. 97f). ${ }^{247}$

[^132]Also in this field, some outdated units may still be found in literature:

- For those admiring the very $1^{\text {st }}$ Nobel laureate in physics (in 1901), Wilhelm Conrad Röntgen, for discovering the X-rays (ruining his hands in those experiments since he could not know better yet), the charge generated by radiation in matter was measured by the unit roentgen $\left.\left.1 R=2,58 \cdot 10^{-4} \mathrm{~A} \mathrm{~s} / \mathrm{kg}\right)\right)^{248}$
- Pour les fidèles amis et admirateurs de Madame Marie Skłodowska Curie (1903 Nobel laureate in physics and 1911 in chemistry), there was a unit curie with $1 \mathrm{Ci}=3,7 \cdot 10^{10} \mathrm{~Bq}=3,7 \cdot 10^{10}$ decays $/ \mathrm{s}$. You can deduct from the size of this unit that larger pieces of radioactive material were 'absolutely no problem' for the pioneers in this field. How could they know? ${ }^{249}$
- A few decades ago, rem (i.e. roentgen equivalent in men ${ }^{250}$ ) measured what sievert does today ( $1 \mathrm{rem}=10 \mathrm{mSv}$ ).
- And $1 \mathrm{~Gy}=100 \mathrm{rad}$ (radiation absorbed dose), which is a lot since there is almost nothing greater than millirad in literature (but see the map overleaf showing iodine-131 fallout exposure ${ }^{251}$ ):

Islands**** (e.g. Bikini, Eniwetok), Mexico, Mururoa,* the Netherlands, North Korea, Pakistan, the Philippines, Romania, Russia, Slovakia, Slovenia, South Africa, South Korea, Spain, Sweden, Switzerland, Taiwan, Tibet,***** the Ukraine,*** the United Arab Emirates, the United Kingdom, the USA, and Xinjiang-Uighur***** (in alphabetical order).
(The countries marked with stars suffer from actions of their so-called 'motherlands' at times between 1945 and today - spotting those colonialists is left as an exercise for the reader. The other countries listed here controlled their industry or military in a way that activated areas within their own territory could or can happen.)
After all, mankind gathers experience with radioactivity. See also the map overleaf.
${ }^{248}$ Conrad Röntgen (1845-1923) died from carcinoma of the intestine, aged 77.
${ }^{249}$ Marie Curie (1867-1934) died from aplastic anemia, aged 66. At the bottom line, it seems that Conrad, although earlier, was either more cautious or just luckier with respect to collateral health damages in his experiments than Marie.

250 This unit must be outdated - it is not regarded gender equitable nowadays anymore.
251 Picture source: Study Estimating Thyroid Doses of I-131 Received by Americans From Nevada Atmospheric Nuclear Bomb Test, National Cancer Institute (1997). See https://commons.wikimedia.org/w/index.php?curid=524198. Not everything on this map is plausible though - north of Montana seem to live no Americans, for instance.


## SECTION 6: CREATING YOUR VERY PERSONAL WP43

Your WP43 is the $1^{\text {st }}$ calculator worldwide allowing for fully customizing the user interface; i.e. you may assign an arbitrary function to almost any location, unshifted or shifted, on the keyboard or in a menu. User mode will then bring your personal assignments to the front, so you can interact with your WP43 via a user interface you designed yourself. Use ASSIGN for storing your personal favorite assignments. ASN allows for reassigning the entire keyboard except $\square, ~(a$, and F1 - F6). We sincerely recommend keeping also other basic functionalities accessible (see pp. 308f for some caveats).

In the explanations starting overleaf, ...

- $\square$ stands for the softkey applicable (optionally headed by $\square$ or $\square$ ),
- [key] represents an arbitrary labelled key of your WP43 (optionally headed by or $\square$ ), and
- name is the name of an almost arbitrary item (be it an operation, function, digit, symbol, routine (i.e. global label), variable, or system flag). A newly created name must follow the rules on p. 59.
Such an item name may be
a) keyed in spelling it via $\propto$ ENTERT ${ }^{252}$ or
b) picked from CATALOG or any other menu or
c) called by accessing the respective key or label on the keyboard (except for ENTER $\boldsymbol{T}$, EXIT $, \boldsymbol{\bullet}, \boldsymbol{\Delta}$, and $\boldsymbol{\nabla}$, see below)

Just pressing ENTERT where the operating system expects an item name is interpreted as an empty name (echoed NULL) and will return to the startup default at the location specified thereafter.

[^133]
## Assigning Your Favorite Functions

Here is how you can tailor the surface of your WP43 according to your individual preferences:

ASN name [key]
will assign this named item to [key] in user mode. It will throw an error if a name keyed in does not exist.
(ASN) name $\square$ in particular will assign this item to the respective softkey of the user menu displayed at the time you press this $\square$ (preceded by or $\square$, if applicable), replacing the label shown there before. It will throw an error if the target is not within a user menu.

Each user assignment will hold until it is overwritten by a new assignment or ENTERT is entered for name (see previous chapter).

Note that all user assignments will be visible and accessible in user mode only (see pp. 308ff) - except the items assigned to the two provided user menus MyMenu and Mya (see below).

## Example 1:

Let's assign the statistical sample standard error to $\square+\mathbf{C C}$ (this location is assigned to $\Varangle$ in startup default). There are three different ways to do this (specified printing all keystrokes necessary here):
a)

$\square$ (T)IM ENTER T

b)


FCNS (S) $\mathbf{s}_{\mathrm{m}}$
This way will work always - even if you do not remember where the target function is stored and how it is spelled exactly. It is demonstrated step by step starting below.
c)

$\square$ STAI $\mathbf{s}_{\mathbf{m}}$ $\square$
On the other hand (and as introduced on previous page),
$\square$ ENTERT
will reset $\square$-shifted (CC) to factory default $[x$ ) as explained at the very end of previous chapter.

We will walk you through solution b) step by step here, starting with a clear stack (press 0 (FILL if necessary). Only the menu section and the command echo row will be shown in the following since all action will take place there:


This is the top view of CATALOG'FCNS. Now enter the $1^{\text {st }}$ letter of the requested command:
©

| ASSIGN - - |
| :--- |

Quickly entering the $2^{\text {nd }}$ letter helps substantially:
M) (if you find you waited too long before pressing (M) just wait another few seconds, then key in (S)(M) quickly here instead)

| STOP | STOS | STO+ | STO- | ST0x | STO/ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| SSIZE? | STATUS | STO | STOCFG | STOEL | STOIJ |
| $\mathbf{s}_{\text {m }}$ | $\mathbf{S}_{\text {mw }}$ | SNAP | SOLVE | SPEC? | SR |

Now, press F1 for the function to be assigned:

... and this assignment is done. Note this last menu view will stay on screen until another view or menu or USER is called or this submenu is EXITed explicitly. And the function $\Varangle$ will stay accessible also in user mode via (CATALOG FCNS $\ddagger$. You can verify your assignment by turning to user mode via USER and pressing ( $£$ ) which will be echoed $\mathbf{s}_{\mathbf{m}}$ now.

Example 2 (assume startup default settings):
Assign the weighted arithmetic mean to the $1^{\text {st }}$ key in MyMenu:


Note that pressing USER here will exit all menus being open at that time and bring MyMenu on the screen (which is empty still).

## F1



Summarizing, ASN STAT $\bar{x}_{w}$ USER F1 did this assignment.
Note MyMenu will show up whenever all other menus are exited, ${ }^{253}$ i.e.

- either after you left all applicable submenus and pressed EXIT in the parent menu
- or after you pressed EXIT longer than 1 second.

MyMenu will then stay displayed until another menu is called. This holds for your WP43 being in user mode or not (see below).

Thus, filling MyMenu may well be the very $1^{\text {st }}$ step of customizing your WP43. You may, for instance, put the six trigonometric functions into the unshifted row of MyMenu and will have them almost always at hand.

As mentioned above, some keys and the menus need special treatment for assigning. Thus, the general rule for ASSIGN reads as listed in the table below (the last row of each column contains the standard case dealt with so far):

[^134]| ASSIGN | ... this functionality | ... to this destination |
| :---: | :---: | :---: |
| ASN |  |  |
|  | ( 0 (E)XDT ( ENTERT |  |
|  | (m) (menu_)name ${ }^{255}$ ENTER ${ }^{\text {( }}$ | (a) EENTTE]R ENTERT |
|  | O T ENTERT | (O) EEXITT ENTERT |
|  | ( $\triangle$ | (0) $\square$ |
|  | ( 4 | (a) $\triangle$ |
|  | [key] ${ }^{256}$ | ( 4 |
|  |  | [key] ${ }^{257}$ |

## Creating Your Own Menus

ASN (USER new_menu_name ENTERT will define a new user menu. In this sequence, ASN USER turns on AIM so you can immediately enter the new menu name (following the rules of p. 59).

## Example:

To create a menu Favfun for your favorite functions, enter: ASN USER (F) $\nabla$ A $V(F)(U) \mathbb{N}$ ENTERT

The new name will be inserted in CATALOG'MENUS (note ASSIGN will throw an error if the menu name specified turns out being defined already). The new menu itself will be created with 18 blank entries - its

[^135]size is fixed, no submenus are allowed in it. ${ }^{258}$ You may fill it now.

## Example:

Assign the $y$-forecasting function to the $4^{\text {th }}$ key in that new user menu (assuming you did not define any other menu starting with 'Fa' before).

Also the solution of this example will be shown step by step: It starts with the last display of last paragraph since MyMenu stays on screen as long as no other menu is called, and we assigned one function to it just above.


[^136]| DATES | DISP | EQN | EXP | Expon: | EXPT |
| :---: | :---: | :---: | :---: | :---: | :---: |
| CHARS | CLK | CLR | CNST | CPX | CPXS |
| ADV | ANGLES | A: | Binom: | BIT | Cauch: |

Here you see the $1^{\text {st }}$ view on all the menus defined on your WP43. Now, enter (F) and the view jumps to the corresponding position in this submenu:

| ASSIGN $\hat{y}$ : |  |  | 0. |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| I/0 | LgNrm: | Logis: | LOOP | L.INTS | MATRS |
| $\mathrm{f}^{\prime \prime}$ | F\&p | Geom: | Hyper: | INFO | INTS |
| Favfun | FCNS | FIN | FLAG | F: | f' |

Press F1 now
Favfun ASSIGN $\hat{y}$ :


Since Favfun was just created above there is nothing to be seen in the menu section of the display yet. Pressing (F4, however, you will get


Summarizing, ASN STAT $\boldsymbol{\Delta} \hat{y}$ CATALOG MENUS Favfun F4 did the job here. Note that Favfun will remain on screen until another menu or USER is called or it is EXITed explicitly.

## Purging User-Defined Items

Also user-defined items (menus, variables, or programs) are contained in CATALOG. Needing space, you can delete such items via DELITM easily. DELITM lives in CLR. Calling it will show this menu:

## PROGS VARS MENUS

## CLR DELITM PROGS <br> $\square$ allows for deleting the program (starting with the global label) selected (cf. CLP). Predefined labels cannot be deleted. <br> CLR DELITM VARS ... $\square$ allows for deleting the variable selected. Predefined variables cannot be deleted. <br> CLR DELITM MENUS ... $\square$ allows for deleting the menu selected. Predefined menus cannot be deleted.

## Assigning Your Favorite Characters

You must be in alpha input mode (AIM) to do the following. Then, ASN character [key] will assign the character specified to [key]. You can pick the character to be assigned from the alpha keyboard or an arbitrary alpha menu as introduced above (on p. 204). [key] may be any legal label location, shifted or unshifted, except (USER), ENTER $\uparrow$, or EXIT. The assignment will become valid when AIM is called in user mode or when user mode is called in AIM.

## Example:

Assign the Yuan or Yen symbol $¥$ (contained in $\underline{\alpha \bullet) \text { to the } 1^{\text {st }} \text { key in Myd }}$ (assume startup default settings once again):



Note that USER will exit all menus being open at that time so Mya can slip on the screen (being empty still).

F1


Summarizing, ASN $\square \square$ USER F1 did this assignment.

## Launching Your Personal User Interface (User Mode)

USER toggles user mode; therein, your (user) assignments become valid wherever they apply. Compare previous chapters. User mode gives you unexcelled freedom for creating your
 personal calculator user interface. Enjoy and play with the opportunities you have: you can reassign everything except , $\square, \infty$, and F1 - F6. We recommend leaving also ENTERT, EXIT / ON, ${ }^{259}$ CATALOG, and OFF untouched. And do not forget you will need the functionality of USER for returning from user mode.

WARNING: Do not remove inevitably necessary functionality from the user keyboard by overwriting it using ASN (but you may move it).

[^137]Example:
We recommend assigning CATALOG to another location before assigning anything to -shifted EXIT else you will be unable to access CATALOG in user mode anymore.

In case of emergency, a hard reset (using the RESET hole) may be your only escape saving you from user mode, but erasing all your precious programs, data, and settings in RAM. Only what you saved in FM (via SAVE or SAVEST, and the
 assignments are strictly at your own risk.

Pressing any function key (or $\square$ or plus a key) displays a preview of the operation currently assigned to this place, left in the T numeric row if you realize you have pressed the wrong key, simply keep it down until the display falls back to NOP after 1 second. This preview and fallback applies to all key functionalities except $0 \ldots 9, \square,[\mathrm{E}$, ( $\dagger$ ) in numeric entry,) © , and F1 to F6. ${ }^{260}$ Pressing EXIT will fall back to EXITALL exiting all menus (consult the $I O I$ ). $\square$ and are echoed but will not fall back. Preview and fallback are particularly helpful in user mode, when the function called by a particular key may not be indicated on the keyboard.

[^138]

Once you have reached a stable user layout, store it (using STOCFG) in a register or variable, together with the other settings mentioned on p. 85. Printing keyboard overlays for your favorite layouts may pay well, especially if you reassign just functions appearing on the bezel (so you can cover them all by one overlay). Overlays cover this bezel entirely and are fixed in 6 slots provided in the keyboard rim (see the ReM, App. F, for the dimensions).

STOCFG and overlays are especially beneficial if you plan having further alternative layouts - you can load any of them one by one via RCLCFG. ${ }^{261}$ Think of creating and storing a dedicated configuration for working with short integers bringing Boole's operations on the bezel, for example.

If, however, you should get lost in your various user assignments and configurations, look for $\mathbf{U}$ top right in the status bar - and remember that pressing USER (or wherever you have put this functionality) will return immediately to the factory default keyboard of your WP43 as you know it from the very beginning. And if you want to get rid of an outdated user layout and free the memory allocated for the respective assignments, simply clear the register where this configuration was stored or delete the respective variable as described on p. 306.

We sign off wishing you long lasting joy and benefit working with your very own, personalized WP43!

[^139]
## APPENDIX 1: KEY RESPONSE TABLE

Here you find all direct keystroke inputs explained, top left to bottom right of the keyboard. For each key, its unshifted function is mentioned first, then its - and its $\square$-shifted function, if applicable.

Most keys will change functionality in alpha input mode (AIM, cf. pp. 203ff), hence the "alpha" meanings are listed thereafter with labels printed in darker fields and descriptions on light grey. Remember AIM is also used for entry of equations.

