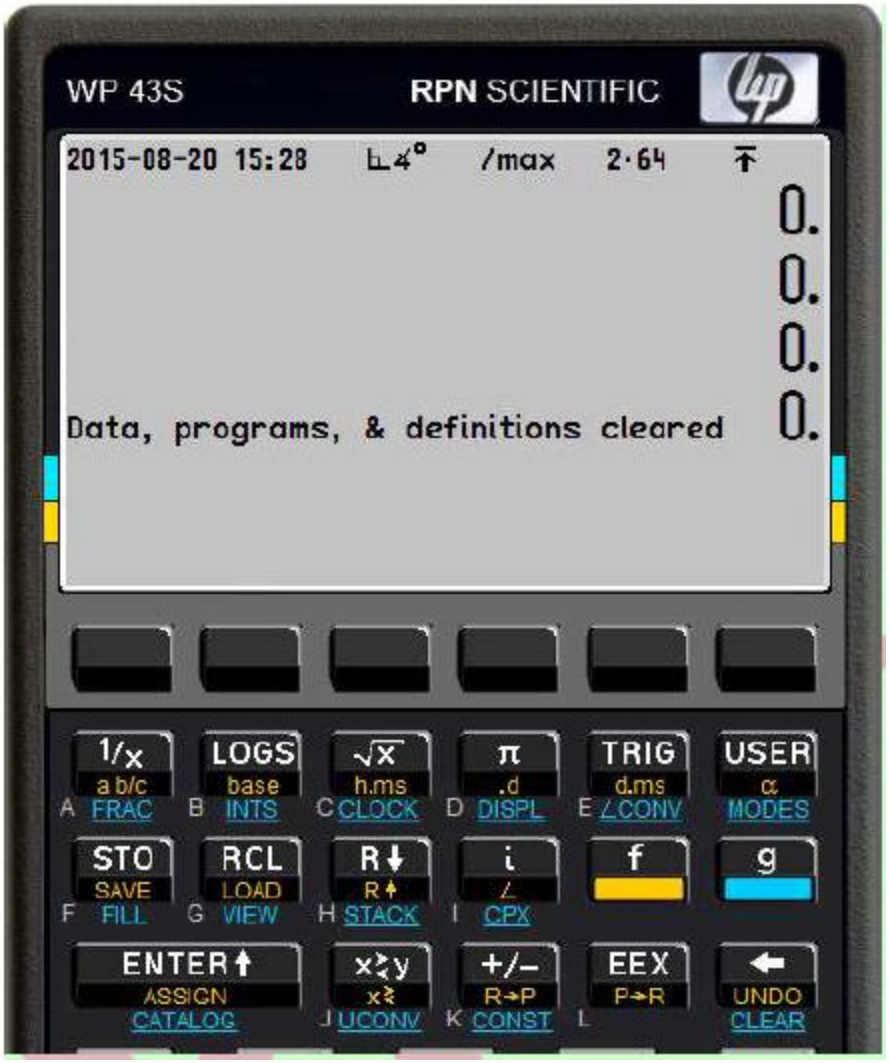
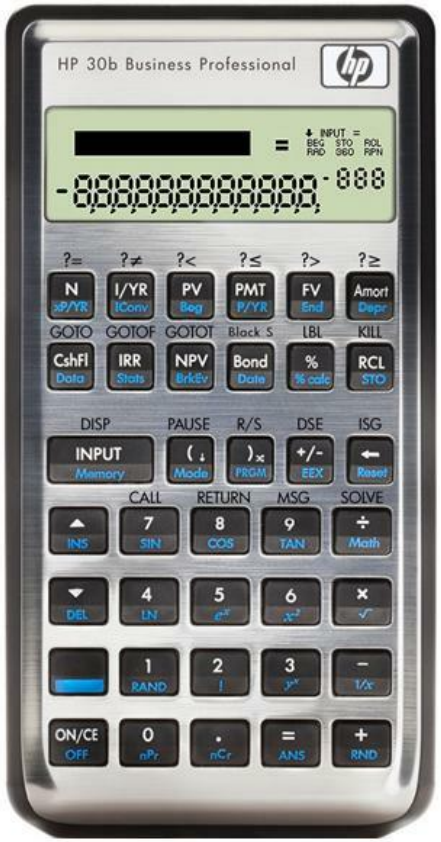
# The WP 43S: A Spec in Need of a Platform