See the pages mentioned explicitly in the table or the ReM for details of all the functions mentioned below.

| R | Keystrokes | Meaning |  |
| :---: | :---: | :---: | :---: |
| 1 | $\square$ | Such an unshifted softkey (actually F1, F2, F3, F4, F5), or F6) calls the item displayed in the bottom menu row of the display ... | ... in the corresponding column 1 to 6 . Executes what is called and allow- |
|  | $\square$ $\square \square$ | A shifted softkey (see above) calls the item displayed in the golden or blue menu row ... | no item is displayed at this position (cf. pp. 28f). |


| 2 | $1 / \mathrm{x}$ | Inverts the number $\boldsymbol{x}$ (cf. p. 33) or all elements of the matrix $\boldsymbol{x}$ (see (MATX for matrix inversion). |
| :---: | :---: | :---: |
|  | $a b / c$ | Enters fraction display mode (FDM), i.e. displays all reals as proper fractions or mixed numbers. If FDM was active already, toggles between proper and improper fractions (cf. pp. 136ff). |
|  | U $\rightarrow$ | Opens the menu of unit conversions (cf. pp. 292ff). |
|  | A | Enters the Latin letter $\mathbf{A}$ or a |
|  | A | Opens the catalog of all Latin letters provided, also accented ones. |
|  | (A) | Enters the Greek letter A or $\boldsymbol{\alpha}$ |


| R | Keystrokes | Meaning |
| :---: | :---: | :---: |
| 2 | EXP | Opens a menu containing $\boldsymbol{y}^{\boldsymbol{x}}, \boldsymbol{x}^{2}, \boldsymbol{x}^{3}$, roots, logarithms, hyperbolic and some exponential functions more (cf. p. 28). |
|  | \# | If pressed trailing integer input, defines its base. Else converts a closed $\boldsymbol{x}$ into a short integer of the base specified (cf. pp. 150ff) or filters the integer or fractional part of $\boldsymbol{x}$. |
|  | BIT | Opens a menu containing Boole's operations (AND, OR, NOT, etc.) as well as bit manipulating commands. |
|  | B | Enters the Latin letter B or b |
|  |  | Enters the character \# |
|  |  | Enters the Greek letter B or $\boldsymbol{\beta}$ |
| 2 | TRI | Opens the menu containing trigonometric and hyperbolic functions and their inverses (cf. p. 30).. |
|  | d.ms | If pressed trailing numeric input, enters an angle in degrees, minutes, and seconds (i.e. sexagesimal notation). Else sets angular display mode to sexagesimal angles (cf. pp. 136ff). |
|  | [x] | Opens the menu of angular conversions (cf. p. 140). |
|  | C | Enters the Latin letter $\mathbf{C}$ or $\mathbf{c}$ |
|  |  | Enters an opening parenthesis |
|  | $\square$ | Enters the Greek letter $\Gamma$ or $\gamma$ |
| 2 | In | Returns the natural logarithm of $\boldsymbol{x}$ (cf. pp. 33 and 92). ${ }^{262}$ |
|  | .. | If pressed trailing numeric input, enters a date (cf. p. 197). Else leaves fraction display mode (see ab/c above) and converts <br> - an integer to a real number (cf. p. 149), <br> - a sexagesimal angle to a decimal number (cf. p. 142), <br> - a sexagesimal time to a decimal number (cf. p. 201). |
|  | 19 | Returns the (common) decadic logarithm of $\boldsymbol{x}$ (cf. (ln). ${ }^{262}$ |

[^140]

| 3 | STO | Stores（copies） $\boldsymbol{x}$ in the destination specified（cf．pp．56ff）． |
| :---: | :---: | :--- |
| ASN | Assigns an item to a key，allowing you to create your very <br> personal user keyboard layout（cf．pp．299ff）． |  |
| SAVE | Saves all your data in a backup file（cf．p．246）in FM from <br> where they may be recovered by LOAD entirely． |  |
| G | Enters the Latin letter $\mathbf{G}$ or $\mathbf{g}$ |  |
| $G ⿴ 囗 十 丌$ | Enters the Greek letter $\Gamma$ or $\boldsymbol{\gamma}$（gamma） |  |


| R Keystrokes |  | Meaning |
| :---: | :---: | :---: |
| 3 | RCL | Recalls (copies) a stored object into $\mathbf{X}$ (cf. pp. 56ff). - If pressed in RBR, leaves RBR after recalling the object at the bottom line or entering a corresponding step (cf. pp. 277f). |
|  | RBR | Calls the register browser (cf. pp. 277f). |
|  | VIEW | Views the destination, i.e. displays its address and contents directly below the status bar until next keystroke (cf. p. 62). |
|  | H | Enters the Latin letter $\mathbf{H}$ or $\mathbf{h}$ |
|  | H | Enters the Greek letter $\mathbf{X}$ or $\chi$ (chi) |


| 3 | R $\downarrow$ | Rolls the stack contents one level downRolls the stack contents one level up $\quad$ (cf. p. 42). |
| :---: | :---: | :---: |
|  | RT |  |
|  | CPX | Opens the menu of commands operating on complex numbers like CONJ, CROSS, DOT, and Re₹Im (cf. pp. 164ff). |
|  | (1) | Enters the Latin letter I or $\mathbf{i}$ |
|  | $\underline{1}$ | Makes next character a subscript (if applicable) |
|  | (1) | Enters the Greek letter I or $\mathbf{l}$ (iota) |


| 3 | CC | Complex closing, composing, cutting, and converting (see pp. 164ff and 323). |
| :---: | :---: | :---: |
|  | \|x| | Returns the absolute (unsigned) value of $\boldsymbol{x}$ (cf. p. 33) or the magnitude of $\boldsymbol{x}$. ${ }^{262}$ |
|  | 【 | Returns the phase of $\boldsymbol{x}$. ${ }^{262}$ |
|  | J | Enters the Latin letter J or $\mathbf{j}$ |
|  | \|x| | Enters $\left.\right\|_{*} \mid$ in an equation or $\mid$ in an alphanumeric string |
|  | [7] | Enters the Greek letter H or $\boldsymbol{\eta}$ (eta) |


| R Keystrokes Meaning <br> 3 $\square$ Prefix to reach a golden function label. Pressing <br> will cancel it. <br>  SNAP Dumps the current screen to a file on the calculator's FAT <br> disk, i.e. takes a snapshot or screenshot. See the IOI. <br> 3 $\square$ Prefix to reach a blue function label. Pressing <br> cancel it. |
| :--- |


| 4 | ENTER | Context sensitive key, see p. 323. |  |
| :--- | :--- | :--- | :--- |
|  | FIIL | Fills all stack registers with $\boldsymbol{x}$ | (cf. p. 42). |
| DROP |  |  |  |
|  | Drops $\boldsymbol{x}$ from the stack |  |  |


| 4 | $x^{2} \geqslant y$ | Swaps the contents of $\mathbf{X}$ and $\mathbf{Y}$ (cf. p. 42). |
| :---: | :---: | :---: |
|  | x ${ }^{\text {a }}$ | Swaps the contents of $\mathbf{X}$ and the destination specified. |
|  | STK | Opens the menu of stack related operations (i.e. drop, swap, and shuffle commands, cf. pp. 40ff). |
|  | K | Enters the Latin letter $\mathbf{K}$ or $\mathbf{k}$ |
|  | $x \geqslant y$ | Enters the character * |
|  | K | Enters the Greek letter K or $\boldsymbol{\kappa}$ (kappa) |


| 4 | + | If pressed during input of mantissa or exponent, changes its sign (cf. p. 25). Else multiplies $\boldsymbol{x}$ times -1 . |
| :---: | :---: | :---: |
|  | $\triangle \%$ | Returns $100{ }^{(\boldsymbol{x}-\boldsymbol{y})} / \boldsymbol{y}$. Leaves $\boldsymbol{y}$ unchanged. Cf. pp. 91f. |
|  | FIN | Opens the menu of financial functions (i.e. various \% functions and TVM - see pp. 282ff and 326ff). |



| 5 | 1 | If there is a pending question like are you sure?, enters $\mathbf{N}$ for 'no'. <br> Else divides $\boldsymbol{y}$ by $\boldsymbol{x}$. For matrices, multiplies $\boldsymbol{y}$ times $\boldsymbol{x}^{-1}$. |
| :---: | :---: | :---: |
|  | (11) | Returns $(1 / x+1 / y)^{-1}$ |
|  | MOD | Returns $\boldsymbol{y}$ modulo $\boldsymbol{x}$. Cf. pp. 157f. |
|  | N | Enters the Latin letter $\mathbf{N}$ or $\mathbf{n}$ |
|  | $\square$ | Enters the character / |
|  |  | Enters the Greek letter N or $\boldsymbol{v}$ ( nu ) |


| R | Keystrokes | Meaning |
| :---: | :---: | :---: |
| 5 | 7 | Enters the digit 7. |
|  | 0 | Enters the Latin letter $\mathbf{O}$ or $\mathbf{0}$ (showing the same shape as the Greek letter o-mikron). |
|  | $\square$ | Enters the character 7 |
|  | 0 | Enters the Greek letter $\boldsymbol{\Omega}$ or $\omega$ (o-mega) |
| 5 | 8 | Enters the digit 8. |
|  | MODE | Opens a menu of operations for setting angular display or rounding modes and maximum denominator (cf. pp. 136ff). |
|  | (P) | Enters the Latin letter $\mathbf{P}$ or $\mathbf{p}$ |
|  | 8 | Enters the character 8 |
|  | $\square$ | Enters the Greek letter $\Pi$ or $\pi$ (pi) |
| 5 | 9 | Enters the digit 9. |
|  | LBL | Enters a label for a particular location in program memory. |
|  | RTN | Returns to the caller (cf. pp. 212ff). |
|  | Q | Enters the Latin letter $\mathbf{Q}$ or $\mathbf{q}$ |
|  | 0 | Enters the character 9 |
|  | - | Works like 0 above. |


| 5 | XEQ | If there is an open question like Are you sure?, confirms it; <br> else - if in PEM - inserts a call to the subroutine with the <br> label specified; <br> else (i.e. in $R U M$ ) calls the routine with the label specified <br> and starts executing it (cf. pp. 212ff). |
| :--- | :--- | :--- |
| GTO | Goes to the specified location in program memory. |  |
| FLAG | Opens the menu of flag commands. These are of most use <br> in PEM (cf. pp. 212ff). |  |


| R | Keystrokes | Meaning |
| :---: | :---: | :---: |
| 6 | X | Multiplies $\boldsymbol{y}$ times $\boldsymbol{x}$. |
|  | x! | Returns the factorial of $\boldsymbol{x}$ (or $\Gamma(\boldsymbol{x}+1)$ for non-integer $\boldsymbol{x}$ ). Cf. pp. 19, 33, and 102. |
|  | PROB | Opens a menu containing combinations, permutations, the Gamma function, a random number generator, and all the probability distributions supported (cf. pp. 102ff). |
|  | R | Enters the Latin letter $\mathbf{R}$ or $\mathbf{r}$ |
|  | $\square$ | Enters the character $\mathbf{x}$ or - |
|  |  | Enters the Greek letter P or $\boldsymbol{\rho}$ (rho) |


| 6 | 4 | Enters the digit 4. |
| :---: | :---: | :---: |
|  | ( | Opens the menu of accumulated statistical sums (cf. p. 128). |
|  | STAI | Opens the menu of operations for $1 D$ and $2 D$ sample statistics: $\Sigma+, \Sigma-$, CL $\Sigma$, various means and measures for scattering, as well as functions for histograms, curve fitting, and measuring system analysis (cf. pp. 105ff). |
|  | S | Enters the Latin letter $\mathbf{S}$ or $\mathbf{s}$ |
|  | 4 | Enters the character 4 |
|  |  | Enters the Greek letter $\Sigma$ or $\boldsymbol{\sigma}$ (sigma) |


| 6 | 5 | Enters the digit 5. |
| :---: | :---: | :---: |
|  | R+ | Calls $\rightarrow$ REC, converting polar coordinates $\boldsymbol{r}$ (in $\mathbf{X}$ ) and $\boldsymbol{\vartheta}$ (in $\mathbf{Y}$ ) to rectangular coordinates $\boldsymbol{x}$ and $\boldsymbol{y}$ (cf. pp. 142ff). |
|  | $\rightarrow \mathrm{P}$ | Calls $\rightarrow \mathrm{POL}$, converting rectangular coordinates $\boldsymbol{x}$ and $\boldsymbol{y}$ to polar coordinates $\boldsymbol{r}$ (in X) and $\boldsymbol{\vartheta}$ (in $\mathbf{Y}$ ). Cf. pp. 20f and 142ff. |
|  | T | Enters the Latin letter $\mathbf{T}$ or $\mathbf{t}$ |
|  | $\square 5$ | Enters the character 5 |
|  | T | Enters the Greek letter T or $\tau$ (tau) |


| R | Keystrokes | Meaning |
| :---: | :---: | :---: |
| 6 | 6 | Enters the digit 6. |
|  | 0 | Calls the timer application (cf. pp. 280ff). |
|  | CLK | Opens the menu of time and date commands (cf. pp. 197ff). |
|  | U | Enters the Latin letter $\mathbf{U}$ or $\mathbf{u}$ |
|  | 6 | Enters the character 6 |
|  | [9] | Enters the Greek letter $\Theta$ or $\vartheta$ (theta) |
| 6 | ( | Context sensitive key, see p. 325. |
|  | 辰 $\triangle$ | Moves the program pointer one step back (cf. pp. 212ff). Will repeat with 2 Hz when pressed longer than $0.5 s$. |
|  | SE | Sets the flag specified. |


| 7 | - | Subtracts $\boldsymbol{x}$ from $\boldsymbol{y}$. |
| :---: | :---: | :---: |
|  | INTS | Opens a menu providing the digits $\mathbf{A} \ldots \mathbf{F}$, operations on integers, and integer sign mode settings. This menu is most useful with short integers (cf. pp. 150, 154ff, and also BIT). |
|  | PARI | Opens a menu containing FP, IP, SIGN, DECOMP, etc. |
|  | V | Enters the Latin letter $\mathbf{V}$ or $\mathbf{v}$ |
|  | $\square$ | Enters a minus sign |
|  | [ $]$ | Opens a menu of mathematical symbols |


| 7 | $\mathbf{1}$ | Enters the digit 1. |
| :--- | ---: | :--- |
|  | ADV | Opens a menu of advanced operations for solving arbitrary <br> programmed equations, finding roots, integrating, <br> differentiating, computing sums and products (cf. pp. 252ff). |
| EQN | Opens the menu of all equations currently defined (cf. pp. <br> 255ff). Allows for editing and interactive solving etc. |  |



| 7 | 3 | If there is a pending question like Are you sure?, enters $\mathbf{Y}$ <br> for 'yes'. <br> Else enters the digit 3. |
| :---: | :---: | :--- |
| $\pi$ |  | Recalls the first 34 digits of the number $\boldsymbol{\pi}$ into $\mathbf{X}$. |
| CONST | Opens a catalog of fundamental physical, astronomical, <br> mathematical, and surveying constants (cf. pp. 287ff). |  |
| $\mathbf{Y}$ | Enters the Latin letter $\mathbf{Y}$ or $\mathbf{y}$ |  |
| $\square \mathbf{3}$ | Enters the character $\mathbf{3}$ |  |
|  | Enters the Greek letter $\mathbf{Y}$ or $\mathbf{v}$ (y-psilon) |  |


| 7 | $\nabla$ | Context sensitive key, see p. 325. |
| :---: | :---: | :---: |
|  | 틈 | Moves the program pointer one step forward (cf. pp. 212ff). Will repeat with 2 Hz when pressed longer than 0.5 s . |
|  | (CF) | Clears the flag specified. |

## R Keystrokes <br> Meaning

| 8 | $\pm$ | Adds $\boldsymbol{x}$ to $\boldsymbol{y}$. |
| :---: | :---: | :---: |
|  | LOOP | Opens a menu containing INC, DEC, and the related loop control commands ISG, DSE, etc. (cf. pp. 228f). |
|  | TESI | Opens the menu of binary tests (cf. pp. 224ff). |
|  | [ | Enters the Latin letter $\mathbf{Z}$ or $\mathbf{z}$ |
|  | $\pm$ | Enters the character + |
|  | Z | Enters the Greek letter $\mathbf{Z}$ or $\zeta$ (zeta) |


| 8 | 0 | Enters the digit $\mathbf{0}$. |
| :---: | :---: | :---: |
|  | I/0 | Opens the menu of I/O-related operations (cf. pp. 246f). |
|  | (凹) | Opens the menu of print-related operations. |
|  | ? | Enters a question mark. |
|  | 0 | Enters the character $\mathbf{0}$. |
|  | [凹] | Enters the printer character. |


| 8 | $\square$ | Usually enters a decimal radix mark (cf. p. 25). If pressed <br> twice in numeric input, allows for entering a fraction directly <br> (cf. pp. 72f and 136ff). In register or flag addressing, <br> heads a local address (cf. pp. 60ff). |
| :--- | :--- | :--- |
| SHOW | Shows the number or string $x$ <br> precision until next keystroke. |  |
| $\square$ | Opens a menu of commands returning system information <br> (cf. p. 227). These operations are most useful in PEM (cf. pp. <br> 212ff). |  |
| $\square$ | Enters a comma. |  |
| $\square$ | Enters a point. |  |

## |R|Keystrokes Meaning

| 8 | R/S | Context sensitive key, see p. 324. |
| :---: | :---: | :---: |
|  | P/R | Toggles program-entry and run mode (cf. pp. 22 and 212ff). |
|  | P.FN | Opens a menu of dedicated programming functions, being of most use in PEM (cf. pp. 212ff). |
|  | $\square$ | Enters a blank space character. |


| 8 | EXIT | I ON : Context sensitive key, see p. 324. |
| :---: | :---: | :--- |
| CATALOG | Opens the catalog of everything (functions, variables, menus, <br> programs, etc.). See ReMS 2 for its structure and contents. |  |
| OFF | If pressed in PEM, inserts the command OFF behind the <br> current step (cf. p. 213). <br> Else turns your WP43 off. |  |

## Context Sensitive Keys

Seven context sensitive keys need longer explanations - find them in the table starting overleaf, sorted alphabetically. If any of these keys is pressed, your WP43 will run top down through a sequence of key-specific tests - for the first test becoming true, your WP43 will execute the corresponding operation and return, waiting for your next input.

| Key | Condition(s) | Meaning |
| :---: | :---: | :---: |
| CC | $\mathbf{X}$ contains an open (input) number, cf. p. 25 | For POLAR, CC closes input, checks, and saves it as real part of a forthcoming complex number, then waits for your input of its imaginary part. <br> Else cCloses input, checks, and saves it as magnitude, and waits for your phase input. Cf. pp. 164ff. |
|  | $\mathbf{X}$ contains a closed complex number, vector, or matrix | For POLAR, (CC splits ('cuts') $\boldsymbol{x}$ into its real and imaginary part, returning the real part in $\mathbf{Y}$ and the imaginary part in $\mathbf{X}$. <br> Else (CC splits $\boldsymbol{x}$ into its magnitude $\boldsymbol{r}$ and phase $\boldsymbol{\vartheta}$, returning $\boldsymbol{r}$ in $\mathbf{Y}$ and $\boldsymbol{\vartheta}$ in $\mathbf{X}$. |
|  | $\mathbf{X}$ and $\mathbf{Y}$ contain two closed reals | Interprets $\boldsymbol{y}$ and $\boldsymbol{x}$ either (for POLAR set) as magnitude and phase, or (for POLAR) as real and imaginary parts. CC combines $\boldsymbol{y}$ and $\boldsymbol{x}$ like a dyadic function composing one complex number $\boldsymbol{x}$. |
|  | $\mathbf{X}$ and $\mathbf{Y}$ contain two closed real matrices of identical dimension | Returns one complex matrix $\boldsymbol{x}$, working in analogy to previous row. |
|  | Else | Throws an error. |


| ENTERT | Waiting for parameter input | Closes pending command input and executes said command (cf. p. 66). |
| :---: | :---: | :---: |
|  | Asking for confirmation | Confirms the question. |
|  | In TIMER | Is honored as described on pp. 280f. |
|  | In numeric input or AIM | Closes input and pushes it on the stack (cf. pp. $36 f$ and 42) or in the pending program step. |
|  | In PEM | Puts ENTER4 in the pending program step. |
|  | Else | Does nothing. |


| Key | Condition(s) | Meaning |
| :---: | :---: | :---: |
| $\frac{\text { EXIT } /}{\mathrm{ON}}$ | WP43 turned off | Turns your WP43 on. |
|  | Waiting for numeric, alphanumeric, or parameter input | If an input menu is open, closes this menu. Else if waiting for parameter input, cancels the pending command. Else closes input. |
|  | SHOW | Erases output of SHOW |
|  | Temporary information displayed | Clears this information (e.g. an error message), returning to the calculator state as was before it was thrown (cf. p. 71). |
|  | Asking f. confirm. | Denies the question. |
|  | In RBR, STATUS, TIMER | Leaves the application (cf. pp. 277ff). |
|  | In a sub-menu | Leaves the current sub-menu returning to its parent menu. |
|  | In a menu or browser | Leaves the current menu or browser without executing anything, returning to the status of your WP43 as it was before. |
|  |  | Stops executing the running program immediately. 9 will be lit until next keystroke. |
|  | In AIM | Closes $\boldsymbol{x}$ and leaves alpha input mode. |
|  | In PEM | Leaves program-entry mode like $\mathbb{P} / R$. |
|  | Else | Does nothing. |
| $\mathrm{R} / \mathrm{S}$ | In TIMER | Starts or stops the timer without changing its value (cf. pp. 280f). |
|  |  | Stops executing the running program immediately. $\Theta$ will be lit until next keystroke. |
|  | In AIM | Inserts a blank character. |
|  | In PEM | Enters STOP in the current program step. |
|  | Else | Runs the current routine (cf. pp. 212ff) or resumes its execution starting with the step after the current step. |


| Key | Condition(s) | Meaning |
| :---: | :---: | :---: |
|  | After STO or RCL | Honored as described on pp. 61ff. |
|  | In RBR, STATUS, TIMER | Honored as described on pp. 277ff. |
|  | In ASSESS | Honored as described in the 101. |
|  | In ASSIGN | Honored as described on pp. 300ff. |
|  | A \& in ( $\underline{\text { a }}$ NTL, A | A... $\overline{\text { ) }}$ / sets lower case. |
|  | $\alpha$ \& in ( $\underline{\alpha / N T L}$, A | . $) \triangle$ sets upper case. |
|  | A else | $\nabla$ sets lower case and continues testing |
|  | $\alpha$ else | $\triangle$ sets upper case and |
|  | In EQN | (4) goes to next or ... to previous equation, if applicable. |
|  | In a multi-view menu | (4) goes to next or ... to previous view in the current menu. |
|  | In PEM | (4) goes to previous or ... to next program step. Will repeat with 2 Hz when pressed longer than 0.5 s . |
|  | In RUM | Browses the current routine with... going to previous program step or ... executing the current program step and going to next step. |


| 4 | Open command parameter input | Deletes the last character entered. If none command like EXIT is left, $\boldsymbol{\uplus}$ cancels the pending ... input (cf. p. 25). <br> Clears the information returning to the calculator state as before this information (e.g. an error message) was thrown (cf. p. 71). <br> Denies the question. <br> Resets the timer (cf. pp. 280f). <br> Deletes the current program step. <br> Calls the command CLX. |
| :---: | :---: | :---: |
|  | Open (alpha-) numeric input |  |
|  | Temporary information displayed |  |
|  | Asking for confirmation |  |
|  | In TIMER |  |
|  | In PEM |  |
|  | Else |  |

## Virtual Keyboards



Virtual keyboard
in TAM for the commands $\#$ and $\rightarrow$ INT (cf. pp. 149f)


Virtual keyboard in TAM for addressing labels, i.e. for GTO, LBL, XEQ, etc. (cf. pp. 218ff)


Virtual keyboard in AIM (cf. pp. 203ff, especially for explanation of the three alpha menus and Greek letters supported by your WP43, and 255ff)

There are further
virtual keyboards effective in FBR and RBR (cf. pp. 277f), STATUS (cf. pp. 279ff), and TIMER (cf. pp. 280ff).


## APPENDIX 2: OPERATOR PRECEDENCE

Your WP43 does not have to care for operator precedence since it always executes just one operation at a time (cf. p. 49). Hence it is your job to control the sequence of operations you present to your WP43. There are common rules and conventions in mathematics dealing with that - you have learned them in school. Here is just one example for affirmation and/or reminding:

$$
1-2 \cdot 3^{4}: 5+\sin 2\left(6-\sqrt[3]{7^{2}}\right) \cdot 8!+\ln \left(-9^{2^{3}} \cdot 45^{(6 / 7)}\right)^{2}
$$

or, written for another part of this world needing more space:

$$
1-2 \times 3^{4} \div 5+\sin 2\left(6-\sqrt[3]{7^{2}}\right) \times 8!+\ln \left(-9^{2^{3}} \times 45^{(6 / 7)}\right)^{2}
$$

This may be solved the following way, for instance, using your WP43 with startup default settings:

|  | returns $-9^{2^{3}}$; note the arguments automatically fill in correctly. |
| :---: | :---: |
| 6 ENTERT 7 (1) 45 x $x^{2} \mathrm{y}$ ( $y^{x}$ | calculates $45(6 / 7)$. |
| $x$ x 10 | solves the rightmost ( $4^{\text {th }}$ ) term. |
| 7 ( $x^{2}[\sqrt[3]{x}] 6 x^{2} \geqslant y-2 x \sin$ | solves the sine. |
| $8 \times$ x $x$ | solves the $3^{\text {rd }}$ term. |
| $\pm$ | adds it to the $4^{\text {th }}$. |
| 3 ENTERT 4 [ $y^{x} 2 \times 5$ (1) | solves the $2^{\text {nd }}$ term. |
| $\square$ | subtracts it from said sum. |

$1 \oplus$ returns the overall solution

### 3300.988666858076

The colors indicate the three stack registers employed for this solution (cf. pp. 40ff). Note $x \geqslant y$ is used twice here to swap arguments.

## APPENDIX 3: FURTHER APPLICATIONS OF TVM

First, let's use TVM for a very old example, before pocket calculators and cash-flows were even invented:

The inventor of chess was asked by his king for a wish as reward. He is said to have requested one grain of rice on the $1^{\text {st }}$ field, two grains on the second, four on the third, eight on the fourth, etc. until the $64^{\text {th }}$ field. The king was reported to have laughed about that modest wish. How much rice had the king to lay out in total after all?

## Solution:


$64 \mathrm{n}_{\text {PER }}$ since grains are laid out at the end of each field,

| $\mathbf{1}$ | PV | 1.00 | grain in the $1^{\text {st }}$ field, |
| :--- | :--- | ---: | :--- |
| $\mathbf{6 4}$ | nefr $_{\text {PER }}$ | 64.00 | fields in total |

Nowadays, one grain of rice weighs about 65 mg . At the time of this legend, rice was smaller and lighter, let's assume 20 mg per grain. So

$$
18.4 \times 10^{18} \times 20 \times 10^{-6} \mathrm{~kg} \approx 370 \times 10^{12} \mathrm{~kg}=370 \mathrm{Gt},
$$

meaning enough for the inventor, his wife, kids, and grandkids including their families and villages for all their lifes by far (the total amount is a major multiple of the global rice production of today).

It is said the king never again asked an inventor for a wish. Other sources report he forced the inventor to count his reward. ${ }^{263}$

Consonant with the definition on p. 282, throughout TVM pictures, amounts received are represented by arrows pointing up, money laid out (paid, invested) by arrows pointing down. Various types of financial

[^141]problems can be sketched like this then: ${ }^{264}$

## Generalized Net Cash Flow Diagrams and Terminology

(Note that diagrams involving payments may be represented with payments at the beginning or end of the period.)


All pictures as well as the text printed in blue throughout this appendix are quoted from the $H P-27 \mathrm{OH}$. All calculations are executed in FIX 2. Enjoy the boundary conditions of that time gone for long - those were the days...

[^142]
## Ordinary Annuities (a.k.a. Payments in Arrears)

An annuity is a series of equal payments made at regular intervals. The time between annuity payments is called the payment interval or payment period. If your payment is due at the end of each payment period, it's called an ordinary annuity or payment in arrears. Examples of ordinary annuities are a car loan (where you drive away now and pay later) or a mortgage (where the payments start one month after you get your loan).

The time/money relationship for an ordinary annuity with monthly payments for a year would look like this $\rightarrow{ }^{265}$


## Example for find-

 ing the number of periods for an ordinary annuity:Through an insurance fund, you have accumulated $\$ 50000$ for your
 retirement. How long can you withdraw $\$ 3000$ every 6 months (starting 6 months from now) if the fund earns $5 \%$ per annum compounded semiannually?

## Solution:

## FIN TVM End

-50000 PV
3000 PMT
5 ENTERT 2 (1) $\% /$ a
1 per/a
$n_{\text {PER }}$
$\qquad$

Example 1 for finding the interest rate for an ordinary annuity:
What is the annual interest rate (a.k.a. APR for annual percentage rate) on a 2-year, $\$ 1775$ loan with $\$ 83.65$ monthly payments? (Continue with the settings of previous example):

Solution:

| 12 per/a | 12.00 months per year, |
| :---: | :---: |
| $2 \times \mathrm{n}_{\text {PER }}$ | 24.00 periods in total, |
| -1775 PV | -1775.00 principal (capital), |
| 83.65 PMT | 83.65 payment; |
| i\%/a | 12.11 \% APR. |

Borrowers are sometimes charged fees related to the issuance of a mortgage, which effectively raises the interest rate. Given the basis of the fee charge, the true annual percentage rate may be calculated.