Jake Schwartz



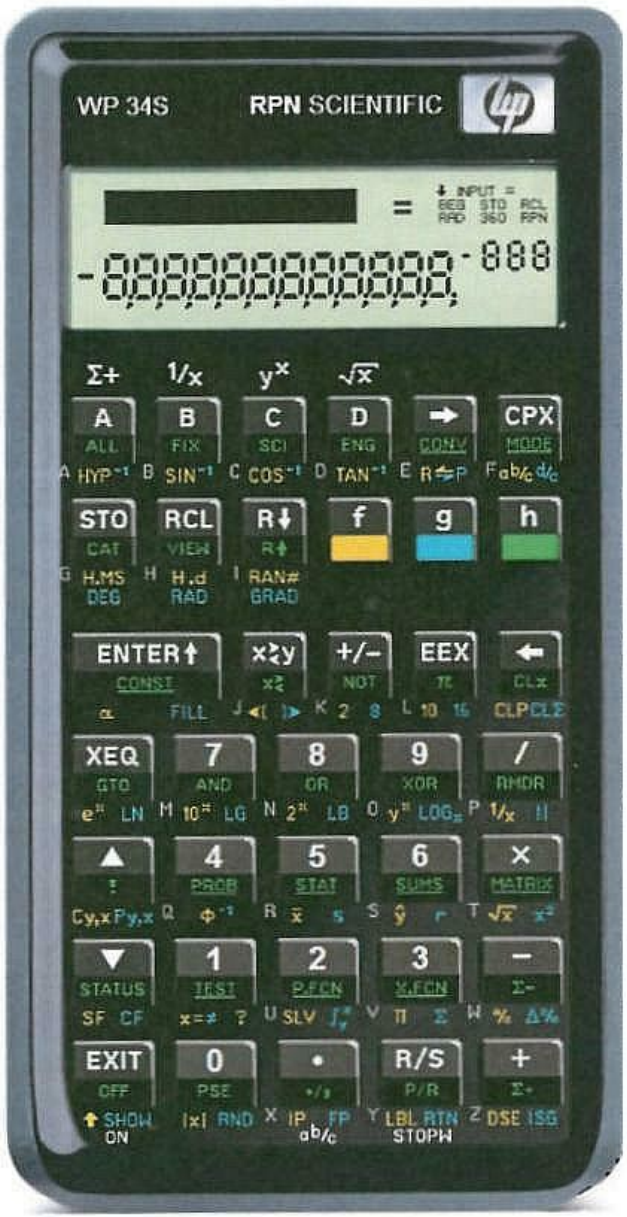
# Back at HHC2010…

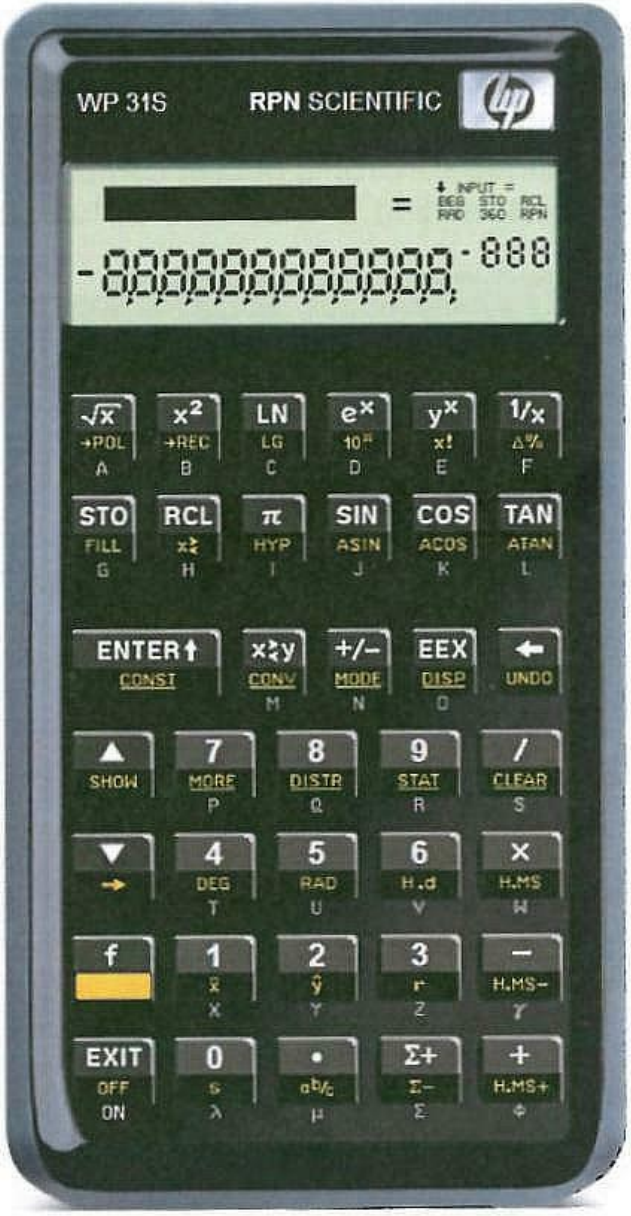
An HP30b Repurposing Project
from Walter Bonin & Paul Dale