Example 2 for finding the interest rate for an ordinary annuity:
A borrower is charged 2 points for the issuance of his mortgage. If the mortgage amount is $\$ 50000$ for 30 years, and the interest rate is $9 \%$ per year, with monthly payments, what annual percentage rate is the borrower paying? ( 1 point is equal to $1 \%$ of the mortgage amount.) (Continue with $F V=0 \ldots$ )

## Solution:

First, compute the payment amount which is based on $\$ 50000$

| 9 i\%/a | 9.00 annual interest rate, |
| :---: | :---: |
| 12 per/a | 12.00 months per year, |
| $30 \times \mathrm{n}_{\text {PER }}$ | 360.00 periods in total, |
| 50000 PV | 50000.00 principal (capital); |
| PMT | -402.31 payment (will be reused). |
| PV . $98 \times \mathrm{PV}$ | 49000.00 effective amount received, |
| i\%/a | 9.23 \% effective APR. |

What's really happening? For a mortgage with fees, the borrower is making payments on the original loan amount, which corresponds with the initial calculation of the payment amount. If you borrow $\$ 10000$, but are
immediately charged $\$ 500$ in fees, you really only receive $\$ 9500$. But, your payments are based on $\$ 10000$. With fees, then, you're really paying the same for less money, which generates the need to compute the true $A P R$.

## Example for finding the payment amount for an ordinary annuity:

Find the monthly payment amount on a 30-year, \$52000 mortgage at 9.75\% annual interest rate. (Continue with the settings of previous example, so per/a and $\boldsymbol{n}_{\text {per }}$ are specified correctly already.)

## Solution:

52000 PV
9.75 i\%/a

PMT
52000.00 mortgage,
9.75 \% annual interest rate; -446.76 monthly payment.

A common financial occurrence is an annuity that has a large payment at the end. The last payment - usually considerably larger although it could also be smaller than the others - is called a balloon payment or balloon.
By subtracting the present value of the balloon payment from the loan amount, the problem effectively becomes "What is the monthly payment on a direct reduction loan?"

Example (finding the payment for an ordinary annuity with balloon):
Yellowstone Sam is heading north, and will invest in an $\$ 8000$ dog sled and team. His loan specifies 60 monthly payments at $10 \%$ with a balloon payment in the $60^{\text {th }}$ month of $\$ 3000$. What will his monthly payments be?

## Solution:

| 12 per/a | 12.00 months per year, |
| :---: | :---: |
| $60 \mathrm{n}_{\text {PER }}$ | 60.00 payment periods in total, |
| 10 i\%/a | 10.00 \% annual interest rate; |
| -3000 FV | -3000.00 future value of balloon, |
| 0 PMT | 0.00 no payments first hand, |
| PV | 1823.37 present value of balloon; |
| 8000 | 8000.00 gross value of loan amoun |

$x^{2} \geq y$ PV 6176.63 net present value of loan amount less
0 FV PMT - 131.24 monthly payment.

## Example for finding the present value of an ordinary annuity:

Yellowstone Sam decides to purchase a snowmobile. He plans to pay $\$ 80$ per month for 3 years, and he's willing to pay $10 \%$ annual interest. How much can he afford to pay for the snowmobile? (Continue...)

## Solution:

| 12 per/a | 12.00 months per year, |
| :---: | :---: |
| $3 \times \mathrm{n}_{\text {PER }}$ | 36.00 payment periods in total, |
| 10 i\%/a | 10.00 \% annual interest rate; |
| -80 PMT | -80.00 monthly payment, |
| PV | 2479.30 price he can pay for the s |

With loan calculations, you generally solve for $\boldsymbol{n}, \boldsymbol{i}, \mathbf{P M T}$, or $\mathbf{P V}$. There is another type of ordinary annuity called a "sinking fund", where you make payments at regular intervals into a fund to discharge a debt (for example, to pay off a bond issue at maturity). With sinking fund calculations, you solve for $\boldsymbol{n}, \boldsymbol{i}$, PMT, or $\boldsymbol{F V}$ (how much you will have in the fund at a future date).

Sinking fund payments start at the end of the first period, like so $\rightarrow$


This is different from opening a savings account with a starting deposit today. Savings are annuity due calculations and will be described later in this section.

## Example for finding the future value of an ordinary annuity:

A $\$ 100000$ bond is to be discharged by the sinking fund method. If, starting 6 months from now, you deposit $\$ 3914.75$ twice a year into a sinking fund that pays $5 \%$ compounded semiannually, will you be able to pay off the bond in 10 years?

## Solution:

| 2 per/a | 2.00 halves per year, |
| :---: | :---: |
| $10 \times \mathrm{n}_{\text {PER }}$ | 20.00 payment periods in total, |
| 5 i\%/a | 5.00 \% annual interest rate; |
| 0 PV | 0.00 start value, |
| -3914.75 PMT | 3914.75 semiannual deposit, |
| FV | 100000.95 balance of the fund after 10 years - it will just make it! |

## Annuities Due (a.k.a. Payments in Advance)



With some annuities - like insurance premiums or a lease - the payment is due at the beginning of the month. This is called an annuity due because the payment falls at the beginning of the payment period. Other terms are payments in advance or anticipated payments.

An annuity due with monthly payments for a year - say, a car insurance policy ${ }^{266}$ - looks like this:

Notice that with an annuity due, you have a payment right away at the beginning of the first interval (with an ordinary annuity, your payment is not due until the end of the first


Policy Purchased period, but you also have a payment at the end of the entire term).

The following calculations all deal with annuity due problems, e.g. savings, insurance, leases, and rents.

[^143]
## Example 1 for finding the number of periods for an annuity due:



Example 2 for finding the number of periods for an annuity due:
If you deposit $\$ 50$ a month in a savings account that pays $6 \%$ interest, how long will it take to reach $\$ 1000$ ? (Continue with the settings...)

## Solution:

| $\mathbf{6}$ i\%/a | $6.00 \%$ annual interest rate, |  |
| :--- | ---: | :--- |
| $\mathbf{0}$ PV | 0.00 | start balance, |
| $\mathbf{1 0 0 0}$ FV | 1000.00 future value; |  |
| $\mathbf{- 5 0 ~ P M T ~}$ | -50.00 monthly payments to the bank. |  |
| $\mathbf{n}_{\text {PER }}$ | 19.02 months. |  |

## Example for finding the interest rate for an annuity due:

Equipment worth $\$ 12000$ is leased for 8 years with monthly payments in advance of $\$ 200$. The equipment is assumed to have no salvage value at the end of the lease. What yield rate does this represent?

## Solution:



## Example for finding the payment amount for an annuity due:

The owner of a building presently worth $\$ 70000$ intends to lease it for 20 years at the end of which time he assumes the building will be worthless (i.e., has no residual value). How much must the quarterly payments (in advance) be to achieve a 10\% annual yield?

## Solution:

| 4 per/a | 4.00 quarters per year, |
| :---: | :---: |
| $20 \times n_{\text {PER }}$ | 80.00 payment periods in total, |
| 10 i\%/a | 10.00 \% annual target yield, |
| -70000 PV | -70 000.00 present value of the building; |
| 0 FV PMT | 1982.27 quarterly payments. |

## Example for finding the present value for an annuity due:

The owner of a downtown parking lot has achieved full occupancy and a 7\% annual yield by renting parking spaces for $\$ 40$ per month payable in advance.
 Several regular customers want to rent their spaces on an annual basis. What annual rent, also payable in advance, will maintain a $7 \%$ annual yield rate?

## Solution:

| 12 | per/a | 12.00 |
| :--- | :--- | :--- |
| 12 | $n_{\text {PER }}$ | 12.00 |


| 7 | i\%/a | 7.00 | $\%$ annual target yield, |
| :--- | ---: | ---: | :--- |
| $\mathbf{4 0}$ | PMT | -40.00 | monthly payments, |
| $\mathbf{0}$ | FV PV | -464.98 | equivalent annual payment. |

## Example for finding the future value for an annuity due:



## Solution: ${ }^{267}$

| 12 per/a | 12.00 months per year, |
| :---: | :---: |
| $2 \times \mathrm{n}_{\text {PER }}$ | 24.00 payment periods in total, |
| 6.1.4 i\%/a | 6.25 \% annual target yield, |
| 50 PMT | 50.00 monthly payments; |
| 0 PV FV | -1 281.34 balance after two years. |

[^144]
## APPENDIX 4: GRAPHICS

You have learned about graphic output via ASSESS and PLOT in OMS 2. Furthermore, your WP43 features the elementary graphic commands CLLCD, AGRAPH, PIXEL, and POINT. The latter three are stored in P.FN'P.FN2; the first three are inherited from the HP-42S.

PIXEL sets the pixel with coordinates $\boldsymbol{x}=\mathrm{IP}(\boldsymbol{x})$ and $\boldsymbol{y}=\mathrm{IP}(\boldsymbol{y})$. The origin, i.e. pixel $(0 ; 0)$ is located in the bottom left corner of the screen, the pixel (399; 239) top right.

CLLCD clears all pixels with $x \geq \operatorname{IP}(x)$ and $y \geq \operatorname{IP}(y)$, i.e. the entire screen northeast of the point specified. Input of $(0 ; 0)$ will erase the whole screen.

POINT sets a square of $3 \times 3$ pixels around the coordinates $\boldsymbol{x}=\operatorname{IP}(\boldsymbol{x})$ and $\boldsymbol{y}=\operatorname{IP}(\boldsymbol{y})$. This is better visible than just a single pixel.

AGRAPH puts a vertical pattern specified by a 64-bit short integer in the source $\boldsymbol{s}$ to the LCD. The point with coordinates $x=\operatorname{IP}(x)$ and $y=\operatorname{IP}(\boldsymbol{y})$ is the bottom pixel of this column (valid inputs are $0 \leq x<400$ and $0 \leq y<240$ ). AGRAPH increments $x$ automatically by 1 which is handy (see an example below). So one 64 px high graphic row starting in column 0 may require 400 such integers - the more blank space is found in this row the less integers are needed for specifying it.

Since most keyboard commands refresh the screen immediately, graphics should be generated by routines. PIXEL, CLLCD, and POINT are self-explanatory. AGRAPH may need a little demonstration:

## Example:

This little program draws an open square with its bottom left corner at (150; 100). It may be provided on your WP43 already. Press

## GTO PROG PICTURE

## LBL 'PICTURE'

 FF FF FF FF FF FF FF FF ${ }_{16}$ set up two pattern registers for vertical... STO 00CO $0000000000000003_{16}$
... and horizontal framing STO 01
62
STO 02
100
ENTER $\uparrow$
150
CLLCD
INC X
AGRAPH 00
AGRAPH 00
LBL 01
AGRAPH 01
DSE 02
GTO 01
AGRAPH 00
AGRAPH 00 RTN

Now do this:

$$
P / R
$$



## XEQ PROG PICTURE

erase the screen NE of $(150 ; 100)$
draw left frame
drawing loop
loop: draw 1 point of top and bottom frame
loop: decrement the loop counter
loop: if $>0$ then run this loop again
else draw right frame

## leave PEM

fill the screen
and you will get a picture like this:


Note the status bar overwrote the upper 20 pixels; calling menus or pressing EXIT will overwrite the menu section - the whole graphic, however, will disappear without any trace when you call a function like ENTERT. Remember to add SNAP or 目LCD to your routine if you want a more permanent output of your graphic.

Another example, may be on your WP43 as well:

## GTO PROG SPIRAL

$P / R$

## LBL 'SPIRAL'

RCL I
$x=0$. ?
1.

STO I
RCL J
$x=0 . ?$
0.3

## |x|

STO J
RAD
LocR 04
1
STO . 00
0
ENTER*
CLLCD
LBL 01
RCL . 00
500.
/
ENTER*
ENTER $\uparrow$
sin
$x \geqslant y$
RCL× J $e^{x}$
$\times$
RCL× J
STO 01
$x \geqslant y$
ENTER $\uparrow$
cos
$x \geqslant y$
$e^{x}$
$\times$
RCLx I
STO . 02

RCLx J $\left.\quad\left[j \times r .00 / 500, \cos (r .00 / 500), j \times e^{\wedge}(\ldots)\right) \times \sin (\ldots), \ldots\right]$
is there an $\boldsymbol{i}$ provided?
default for $\boldsymbol{i}$
is there a $\boldsymbol{j}$ provided?
default for $\boldsymbol{j}$
ensure positive $\boldsymbol{j}$
allocate 4 local registers for this routine
initialize the counter r. 00
clear the entire $L C D$ (everything 'northeast' of $(0 ; 0)$ ) drawing loop
start with 0.002 in first run
$[r .00 / 500, r .00 / 500, r .00 / 500, \ldots]$
$[j \times r .00 / 500, \sin (r .00 / 500), r .00 / 500, \ldots]$
$\left[\mathrm{e}^{\wedge}(j \times r .00 / 500) \times \sin (r .00 / 500), r .00 / 500, \ldots\right]$
$\left[j \times e^{\wedge}(j \times r .00 / 500) \times \sin (r .00 / 500), r .00 / 500, \ldots\right]$
$\left.\left[r .00 / 500, r .00 / 500, j \times \mathrm{e}^{\wedge}(\ldots)\right) \times \sin (\ldots), \ldots\right]$
$\left.\left[e^{\wedge}(\boldsymbol{j} \times r .00 / 500) \times \cos (r .00 / 500), \boldsymbol{j} \times \mathrm{e}^{\wedge}(\ldots)\right) \times \sin (\ldots), \ldots\right]$
$\left.\left[i \times \mathrm{e}^{\wedge}(j \times r .00 / 500) \times \cos (r .00 / 500), j \times \mathrm{e}^{\wedge}(\ldots)\right) \times \sin (\ldots),, \ldots\right]$

| $\rightarrow \mathrm{POL}$ |  |
| :---: | :---: |
| STO . 03 | store the radius |
| 234 | limit value |
| $x \leq$ ? $Y$ | if radius $y$ is beyond the limit |
| RTN | then return |
| RCL . 01 | else continue with y |
| 120 |  |
| + | add the center coordinate for y |
| ROUNDI | for an integer number of pixels |
| RCL . 02 | recall x |
| 200 |  |
| + | add the center coordinate for x |
| ROUNDI | for an integer number of pixels |
| SF 'IGN1ER' | ignore 1 error |
| POINT | while drawing the next $3 \times 3 \mathrm{px}$. point at ( $\mathrm{x} ; \mathrm{y}$ ) |
| CF 'IGN1ER' | watch errors thereafter again |
| PAUSE 00 | update display |
| RCL .03 | recall the radius |
| 1/x |  |
| SDL 03 | multiply times 1000 |
| IP |  |
| INC X |  |
| STO+ . 00 | add this to r. 00 |
| GTO 01 | run this drawing loop again until above limit is reached |
| END |  |

Now do this:

P/R
XEQ PROG SPIRAL and wait (for $\sim 60 \mathrm{~s}$ with power supplied by battery). You will get a picture like this:

2023-03-27 15:20


You find more programs in testPgms.bin. Feel free to try them.

## APPENDIX 5: POWER SUPPLY

Your WP43 is powered by a single CR2032 coin cell (3V). It is not rechargeable. Insert it with $\boldsymbol{\text { facing up (away from the board). With a }}$ good battery installed, your WP43 may be also powered through its USB port - running at even higher speed then. When the battery becomes low, replace it with a new quality cell (cf. pp. $15 f$ and see the $R e M, A p p$. A).

WARNING: Removing the battery for longer than xxx seconds may erase all data in $R A M$ - only data in $F M$ will remain.

See here what sufficed for explaining the basic functionality of HP's first
 'advanced scientific' calculator, the HP-45, o n it s backside in 1973:

Though it featured only 59 functions, neither menus, catalogs, data types, browsers, applications, advanced operations (just four statistical sums, $\overline{\mathrm{x}}$ and (s) ), named variables, programming, diagrams, graphics, nor customizing - but your WP43 provides them all and much more.

## APPENDIX 6: TIME LINE OF QUOTED MANUALS

HP-35 OM ..... 1972
HP-55 OH, HP-21 OH, HP-25 OH ..... 1975
HP-27 OH, HP-67 OHPG ..... 1976
HP-97 OHPG, HP-32 OH, HP-33 OH ..... 1978
HP-34C OHPG ..... 1979
HP-41C/41CV OHPG ..... 1980
HP-16C Computer Scientist OH ..... 1982
HP-15C OH ..... 1987 (reprint 2011)
HP-22S Science Student Applications, HP-27S OM, ..... 1988
HP-42S OM
HP-21S OM ..... 1990


For comparison: In 1976, Continuous Memory was a breathtaking innovation; and a grand total of 72 built-in functions, 8 GP registers, and 49 (!) merged program steps on the HP-25C sufficed for professional engineers and scientists doing their work - and of course they did for students striving for their Ph.D.

Though you had to program linear regressions, correlations, and forecasting manually if you needed them


## APPENDIX 7: RELEASE NOTES

|  | Date | Release notes |
| :---: | :---: | :---: |
| 0 | 29.11 .12 | Official project start with first publication of the 43 S concept and layout on one of the forums of the Museum of HP Calculators (https://www.hpmuseum.org/cgisys/cgiwrap/hpmuseum/archv021.cgi?read=234685\#234685). <br> Though internal discussions started in September 2011, and there are even found far older traces of a ' $43 S^{\prime}$ ' denoting a 'Super HP$42 S$, though in various more or less fictional cases - pure vapourware ${ }^{\text {TM }}$. |
| 0.1 | $\begin{gathered} 2.2 .14 \\ 23.5 .15 \end{gathered}$ | Manual setup based on the one of WP 34S. <br> Passed to Jake Schwartz, Eric Smith, and Richard Ottosen for first information. <br> See the vivid discussions starting in February 2014 in this thread https.//www.hpmuseum.org/forum/thread-593.html (it was stopped at 10.1.16 but still is the thread most viewed on that forum). |
| 0.2 | 3.10 .15 | Update based on Jake's feedback and further thoughts, distributed to Eric, Jake, Marcus v. Cube, and Paul Dale. |
| 0.3 | 21.3.16 | Split the manual in three; moved LBL onto the keyboard, renamed STOM to STOCFG, RCLM to RCLCFG, SERR to $\mathrm{s}_{\mathrm{m}}$, and SERR ${ }_{\mathrm{w}}$ to Smw; refined the Key Response Table. Passed to Michael Steinmann for information. |
| $\begin{aligned} & 0.4 \ldots \\ & 0.22 \end{aligned}$ | 15.9.22 | See the full Release Notes in the ReM. |
| 0.23 | 3.12.22 | Added omicron in AIM. Relabeled WP43S to WP43. (1) is replaced by $\because$ on the keyboard. Added omicron in AIM. Pilot run performed and distributed. Introduced $\sqrt{1+x^{2}}$, status indicators 4 and 8 , and virtual keyboards for RBR, FBR, STATUS, and TIMER. Elaborated about histograms. Renamed TIMER ADD to T. $\Sigma+$. Refined App. 4, AGRAPH, CPX, erf, erfc, EXITALL, EXPT, J $\rightarrow$ D, LNГ, MANT, MATR, PART, $V_{4}$, $x!$, aMATH, $\Gamma(x), \zeta(x), \neq$, and $\boldsymbol{๒}$. Corrected. |
| 0.23.3 | 21.1.23 | Implemented DELITM, SAVE and LOAD of named variables, confirmation for CLP, RBR for named variables, 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, $\Sigma+$ memory leakage, CNST indirection, and possible crash of GTO/XEQ after LOAD. Optimized some common functions. |


|  | Date | Release notes |
| :---: | :---: | :---: |
| 0.23.5 | 23.2.23 | Implemented VARMNU, NUM.IN and ALP.IN. Fixed possible crash of CLP, $\mathrm{W}_{\mathrm{p}}$, and matrix division; corrected tan in Multm, KEYG and KEYX (removed K as alpha register, cf. 0.7), DECOMP, M.LU; corrected complex matrix subtraction, eigenvalues and eigenvectors, $\mathrm{y}^{\mathrm{x}}$, and $\beta$; cube root with FASTFN set, COMB and PERM, special cases of $\ln \Gamma, g_{d}, \rightarrow R$, and DECOMP. |
| 0.23.6 | 23.2.23 | As 0.23 .5 but with updated manuals. |
| 0.23 .7 | 11.3.23 | Implemented ACTUSB (also on the calculator keyboard), DMCP, and DUMP. Refined NEXTP; refined VARMNU, $x \rightarrow \alpha$, and handling of empty strings to meet the specs of HP-42S. Elaborated on I/O. |
| 0.23.8 | 11.3.23 | Corrected a fatal bug in the simulator. Else unchanged. |
| 0.23 .9 | 14.3.23 | Added the angular unit mil. Refined ALL. |
| 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 | 9.4.23 | Refined BIT, CLPALL, INTS, PAUSE, RCL, VARMNU, VIEW, \#, the comparisons, App. 4, and the QRG. Corrected. |
| 0.23 .12 | 5.5.23 | Added LOADST and SAVEST. Refined handling of undefined variables, I/O, SOLVE, VARMNU, $x \rightarrow a$, $\int$, the status bar, OMS 1, 3, and 4, ReMS 3, App. B and J. Corrected. |
| 0.23 .13 | 1.6.23 | Added error 53. Removed error 19, FLASH, RAM, PRCL, and PSTO. Reorganized access to FM. Implemented READP and WRITEP. Refined CLREGS, LOADP, LOADR, LOADSS, LOADV, LOAD乏, PROGS, SNAP, SOLVE, App. I, creating user menus, and displaying global labels. Corrected. |
| 0.24 | 2.7.23 | Added DISK? and character translation for editing programs on a computer. Refined head text of IOI, EXITALL, INFO, INT?, J/G?, SNAP, x!, $\underline{\alpha \bullet}$, some names of conversions, and stack diagrams. Removed FLASH? and $\overline{\boldsymbol{q}}$. Corrected. |

## INDEX

This index lists special terms and keywords used in this OM. Furthermore, it points to the most prominent of the over 170 examples included, marked '(ex.)'. You find the items provided on your WP43 explained in the IOI (and a list of all of them in ReM, App. J); since this will also point you to further explanations if applicable, particular items are listed below only if they are extensively covered in this OM. The Table of Contents is recommended as well - chapter headings are not repeated below.

## 42 (ex.) 149

about to die (ex.) 205
account balancing (ex.) 35
address space 56
addressing, indirect 67
ADM 139
advertising pictures (ex.) 102
AIM 81, 203
aircraft navigation (ex.) 144
Albert Einstein 291
Alpha Centauri (ex.) 53
alpha input mode 81, 203
angular display mode 139
APR 332
archery statistics (ex.) 107
arithmetic shift 156
automatic stack lift 38
balloon trip (ex.) 90
base conversions 153
bending steel ring (ex.) 100
bicycle gearing (ex.) 91
bit numbering 154
black or white cats (ex.) 206, 207
branching 224, 228
bubble sort (ex.) 230
caesium-133 290
caesium-137 97
calculator stand (ex.) 138
cannery (ex.) 215
carbon-14 dating (ex.) 259
carry 155
catalogs 29
CDF 104
chi-square statistic (ex.) 115
cleaning a veranda (ex.) 295
closing numeric input 25
complex aircraft navigation (ex.) 168
compound interest (ex.) 51
confidence limits (ex.) 117
connecting peaks (ex.) 101
Conrad Röntgen 297
continuous distribution 104
creating a new name 59
cross product (ex.) 169, 184
cubic inches (ex.) 295
Cullen numbers (ex.) 229
current step 213
data type 72,74
data type conversions 78
dating (ex.) 200
decrement and skip 228
dice from Las Vegas (ex.) 116
diffraction pattern (ex.) 274
digit group separation 84
discrete distribution 103
dot product 169, 184
dyadic functions 35
earthquakes (ex.) 95
electron-volts (ex.) 291
engineer's notation 85
enter exponent 25
ENTER4 37
error probability 104
extrapolation (ex.) 113
falling in Pisa (ex.) 110
falling with drag (ex.) 238
fencing land (ex.) 20
FILL 41
filling tires (ex.) 295
fixed point notation 85
flags 57
forecasting (ex.) 113
free fall (ex.) 110
Fukushima (ex.) 97
Galileo Galilei 110
general purpose registers 57
Golden Bow (ex.) 107
Greek letters 203
Horner scheme (ex.) 52
improper fraction 137
increment and skip 228
indicators, status bar 79
indirect addressing 64
INPUT 233
integer sign mode 151
interpolation (ex.) 112
lodine-131 298
ISM 151
item 27
item name 59
keypress checking 235
keystroke programming 212
long integers 149
Mach number (ex.) 48
Marie Skłodowska Curie 297
markup and margin (ex.) 133
measuring capability (ex.) 122, 193
measuring system analysis (ex.)
121
menu 27
menu section 28
menu view 27
modulo 158
monadic functions 34
Mother's Kitchen (ex.) 215
Mt. Everest (ex.) 93
MVAR 234, 262
MyMenu 302
Mya 307
names 59
navigating in space (ex.) 146
new item name 59
Odysseus 141
operating characteristics (ex.) 124
overflow 152
parachutist (ex.) 238
PDF 104
PEM 213
Penelope 141
PMF 104
prefix 19
primary function 19
process capability 108
program editing 220
program pointer 213
proper fraction 137
proton in magnetic field (ex.) 184
quantile function 105
$R \downarrow$ R4 41
radioactivity (ex.) 97, 259
radioactivity (units) 296
radix mark setting 84
RCL 61, 64
remainder of division 157
Rigel Centaurus (ex.) 53
rotate bits 156
rotate text 209
RPN 15, 35, 49
scientific notation 85
scrap rate (ex.) 117
search text 209
shift bits 156
shift text 209
short integers 150
significant change (ex.) 122
significant improvement (ex.) 109
Sirius (ex.) 53
skiing (ex.) 88
skydiving (ex.) 238
sorting numbers (ex.) 230
special registers 57
squaring circles (ex.) 87
stack 36
stack overflow 47
startup default 55
statistical registers 55, 57
status bar 79
submenu 29
surfaces of Jupiter's moons (ex.) 22

TAM 60, 220, 277
tax deduction (ex.) 133
temporary alpha mode 60
temporary information (TI) 71
tool life (ex.) 125
Tower of Pisa (ex.) 110
triadic functions 39
Upper Lagunia (ex.) 101
user flags 58
variables 59
VARMNU 234
vector operations in 2D 185
Venturi tube (ex.) 89
VIEW 62
virtual keyboard 60, 204, 220
Willie's Widget Works (ex.) 102
XEQ 23


[^0]:    ${ }^{1}$ RPN stands for Reverse Polish Notation, a fast, effective, and coherent method for most efficient solutions to computing problems. It is based on a mathematical logic known as Polish Notation (since invented by the Polish logician Jan Łukasiewicz, see https://www.youtube.com/watch?v=qRrAj-GCTQM). He put operators before numbers or variables instead between them as in conventional mathematical notation. $\underline{R P N}$ places operator behind numbers. See Section 1 of this manual.

[^1]:    ${ }^{3}$ They did this for their DM42 launched in 2017. Its layout is closely linked to the HP-42S produced until 1995, its firmware uses Thomas Okken's Free42 simulating the HP-42S.
    ${ }^{4}$... with major contributions of Italian Gianluca Puggelli, Dutch Harald Overbeek, German Gert Menke and Friedrich Mütschele.
    This part of your WP43 is based to large extent on the experience Paul and I gained with the WP 34S RPN Scientific calculator. We started that project in 2008 when the flashable HP-20b became available. Together with Marcus von Cube, we launched it in 2011. You find specific information about the WP 34S and its derivative, the WP 31S of 2014, at https://sourceforge.net/projects/wp34s/ and the links mentioned there. Both these calculators are based on HP hardware.

[^2]:    ${ }^{5}$ If you are interested in the long and winding road how your WP43 got the features, shape, and layout you are facing now, see the Release Notes in App. 7 of this manual. And feel free to look up GitLab to find out who contributed when and what.
    ${ }^{6} H P-42 S$ is our reference - if in doubt we follow it. Note WP43 is not (and was never
    planned to be) $H P-42 S$ compatible, however.

[^3]:    ${ }^{7}$ The last three not participating yet are Liberia, Myanmar, and the USA. We have no idea what they are afraid of - and they obviously do not know what they miss. Find basic information about S/ in https://en.wikipedia.org/wiki/International System of Units. See also the chapters about Constants and Unit Conversions in OMS 5.

[^4]:    ${ }^{8}$ No, we do not think you are stupid, irresponsible, or lack any life experience. Such a waste of print space became inevitable, however, due to the legal system, lawyers, and courts of that great nation where such kind of warnings appeared first.

[^5]:    ${ }^{9}$ For better readability in the manuals, we will usually refer to keys using dark print on light background from here on. Referring to shifted functions like x! or PROB, we will omit $\square$ or $\square$ most times since redundant by color print.
    ${ }^{10}$ Assume it in suburbia (remember?).
    ${ }^{11}$ Here, we use outdated British Imperial units since they are still employed in some former British colonies and territories.

[^6]:    ${ }^{12}$ Press , then $\mathbb{E}$ to access DISP; then the menu DISP will open in the lower third of the screen. Now press ©1 for FIX. Then enter © (1), telling your WP 43 it shall display only 1 decimal digit (internally, everything is handled with full precision always). In OMS 2, you will learn more about output formatting.

[^7]:    ${ }^{15}$ Program $\mathbf{T}$ as recorded above is very short and simple: Its center part contains five keystrokes only ( ENTERT $\boldsymbol{x} \square \pi \mathbb{X}$ ). You may store far more in program memory - the overall procedure of storing and running programs will remain unchanged; only the center part of the program will grow. Programming is covered in OMS 3.

[^8]:    ${ }^{16}$ There are also other numeric data types like integers, times, or dates available on your WP43 - they will be covered in OMS 2 together with more output formats provided.
    ${ }^{17}$ If you pressed a prefix erroneously, press it once more to cancel it.

[^9]:    ${ }^{18}$ Throughout the OM and ReM, we print menu names and menu labels underlined for the same reason. Just on the keys EXP and TRI, the lines are omitted for space reasons. Note, however, all menu names are stored in your WP 43 without an underline; hence they are also displayed without it.

[^10]:    ${ }^{19}$ Note the multiplication and division keys on the physical WP43 are labelled $\mathbf{x}$ and $\div$ for sake of tradition only, although such a printed $\div$ is visually quite similar to a $\oplus$ and irritates a great set of users initially. Instead, those keys could read $\bullet$ and $\square$ as well but this would have upset another great set of users.
    As a compromise, we print $\square$ in the manuals whenever we talk about the division key. It is displayed as / in WP43 programs and equations for the same reason.
    Note ISO 80000-2 recommends not using the symbol $\div$ at all. See also the ReM.
    $\mathbf{x}$ is employed for multiplication although this choice may cause ambiguities in vector calculus, see OMS 2.
    Users preferring $\bullet$ and $\ddagger$ over $\boldsymbol{X}$ and $\doteqdot$ can achieve this easily using a black permanent marker.

[^11]:    ${ }^{23}$ If it requires parameters it will execute as soon as parameter input is completed.

[^12]:    ${ }^{24}$ This rule applies to functions regardless of the kind of objects they operate on. This way of writing operations is called postfix notation since the operator is entered after the operands (cf. f. 1). It suits electronic calculating very well; and also eases work for human brains - see further below.

    The picture overleaf is copied from HP's brochure ‘ENTERT vs. ©’ of 1974.
    Some people might claim said global rule strictly holds for $R P \underline{L}$ only. $R P L$ (meaning Reverse Polish Lisp) is a programming language and notation developed from RPN in the 1980's. Maybe those people are even right. In my opinion, however, RPL strains the postfix principle beyond the pain barrier, exceeding the limit where it becomes annoying for human brains. Not for everybody, of course, but also for many scientists and engineers. Thus, we stick to classic RPN on WP43 as we did on WP 34 S and 31S.

[^13]:    ${ }^{25}$ This optional 8-level stack was invented by Paul and me and launched with WP 34S in 2011. WP 31S features it as well. See the further text for its advantages.

[^14]:    ${ }^{26}$ There are other dialects of RPN but this is the main stream. - The moving background colors printed in the stack diagram are explained in next chapter.

[^15]:    ${ }^{30}$ This traditional basic review functionality turns pointless when all four stack registers are permanently displayed for a traditional 4-level stack on your WP43. $\mathrm{R} \downarrow$ and $\mathrm{R} \uparrow$ may be beneficial, however, under particular conditions nevertheless - see examples further below.

[^16]:    ${ }^{40}$ And there is (and will be) no warning: even a watchdog on $\mathbf{T}$ will not help since some special advantages of RPN require T loaded (see next chapter). So you shall rather watch it yourself if you chose a 4-level stack.

[^17]:    ${ }^{41}$ We quote this US-American Mach-number formula without having verified it.

[^18]:    ${ }^{42}$ Contriving an example leading to stack overflow even for 8 stack registers is trivial; though that will be just a contrived example - no real-world formula. This is why there is no need for more than 8 stack registers. A so-called 'infinite' stack would render top stack register repetition impossible - see next chapter for benefits of the latter.

[^19]:    ${ }^{43}$ You might ask: With the opportunity of an 8-level stack, why are there only four stack levels displayed, not more?
    Simple reason: Once accustomed to RPN, you know the way it deals with your data on the stack. Consistently. Thus, watching the stack mechanics reliably working as expected carries no valuable information and will become boring or distracting very soon. The overwhelming majority of RPN pocket calculators built so far displayed $\boldsymbol{x}$ only, and $\boldsymbol{y}, \boldsymbol{z}$, and $\boldsymbol{t}$ were quietly acting unseen. Users were doing all sorts of tricks using them just tracking $\boldsymbol{y}, \boldsymbol{z}$, and $\boldsymbol{t}$ in their minds. Even HP's RPL calculators (although featuring a so-called 'infinite' stack) display no more than the bottom four registers.
    Nobody expects you tracking $\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{t}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, and $\boldsymbol{d}$ in your mind - it is simply not necessary at all. You know how the contents move, and they will show up when needed. No operation will use more than $\boldsymbol{x}, \boldsymbol{y}$, and $\boldsymbol{z}$ at once.
    Assuming human mental abilities did not deteriorate in last decades, displaying more than four registers carries no lasting benefit. This holds in particular since the odds for stack overflow in real-world calculations are reduced to zero when employing an 8-level stack, calculating inside out as recommended.

[^20]:    44 TNG: Compound interest $=$ Zinseszins.
    ${ }^{45}$ Those were the days, my friend ... ! Inflation was balancing those interest rates but saving was definitely more fun then, nevertheless.
    ${ }^{46}$ Debt calculations are significantly more complicated - thus, avoid debts as long as possible, in your very own interest! In the long run, it is better for you and your economy. When debts become inevitably necessary, however, you and your WP43 can cope with them as well (see TVM in OMS 5).

[^21]:    ${ }^{47}$ Try solving such a problem with an 'algebraic' calculator (made by e.g. Texas Instruments, Casio, or Sharp) for comparison - it will cost you many more keystrokes ${ }^{48}$ The few commands changing $\boldsymbol{x}$ but not loading $\mathbf{L}$ are mentioned explicitly in the IOI.

[^22]:    ${ }^{49}$ This is identical with Alpha Centauri. Rigel usually means a star in constellation Orion. ${ }^{50}$ Looking at the precision of the other inputs, an input of $9.46 \times 10^{15}$ had sufficed here.
    ${ }^{51}$ Allocating a dedicated keyboard label to LASTx (like on HP-42S or DM42) does not pay on your WP43 since no keystroke will be saved.

[^23]:    ${ }^{52}$ Note $\curvearrowleft \sim$ will not undo any operation you confirmed explicitly (like RESET, see next page), however. Look up the $I O I$ in this matter. And $\curvearrowleft \sim$ will undo just the very last operation before ( $\curvearrowleft$, nothing more - i.e. $(\curvearrowleft)(\curvearrowleft)=(\curvearrowleft)$.
    Previous RPN calculators used LASTx for error correction - $\curvearrowleft$ works easier and more comprehensive.

[^24]:    ${ }^{53}$ You can as well use DROP $\downarrow$ to get rid of $\boldsymbol{x}$. See the $/ O /$ for the subtle differences.
    ${ }^{54}$ CLREGS, CLFALL, CLPALL, CLALL, and RESET ask for your confirmation. Find more about GP registers in next chapter. Note stack and variables are not touched by CLREGS.
    ${ }^{55}$ Note display formats as well as other user settings and assignments will remain unchanged. Only RESET clears everything except FM (see OMS 3 and 6).
    ${ }^{56}$ If you cannot reach the command RESET for any reason whatsoever, a hard reset will do almost the same: use the RESET hole on the calculator backside.

[^25]:    ${ }^{57}$ System flags, on the other hand, carry names and reflect or control specific system states - see ReMS 1; some of them overlap with lettered user flags for easier access.

[^26]:    ${ }^{58}$ What is printed in white on your physical WP43, on the other hand, is called a default primary function.

[^27]:    ${ }^{59}$ For ...EL and ...IJ see OMS 2.

[^28]:    ${ }^{60}$ In OMS 3, you will learn about a method preventing your programmed routines from interfering with data of other programs.

[^29]:    ${ }^{61}$ We recommend calling variables already defined via VAR instead of keying them in.
    62 Note you can skip pressing $\square$ here (cf. the virtual keyboard on p. 59).
    ${ }^{63}$ Cf. p. 59. - Attempted writing in a variable undefined at execution time (e.g. UNDF) will create it automatically, containing zero initially. On the other hand, attempts to read from an undefined variable will throw an error.

[^30]:    64 For $($ RCL and STO only, any of the keys $\oplus, \square, \boldsymbol{x}, \square, \Delta$, or $\square$ may precede step 2 here. Entering such a key twice will cancel it (e.g. $R(\mathbb{X C L} \boldsymbol{x}$ equals RCL ). See the chapter after next chapter for this store and recall arithmetic.

    Note such operators are disallowed in RCL Config (= RCLCFG), RCL Stack (= RCLS), RCLEL, RCLIJ, and the corresponding store operations.

[^31]:    ${ }^{65}$ Where applicable, we recommend calling system flags via their shortcuts or via SYS.FL and variables already defined via VAR, instead of keying them in using $\alpha$.

[^32]:    ${ }^{66}$ Several vintage calculators, on the opposite, featured just a single dedicated register for indirect addressing, if at all. See the HP-34C or HP-15C, for instance.

[^33]:    ${ }^{67}$ If you choose displaying less than four stack registers (look up DSTACK in the IOI), echoes and $T /$ will nevertheless show up at the positions specified above, whenever applicable. And operations resulting in multi-row output will display their entire output independent of the DSTACK setting always.
    ${ }^{68}$ These data types are covered comprehensively in dedicated chapters further below. All data types provided are listed in the ReM.

[^34]:    ${ }^{75}$ Where a field carries a red number here, the combination is physically illegal but frowningly allowed for people who cannot get accustomed to strong data types.
    ${ }^{76}$ For example, $15 / 3$ returns 5 while 14/5 will return 2.8.
    ${ }^{77}$ The time interval will be converted into decimal hours before division.
    ${ }^{78}$ The matrix $\boldsymbol{x}$ must be invertible. Dividing by $\boldsymbol{x}$ is equivalent to multiplying times the inverted matrix $\boldsymbol{x}^{-1}$ (see the chapter Vectors and Matrices: Calculating below).

[^35]:    ${ }^{81}$ The symbol " $\&$ " denotes a logical "and" and a comma a logical "or" in this table.

[^36]:    ${ }^{82}$ See https://en.wikipedia.org/wiki/Decimal separator for a world map of radix mark use. Looks like an even score in this matter. Thus, the international standard ISO 80000-1 allows either a decimal point or a comma as radix mark and requires a narrow blank as unambiguous separator of digit groups (it explicitly states that points or commas shall not be used as group separators to avoid ambiguity).
    ${ }^{83}$ As far as we know, WP43 is the $1^{\text {st }}$ pocket calculator displaying numbers the way internationally agreed on. Previous calculators featuring limited displays had to use e.g. points or commas as crutches since they could not display narrow blanks.
    ${ }^{84}$ See https://en.wikipedia.org/wiki/Date format by country also for a world map of date formats used. The international standard ISO 8601 states Y.MD for dates and 24h for times. This combination carries some internal logic and is common in East Asia (see SETCHN and SETJPN).
    ${ }^{85}$ Officially, the Gregorian calendar became effective at 1582-10-15 in the catholic world. Many states and territories switched later for various reasons. You can enter the date applicable at your location using J/G (see the $I O /$ for this command). Note there are still other calendars widespread - see the chapter Dates below.

[^37]:    ${ }^{86}$ Chinese counting and traditional mathematics work with powers of 10000 while （originally Indian，then Persian，then）European counting and mathematics work with powers of 1000．Thus，Chinese count using intervals－（1），十（10），百（100），千（1000），万（10000），十万（10×10000），百万 $(100 \times 10000)$ ，千万 $(1000 \times 10000)$ ，亿 $\left(10^{8}\right)$ ，十亿 $\left(10 \times 10^{8}\right)$ ，百亿 $\left(100 \times 10^{8}\right)$ ，千亿 $\left(1000 \times 10^{8}\right)$ ，etc．－compare the detail of a Chinese calculator keyboard pictured overleaf．The command GAP 4 takes care of this notation．GAP 3 formats the European way．
    ${ }^{87}$ Proper South Asian（a．k．a．Indian）formatting would require separators every two digits for numbers $>1000$ ．Think of lakh $=10^{5}$ and crore $=10^{7}$ ．Actually，an amount of 50 cr ． $\left(=5 \times 10^{8}\right)$ Rupees reads $50,00,00,000$ Rs．in Indian newspapers．
    ${ }^{88} 24$ h（Indian so－called＇military time＇）is taking over in the UK，so SETIND will work there as well then．

[^38]:    ${ }^{89}$ Actually, it stores even more - see OMS 6.
    ${ }^{90}$ No matter what format or notation you select, these options affect the display rounding only. Internally, your WP43 continues using its full precision (typically 34 digits for reals); it can be shown for $\mathbf{X}$ by SHOW until next keystroke.

[^39]:    ${ }^{91}$ FIX 2 is the favorite format of financial people. - Deviating from previous calculators, output of $\times 10^{\circ}$ is suppressed on your WP43 for SCI and ENG.

[^40]:    ${ }^{92}$ We actually helped this example by assuming the manometer was read with 1 mm precision. The original just stated 9 cm but printed a 5 -digit result.
    ${ }^{93}$ In the HP-21 OH of 1975, the balloonist Ike Daedalus had to increase the radius from 25 to 27 feet. 1976 brought some progress, at least.

[^41]:    ${ }^{94}$ Note you achieve only 12 out of 21 theoretically possible gears here. This is due to gear overlaps; and you will want to avoid extreme chain skew for sake of chain life.
    Conditions change if you plan for a recumbent bike featuring a long chain: Then you may think about a combination of three sprockets with a 7-gear cluster leading to 17 different, usable gears following a scheme called 'half-step-plus-granny'; speed increase in halfstep range will be $9 \%$ per gear step; this gearing will cover velocities from 5 to over $40 \mathrm{~km} / \mathrm{h}$ for 60 rpm pedalling frequency.
    During the Tour de France in 2022, 36 vs. 30 teeth was the granny gear for mountain roads in the Alpes. Some training was required obviously.

    For detailed specifications as well as pictures, graphics, diagrams, tables, and further information about gearing bicycles yourself, please order "Die Fahrradschaltung" (a book of 144 pages) written by the same author - just contact me.

[^42]:    ${ }^{95}$ You may be used to a calculator label LOG or log for the decadic logarithm; though this is a mathematically ambiguous notation, so we avoided it (cf. ISO 80000, 2-12.5 and 2-12.6; see also https://physics.nist.gov/cuu/pdf/sp811.pdf, p. 33, 10.1.2).

[^43]:    ${ }^{97}$ Maybe this is the reason why the last three countries on this planet do not switch to SIdo they fear the recalibrations inevitably necessary for their measuring equipment?
    ${ }^{98}$ The actual moment magnitude scale for earthquakes differs but is logarithmic as well.

[^44]:    ${ }^{99}$ Taking into account that published magnitudes of earthquakes never show more than one decimal, we did not lose anything real setting WP43 to FIX 0 here.
    For comparison: The asteroid impact at Chicxulub some 66 million years ago leading to extinction of the dinosaurs is estimated having caused an earthquake of magnitude 11.

[^45]:    100 In December 2011, the radioactive iodine blown out of the reactor in March had become irrelevant already ( ${ }^{131}$ I has a half-life of only 8 days). Thus, the activity recorded in the map is caused predominantly by caesium.
    Natural radioactive background is $(0.1 \pm 0.08) \mu \mathrm{Sv} / \mathrm{h}$ on average, depending on location.

[^46]:    ${ }^{105}$ The HP-32 was HP's $1^{\text {st }}$ pocket calculator featuring hyperbolic functions on its keyboard. It was launched in 1978. Note that the SR50 of Texas Instruments (HP's arch rival in those years of the so-called 'calculator wars') provided hyperbolic functions on its SR50 four years earlier already.

[^47]:    107 These challenges changed the photographer significantly within one year.
    ${ }^{108}$ In a nutshell, discrete statistical distributions deal with (an integer number of) "events" governed by a known mathematical model. Such statistical events may be persons entering a store, radioactive nuclei decaying, faulty parts appearing, etc. The PMF then tells the probability to observe a certain number of such events, e.g. 7. And the CDF gives the probability to observe up to 7 such events then, but not more.
    For doing statistics with continuous statistical variables - e.g. the heights of three-yearold toddlers - similar rules apply: Assume we know the applicable mathematical model; then the respective CDF gives the probability for their heights being less than an arbitrary limit, for example $<1 \mathrm{~m}$. And the corresponding PDF tells how these heights are distributed in a sample of let's say 1000 kids of this age.

[^48]:    ${ }^{109}$ Actually, $\Sigma+$ stores also $y$. Hence, your WP43 returns SDs for both x and y data but only the $S D$ for x is relevant for this problem.

[^49]:    112 Heaven beware! A big pebble would have done as well if not better. Just its touchdown must be heard.
    ${ }^{113}$ These raw data really do not look very plausible, and actually it is dubious whether Galileo made such experiments using the Tower of Pisa at all; but at least HP believed that its customers would believe that story in 1976.

[^50]:    ${ }^{115}$ Perhaps his ancestors emigrated from Greece: Hephaistos ("Н甲аıбтоऽ) is the ancient Greek god of fire and forging (and maybe of all underground natural resources, too?).
    ${ }^{116}$ Differentiating from los Estados Unidos Mexicanos, for example. But you have guessed the state probably by the weird units used below.

[^51]:    117 The boss computes himself! And he is even able to do it properly! Looks like that was a time before general managers, CEO's, and big staffs became fashionable. Though see also next example.

[^52]:    118 If he had plotted his data, Hephaestus could have been warned looking at his last three data points. Often, plots carry extra information which may be lost easily when dealing with numbers only. HP's example is a suboptimal choice for extrapolating without plotting and actually an inadvertent warning of thoughtless (or lazy) number crunching.
    ${ }^{119}$ Hardly any real-world agronomist would be interested in the sums mentioned in HP's example. Also mean and SD were for demonstration purposes only in said OH .

[^53]:    ${ }^{120}$ Extrapolations to higher nitrogen applications must be rated dubious since the plotted data show an evident curvature; so another curve fit model should be tried there.

    121 'Goodness of fit' tells how good both sets match. This paragraph and the following example are quoted from the $H P-27 \mathrm{OH}$.
    ${ }^{122}$ Do not confuse this $\chi^{2}$ defined here with the $x^{2}$ distribution mentioned in previous chapter. They are different! Unfortunately, however, both chi-squares are called and spelled equally. It looks like the mathematical naming commission was inattentive here at the crucial time, perhaps distracted by discussing the outcome of a recent casino visit (note probability theory and statistics were originally invented striving for comprehending gambling).

[^54]:    ${ }^{123}$ Does this confidence level satisfy you?

[^55]:    ${ }^{124}$ This is a simplified example. In real life, also other errors will contribute.

[^56]:    ${ }^{125}$ Some sets of data may consist of a series of $\boldsymbol{x}$-values (or $\boldsymbol{y}$-values) that differ from each other by a comparatively small amount. Here you may save considerable input time by keying in only the differences between each value and a number $\boldsymbol{m}$ approximating the average of the values. This $\boldsymbol{m}$ must be added to the calculated result of $\overline{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{x}}$, or $\mathrm{a}_{0}$ of L.R then. The SD does not change.
    E.g., if your $x$-values are 665 996, 665997 , and 665998 , you should enter them as 0 , 1, and 2 for $\boldsymbol{\Sigma +}$. When calculating $\overline{\boldsymbol{x}}$ afterwards, add 665996 to the answer.

[^57]:    ${ }^{130}$ Specifications are written facilely, actually verifying them in a production environment may become really hard.
    ${ }^{131}$ Nature allows for positive susceptibilities only. - Note it is not required to know the exact susceptibilities of your samples beforehand: they shall just fall in said range, cover it fairly homogenously (cf. the plot below), and be sufficiently resilient to stay constant in your measurements. No need for any investment in expensive gauges here - real life has proven various pieces of stainless steel scrap may well do here. But 30 samples are the minimum number necessary.

[^58]:    ${ }^{132}$ If the gross shape of your scatter plot deviates fundamentally from the one shown here, this may point to surprises hidden in your experimental setup and / or data acquisition. Consult an expert in metrology.
    ${ }^{133}$ Although displaying all $\geq 30$ measured points clearly separated might exceed the resolution of a small pocket calculator screen by far, the calculations of steps 6 and 7 carry full precision, and the resulting $\mathrm{S}_{\mathrm{mi}}$ will be correct.

[^59]:    ${ }^{134}$ This test assumes your samples were both drawn from a Gaussian process which is frequently the case in real life (but shall be verified).

[^60]:    ${ }^{135} \boldsymbol{n}$ BINS should be $\leq \sqrt{\boldsymbol{n}}$, with $\boldsymbol{n}$ being the total number of your points. Default is $\sqrt{\boldsymbol{n}}$.

[^61]:    ${ }^{136}$ Works like RND did on HP-42S.

[^62]:    ${ }^{137}$ Markup is the price difference as a percentage of cost (or wholesale) price.
    TNG: Markup = Aufschlag (z.B. auf die Produktionskosten), Handelsspanne (aber siehe nächste Fußnote).

[^63]:    ${ }^{138}$ Margin is the price difference as a percentage of selling (or retail) price.
    TNG: Margin = Handelsspanne, Marge (bezogen auf den Endpreis). Der Begriff Handelsspanne ist also leider mehrdeutig.
    139 See the ReM for the formula.
    ${ }^{140}$ Every engineer or scientist should be able to produce the very same result significantly faster via 221.82 ENTERT 1.19 (1).
    Seeing functions like \%+MG (and \%T in particular) provided on financial calculators, you may get the impression that average financial people might be mathematically slightly challenged and need some extra support.
    On the other hand, there was a saying in technical quarters (before 2008 already): "Wenn man sieht, was Kaufleute allein mit Plus und Minus alles anstellen, sollte man sie an höhere Rechenarten erst gar nicht ranlassen." (translated into English: 'Looking at the results financial / business people produce with plus and minus alone, their access to more advanced operations should be denied.')

[^64]:    141 This works almost like on the HP-32SII and its successors but with higher precision. 142 ... following an idea of Craig Finseth. Startup default denominator is 4.
    ${ }^{143}$ This display of a pure integer tells you unambiguously your WP43 is in proper fraction display mode. In improper fraction display mode, $1 / 1$ will be displayed instead. (For comparison, note the HP-32SII reads $1 \square \square 4$ as $1 / 4$ - though this is neither coherent with its other input interpretations nor does it save any keystrokes.)
    ${ }^{144}$ Actually, your WP43 continues calculating with reals, it just displays fractions as close to them as possible within the boundary conditions set by you (see next page).

[^65]:    ${ }^{145}$ TNG: Proper fractions decken sowohl echte Brüche (wie $3 / 4$ ) als auch gemischte Brüche (wie $21 / 2$ ) ab. Bei improper fractions wird der ganzzahlige Anteil nicht herausgezogen, so dass hier der Zähler größer als der Nenner sein kann.
    ${ }^{146}$ This conversion was newly introduced on RPN calculators with the WP 34 in 2011.

[^66]:    ${ }^{147}$ No such differentiation in SI - here a volume stays unchanged regardless of the matter filling it, be it e.g. air, liquid water, mercury, or solid lead.
    Honestly, unless you were raised on such an isolated irrational island, we bet you assumed fluid ounces being a unit of mass, didn't you? Since you have heard of ounces once before and just thought ... terribly wrong! Do not think there - you may run into deep troubles easily (thinking less, however, you might reach top positions in administration - see experimental evidence between 2016 and 2020).

[^67]:    ${ }^{148}$ All ADM setting commands (except D.MS) are stored in MODE. GRAD is (was?) used in surveying, MIL in military (artillery).
    $T N$ : The traditional calculator notations DEG and GRAD are misleading in German at least: DEGrees on your WP43 mean Grad, while calculator GRADs are generally called Gon in Continental Europe. MIL corresponds to Strich in German.
    ${ }^{149}$ Note there are no leading zeros in the angular minutes and seconds sections. And this ADM can neither digest nor display anything smaller than 0.01 ". On the other hand, it

[^68]:    150 This example is found in the HP-25 OH first. It was reprinted thereafter in each and every HP scientific pocket calculator manual until the HP-41C/41CV OHPG.
    ${ }^{151}$ This formula means that 1 nautical mile (nmi) corresponds to 1 angular minute on a great circle. This doesn't hold exactly but precisely enough for practical sailing.
    TN: Latitude means geographische Breite in German. Hence it is represented by the letter $\boldsymbol{B}$ in the formula above.

[^69]:    ${ }^{152}$ An alternative way for solving this kind of $2 D$ vector problems is shown on $p .159$.
    ${ }^{153}$ This kind of plane is still in service after fifty years. Our grandparents could not imagine.

[^70]:    ${ }^{157 \text { CC } \dagger \text { ( CC does the same one keystroke faster (see Complex Numbers below). }}$
    158 Those looking for an extra challenge can compute next how flat the crew will become within seconds after the auxiliary engine is ignited.
    This universal problem was reprinted in the HP-34C OH one year later. It vanished in hyperspace thereafter, without a trace.

[^71]:    ${ }^{159}$ Unless $\bmod (\boldsymbol{n} ; 9)=4$ or 5 . See two chapters below for the function mod.
    ${ }^{160}$ See https://phys.org/news/2019-09-sum-cubes-solvedusing-real-life.html

[^72]:    161 Your WP43 encompasses all the integer and bit manipulation operations of the dedicated Computer Scientist's HP-16C, plus all bases and the entire extended function set of WP 34S, actually making a second calculator obsolete.

[^73]:    162 This shortcut will be left as soon as you enter $\square, \boxed{E}, \mathbf{C C}$, or \# in input, even if you delete this keystroke thereafter.
    Illegal digits keyed in (e.g. 2 in base 2 or $\mathbf{B}$ in base 10) can be detected no earlier than said input is completed. And you may key in more than the current word size can take - also this will be checked when input is closed. So an error will be thrown only then.
    ${ }^{163}$ They behave like they did on HP-16C and WP 34S.
    ${ }^{164}$ Note the gap automatically inserted every four bits here for easier reading of binary numbers.

[^74]:    165 Actually, changing signs should have no meaning in unsigned mode per definition. Thus, $\dagger / \boxed{\infty}$ should be illegal here or result in no operation at least. "In unsigned mode, the most significant bit adds magnitude, not sign, so the largest value represented by a 12-bit word is 4095 instead of 2047" (quoted from the HP-16C Computer Scientist Owner's Handbook of April 1982, p. 30).
    +/] in unsigned mode was allowed, however, by the designers of the HP-16C and implemented as shown above. So we follow that implementation for sake of backward compatibility (as we did with WP 34S), though frowning.
    Luckily, this inherited error carries a benefit: You can easily (by a single key press) determine that a very big positive unsigned number in display may correspond to a small negative number for the word size chosen. This may be quite useful.

[^75]:    ${ }^{166}$ During numeric input, gaps are inserted suiting the base chosen (cf. two pages above). If there is no such base yet, GAP 3 applies in input since your WP43 cannot know in advance what you have in mind.

[^76]:    ${ }^{168}$ If you want to convert hexadecimal numbers to split octal, for whatever reason, you can do so inserting a $\mathbf{0}$ after every 2 hex digits. Vice versa you shall insert a $\mathbf{0}$ after every 3 octal digits. Example: ABOCDOEF0FFH $=253031503570377_{8}$.

[^77]:    170 The picture for ASR correctly describes this operation for ISM 1 and 2 only. ASR 3, however, equals a signed division by $2^{3}$ in all modes of the HP-16C, hence the different results for the latter two ISMs above. Turn to the IOI for further details.

[^78]:    ${ }^{171}$ Note both FLOOR and CEIL operate also on reals and return long integers.

[^79]:    ${ }^{173}$ E.g. $\sqrt{\boldsymbol{x}}$ will return 8 for an input of 64 , i.e. for a proper square, and $8.062 \ldots$ for an input of 65 , for instance. For an input of -64 , it may return $0 .+i \times 8$. (see Complex Numbers below). In analogy, $8 \sqrt[3]{x}$ returns 2, $1024 \mathrm{lb} \times$ returns 10, 1000 (1g returns 3, 1296 ENTERT $4 \sqrt[x]{y}$ returns 6, and 625 ENTERT $5 \log _{x} y$ returns 4, for instance.

[^80]:    ${ }^{174}$ Entering both parts vice versa would be more like RPN: first the imaginary part, then (CC) interpreted as $\boldsymbol{i} \times$, finally the real part to be added. But it was decided differently for the HP-42S already. So we follow this tradition here.
    For those working on the field of electronic engineering, an alternate format is provided for complex numbers employing the letter j for the complex unit instead; the respective system flag is called CPXj for obvious reasons.

[^81]:    Note that pushing 4 complex numbers on the stack is a challenge on HP-42S and DM42. It is easily performed on your WP43.

[^82]:    ${ }^{175}$ Whenever a complex number is returned, your WP43 will set CPXRES and light $\mathbb{C}$ in the status bar unless they are set before already.

[^83]:    ${ }^{176}$ Choosing rectangular notation and multiplication dots allows for displaying real and imaginary components using large font within $\left(10^{-999}, 10^{999}\right)$ in SCl 4 together. It will work in SCI 5 for both components within $\left(10^{-99}, 10^{99}\right)$. Staying with the startup default (i.e. MULTX) instead will cost you one displayed decimal in complex domain.

    In polar display mode, angles will be normalized to $(-\pi, \pi$ ] always, so there will be no space for a power of ten needed for an angle; hence this will allow for SCl 6 within ( $10^{-999}, 10^{999}$ ) regardless of the multiplication symbol chosen, and for SCI 7 within ( $10^{-99}, 10^{99}$ ). See HIDE and RANGE in the ReM.

[^84]:    177 If the problem you're working requires a true (three-dimensional) vector as a result, use a $1 \times 3$ or $3 \times 1$ matrix to represent each vector in three dimensions. See chapters after next.

[^85]:    ${ }^{178}$ ATTENTION: For odd (integer) $\boldsymbol{x}$ and $\boldsymbol{y}<0, \boldsymbol{y}^{1 / \boldsymbol{x}} \neq \sqrt[x]{\boldsymbol{y}}$. You should know that the latter returns the correct solution.
    ${ }^{179}$ Works like RND did on HP-42S.

[^86]:    ${ }^{180}$ ALL may be an even more favorable format sometimes. It will be, however, not chosen automatically.

[^87]:    ${ }^{181}$ Remember matrix multiplication behaves different than multiplication of plain numbers.
    Generally, for two arbitrary matrices $[A]$ and $[B]$ of matching sizes and shapes, $[A] \cdot[B] \neq[B] \cdot[A]$. Only square matrices can be squared.
    And matrix division is special: It is defined as multiplication of the numerator times the

[^88]:    inverse of the denominator. Therefore, $\boldsymbol{x}$ must be a invertible (i.e. nonsingular) matrix here - else you cannot divide by that matrix. Only square matrices may be invertible.

[^89]:    3 ENTERT 1 MATX DIM $\boldsymbol{\alpha}$ (ENTERT creates $\mathbf{v}$ as a $3 \times 1$ matrix. RCL VAR $v$

[^90]:    182 This procedure works as it did on the HP-42S. The number of unknowns is only limited by the free memory available in your WP43 at execution time.

[^91]:    183 TN: The German term eigen may translate here into English inherent, appropriate, or proper.

[^92]:    184 Steps 5 through 7 are the same as shown in the chapter Real Numbers: Some Industrial Problems Solved with input of separate data points already. They are repeated here just for your convenience.

[^93]:    185 TN: Numeric output of WDAY corresponds directly to Chinese weekdays 1 to 6. For Portuguese weekdays ('segunda-feira’ etc.), add 1 to days 1 to 5 .

[^94]:    ${ }^{186}$ Note the real time clock in your WP43 may deviate from true time by up to one minute per month or two seconds per day (i.e. $\pm 25 \mathrm{ppm}$ approximately, caused by parts tolerances; you live with these inaccuracies wearing a quartz watch as well - mechanical watches are less accurate generally). This does neither affect real-world time calculations nor the TIMER application described in OMS 5. If you are accustomed to radio controlled timepieces, however, you might find regular adjustments necessary.

[^95]:    ${ }^{187}$ For people writing German, for example, Mya might look like pictured overleaf. Feel free to put other letters in - see OMS 6 for how to populate Mya.

    188 This works wherever applicable, with "homophonic" following classical Greek pronunciation. We assigned Gamma also to $C$ following the alphabet, and Chi to $\mathbb{H}$ since this letter comes next in pronunciation.
    Three Greek letters require special handling since they lack single-letter equivalents in the English language: Psi is accessed via $\square$ ( $\mathbf{W}$ (since looking like $\mathbf{W}$ in a way), Theta via $\square U$ (following $T$ ), and Eta via $\square$ (J) these three letters are printed in blue on the physical keyboard as reminders. Omicron is not needed since looking exactly like the Latin letter $\mathbf{O}$ in either case. Note there is an 'alpha helper' printed on the calculator back supporting users challenged by Greek.
    Kudos to Thales, Pythagoras, Heraclitus, Leucippus, Democritus, Aristotle, Archimedes,
    
     and their colleagues for laying the foundations of logics, mathematics, and physics (i.e.
     - starting 2600 years ago. Note that the first two disciplines were called "practical arts" and the latter "theoretical science".
    And kudos to those unnamed Babylonian mathematicians who laid the foundations for the Greeks, actually recording e.g. what we now call "Pythagoras' theorem" as early as 1200 years (!) before him.

[^96]:    189 This corresponds to Latin MORITVRI TE SALVTANT．I created this example in 2015 looking at a banner in Athens．

[^97]:    ${ }^{190}$ Alas, he deviated from this motto later. And his successors follow that deviation so far, for whatever reasons or fears.

[^98]:    ${ }^{191}$ Some text strings displayed may be hard to tell apart from numbers. Text strings are, however, always shown in small font and there will be no automatic gaps in them.

[^99]:    ${ }^{192}$ You cannot see that first END but the last one is visible - pictured e.g. on p. 208.

[^100]:    194 We chose named variables instead of registers.

[^101]:    ${ }^{195}$ No units are mentioned in this problem. Assume they talk about inches, so the output is in square and cubic inches. May be necessary to convert in fluid ounces?

[^102]:    ${ }^{196}$ Mother's Kitchen might tend to use a spreadsheet application on its laptop to calculate such outputs nowadays. Your WP43 is smaller, boots faster, and is less prone to be hit by spaghetti sauce..

    197 These search procedures for local and global labels are as known since the HP-41C. Though the range of local labels varies.

[^103]:    ${ }^{198}$ Note you can skip pressing here - see overleaf. See also an alternative there.
    ${ }^{199}$ Indirect addressing works with all these operations except LBL).
    ${ }^{200}$ Said label must contain at least one letter. Labels are case sensitive. The $7^{\text {th }}$ character will terminate entry and close AIM - shorter labels need a closing ENTERT.

    201 ... if the respective local register is allocated. Note XEQ. is not allowed neither in RUM nor PEM. Some lettered registers may be dedicated to special applications. Cf. Addressing and Manipulating Objects in RAM in OMS 1.

[^104]:    202 ... or you interrupt it manually by pressing R/S or EXIT - then it will stop after the current step is executed completely. For resuming its execution, press R/S again.
    ${ }^{203}$ This is the standard way to run routines. Furthermore, you can define shortcuts to your favorite routines by customizing your WP43 as described in OMS 6.

[^105]:    ${ }^{204}$ Pressing (or $\overline{\underline{E}} \boldsymbol{\triangle}$ if a multi-view menu is displayed), on the other hand, moves the program pointer backwards in the current routine without executing anything.
    ${ }^{205}$ Watch that your additional checks do not alter the status of your WP43 in a way deviating from its status in automatic execution; else you shall compensate. Also take care when browsing backwards.

[^106]:    ${ }^{206}$ A similar picture is printed on the back of your WP43.

[^107]:    ${ }^{207}$ Cf. p. 65. Note that the HP-42S allowed for viewing the present value of one of the variables displayed by just pressing and the corresponding softkey. We cannot support this on WP43 since this offers you three menu rows.

[^108]:    ${ }^{208}$ Note R/S and EXIT cannot be queried since they stop program execution immediately.

[^109]:    ${ }^{209}$ This example is quoted from the Hewlett Packard Journal of November 1975. Some parts of the setup look arbitrary but it is for demonstration purposes only. We modified the original HP-25 program putting its quantities in named variables instead of registers for better readability and understanding.

[^110]:    ${ }^{210}$ Turn to the ReM, App. I, for background information about this method.

[^111]:    ${ }^{212}$ See HP-42 OM, pp. 145-148, and its Programming Examples and Techniques, pp. 29

    - 39, 92 -99, 158-160, and 184-192, for examples using the programmable menu.

[^112]:    ${ }^{213}$ Without this heading ', some special strings might be misinterpreted as numbers or commands in program listings. The character ' is not found in any menu.

    214 ...but you can, of course: if you do while trying to enter a new program step, you will read an error message RAM is full ; see the ReM, App. $C$ and $B$, for ways to escape from such a situation.

[^113]:    ${ }^{215}$ There will be only one backup in FM - calling SAVE again will overwrite the previous backup.

[^114]:    ${ }^{216}$ Instead, STOCFG stores the WP43 configuration in a register or variable, i.e. in RAM. Note system state files are not easily user-readable, nor are configurations.
    ${ }^{217}$ FM may not survive more than some 100000 flashes. Although this will be hardly reached in calculator normal use, we made commands writing to FM (SAVE, SAVEST, and WRITEP) non-programmable.

[^115]:    ${ }^{218}$ For DUMP and SNAP, the Simulator provides easier alternatives - see the ReM, App. E.

[^116]:    ${ }^{219}$ Type what you need. The index $\oplus$ is found in $\underline{\alpha M A T H}$, and accessed via $\square \square \ldots$ Picking one character from $\underline{\alpha M A T H}$ will insert this character into the equation and close $\alpha$ MATH, so you can continue editing. The same applies in analogy to the other two alpha menus when writing an equation.

    220 The system uses multiplication dots here for sake of better readability (you could hardly distinguish $\times$ and $\mathbf{x}$ in an equation). For the same reason, spaces are inserted automatically. Note implicit multiplications are not allowed in equations.
    ${ }^{221}$ A dashed line will show up over the current equation when there are more. To select another equation, press $\Delta$ or $\nabla$ then until the requested one appears.

[^117]:    ${ }^{222}$ Thus, the following (lazy) single-letter variables are not allowed in any equation due to name conflicts: a, c, e, k, A, B, C, D, F, G, I, J, K, L, R, T, X, Y, and Z.
    ${ }^{223}$ TNG: Der eingebaute Gleichungslöser (Solver) Ihres WP43 erlaubt Ihnen das Auflösen nach einer beliebigen Unbekannten bzw. das Finden der Nullstelle(n) einer beliebigen Gleichung.

[^118]:    ${ }^{224}$ Actually, the equation $h_{0}+v_{0} t-\frac{1}{2} g t^{2}=0$ was solved here. Its exact physical solution can be calculated being $t=\left(v_{0}+\sqrt{v_{0}^{2}+2 g h_{0}}\right) / g$. Take care $\left(v_{0}^{2}+2 g h_{0}\right)$ stays positive here - thrown out of a deep pit, the ball may never reach the ground. Note you can as well use SLVQ for solving freeFal.

[^119]:    ${ }^{225}$ You will have noticed that IBess is not an equation but just one side of it. To keep the system lean, such functions are listed under EQN nevertheless.

[^120]:    ${ }^{226}$ Note that $\int \approx$ vanishes with like every temporary information disappears with the next keystroke.
    We could have included the division by $\boldsymbol{\pi}$ in our function IBess. We did not, however, since IBess is evaluated many times during the integration process; thus, the fewer steps the integrand contains the faster the result can be returned.
    ${ }^{227}$ You may have noticed a function $\mathrm{J}_{y}(x)$ is implemented in your WP43 directly returning values of the Bessel function of $1^{\text {st }}$ kind and order $\boldsymbol{y}$. Feel free to compare the results.

[^121]:    ${ }^{228}$ Note this follows closely the procedure as described for the Solver above.

[^122]:    ${ }^{229}$ Note this follows loosely the procedures as described for the Solver and Integrator.

[^123]:    ${ }^{231}$ I.e. after some 15 s using the WP43 Simulator (cf. ReM, App. E) on a PC running Windows 10 or some 12 s with FASTFN. It will take 16 minutes on your battery powered calculator, 9.5 ' with FASTFN, some 4' with USB power supply but FASTFA, and 3' with USB and FASTFN.

[^124]:    $232 \ldots$ inspired by STATUS on HP-16C and WP 34S. - Note the font browser FBR shares the same virtual keyboard.

[^125]:    ${ }^{233}$ The timer works exactly as in WP 34S but the display differs. With respect to the timer of the HP-55, there are two differences:

    1. Your WP43 will not take the content of $\mathbf{X}$ when calling TIMER as start time of the timer; start times are supported by RCL here instead (see next page).
    2. Your WP43 will display tenths instead of hundredth of seconds. Reaction times of the hardware do not allow for more precision anyway.
    ${ }^{234}$ On the HP-55, input of a single digit sufficed for storing since only 10 registers were featured for this purpose there. And there was no automatic address increment.
[^126]:    ${ }^{236}$ TVM was launched with HP's very $1^{\text {st }}$ financial 'problem solver', the HP-80 of 1972, and was implemented on each and every HP business or financial calculator thereafter. An early advertising sheet promised 'you can improve and simplify your time-and-money management' applying TVM quickly. 'Random-entry financial keys let you key in problems in any order. And you can change any number at any time.' All this was true for certain - remember it was a time before spreadsheet software became available. It may even hold nowadays to some extent since hardly any modern financial tool solving such problems is more compact and faster at hand than a pocket calculator featuring TVM.

[^127]:    ${ }^{238}$ The examples in this chapter are quoted from the HP-27 OH. Enjoy the typical amounts and interest rates of a time long passed.
    And get used to an amount of 1234.56 US\$ reading $\$ 1234.56$ in the USA (even crying with dB95 ... ummh ... 95 dB will not help); logical coherence reaches maximum.

[^128]:    ${ }^{240}$ Else you shall load all 6 TVM variables except the unknown before solving. (Background: Solving the TVM equation does not change any of the five variables given.)

[^129]:    ${ }^{241}$ Think of a thunderstorm, blizzard, or alike, maybe even fostered by anthropogenic climatic change. Though do not be afraid, this is all fake news created by insane minds according to the greatest president of that blessed nation (feel free to swap adjectives).

[^130]:    ${ }^{242}$ For comparison, 1 kg of TNT releases 4.6 MJ . So 18 MJ are enough for a great blast.
    ${ }^{243} \ldots$ and proper UHV vessels show very low leak rates as well, so the annihilation energy may be released in little quantums over a longer time interval - power supply may be reestablished in time and vacuum pumps operating again.
    For crucial applications, however, uninterruptible power sources based on batteries and/or independent generators are sincerely recommended to be installed locally wherever supplies are threatened by the actual state of public infrastructure being significantly less than great.
    Furthermore, be aware that ordering antipasti and pasta together in an Italian ristorante is strictly at your own risk. You have been warned.
    ${ }^{244}$ The SI system of coherent units of measurement is agreed on internationally and adopted by almost all countries on this planet for long. Advantage: You do not need any

[^131]:    ${ }^{245}$ Generally, science and engineering calculations are executed the easiest (and safest) way using S/ base or derived units without prefixes.
    Some British Imperial project team members, however, have asked for 'kitchen units' (they meant millimeters, kilometers, grams, hectares, liters, and milliliters) appearing explicitly on the SI side of some conversions (applying the rule of three may be too difficult ...). In consequence, you have to watch out now that you do not lose powers of 10 when using such conversions (see example 4 , for instance).

[^132]:    ${ }^{246}$ Sounds like a unit created for the great Alexander visiting Gordion in 333 BC.
    ${ }^{247}$ Such emergencies would never happen, of course, the responsible parties affirmed. Our warmest regards go to Algeria,* Argentina, Armenia, Australia,** Austria, Belarus,*** Belgium, Brazil, Bulgaria, Canada, China, Czech Republic, Finland, France, Germany, Hungary, India, Iran, Israel, Japan, Kazakhstan,*** Kiribati,** Lithuania,*** the Marshall

[^133]:    ${ }^{252}$ Here, upper and lower case letters will be checked, so your WP43 will regard Var1 and VAR1 as being different names. Superscripts and subscripts are not discriminated from normal symbols, so e.g. data1T and data ${ }_{1}{ }^{\top}$ are interpreted as the same name (but the first may ease typing and the latter reading for you).

[^134]:    253 ... unless your WP43 is in alpha input mode. There, Mya will appear instead of MyMenu, and it will stay displayed until another menu is called.

[^135]:    254 This is to assign the functionality of the ENTERT key to another location. ASN $\propto$ turns on AIM. Note spelling ENTER (without $\uparrow$ ) is sufficient here.
    ${ }^{255}$ A (menu) name must be spelled.
    ${ }^{256}$ This may be also or $\square$ plus a key; it may open a (possibly multi-view) menu (e.g. CATALOG'FCNS) containing the functionality to be assigned. In the latter case, press the corresponding softkey therein.
    ${ }^{257}$ This may as well be or plus a key; or it may open a user menu containing the destination to be assigned. In the latter case, press the softkey pointing to this destination.

[^136]:    ${ }^{258}$ I.e. you must not refer to an undefined menu therein - undefined at the time you enter its name. You can refer to any existing menu though.

[^137]:    ${ }^{259}$ Note EXIT and ON are permanently connected.

[^138]:    ${ }^{260}$ On the HP-42S, preview and fallback were absent also for PRGM, ASSIGN, STO, RCL, XEQ, SHOW, and OFF.

[^139]:    ${ }^{261}$ Note RCLCFG will throw an error if you attempt recalling something unless a configuration.

[^140]:    ${ }^{262}$ I.e. either of the number $\boldsymbol{x}$ or of all elements of the matrix $\boldsymbol{x}$.

[^141]:    ${ }^{263}$ There are easier ways to calculate said total amount of rice, but this is the easiest using TVM. Calculating exactly, the inventor should get 18446744073709551615 grains of rice.

[^142]:    ${ }^{264} \mathrm{TN}$ : You can use this picture as a tiny dictionary of some financial terms in American English. Here, the word "with" is abbreviated by "w/" although this does not save any space at all. Abbreviamania ...

[^143]:    ${ }^{266}$ TNG: Policy entspricht hier einer Police.

[^144]:    ${ }^{267}$ The signs or arrow directions are debatable here, in my opinion.