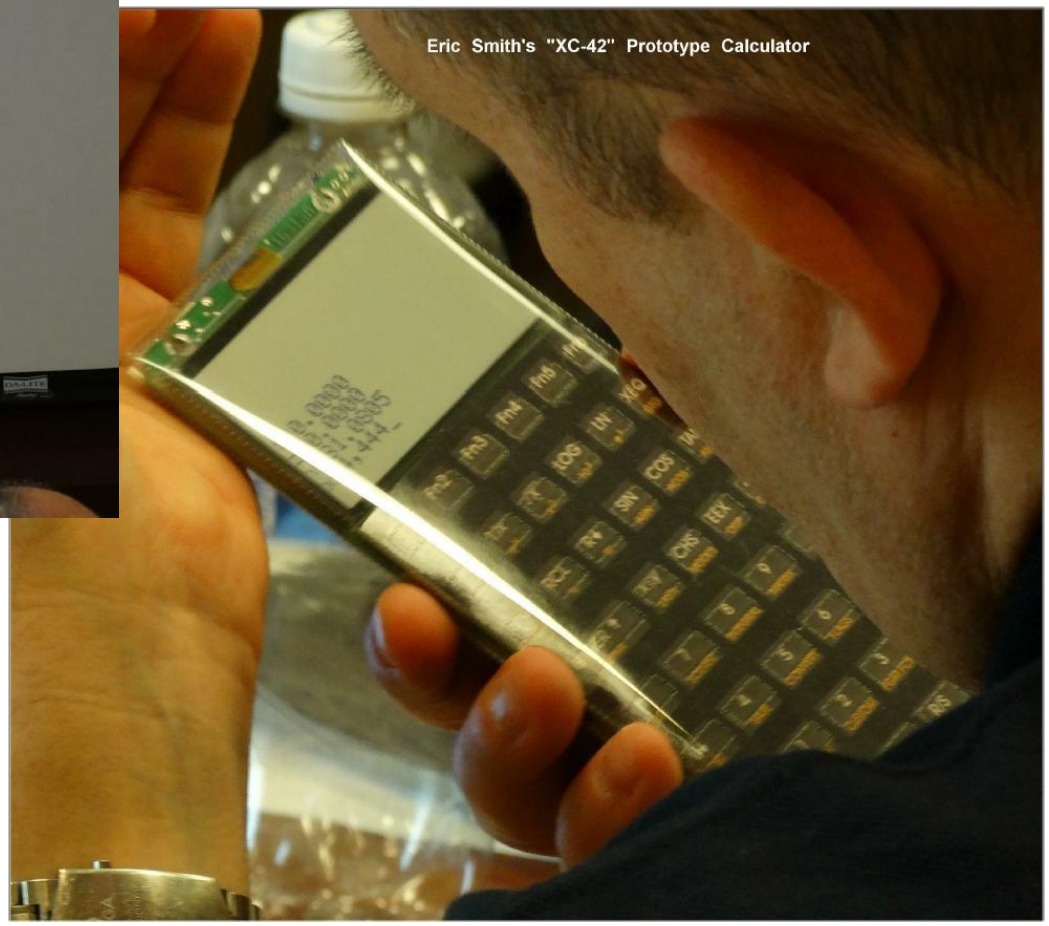

HHC MMX

# Ultimately, the 34S and 31S Emerged

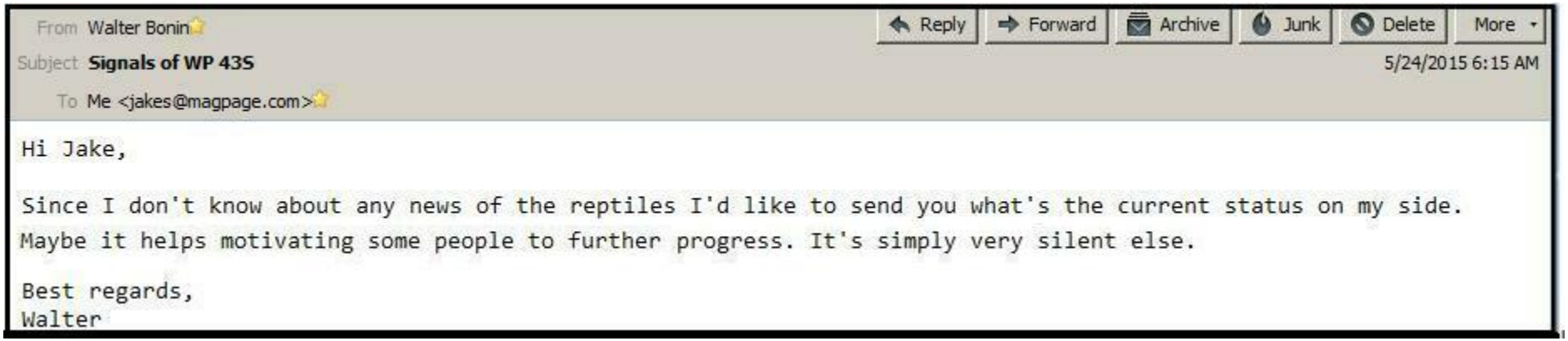


# (Eric Smith at HHC2014)

# Email from Walter this past May



From Walter Bonin
Subject **Signals of WP 43S**
To Me <jakes@magpage.com>

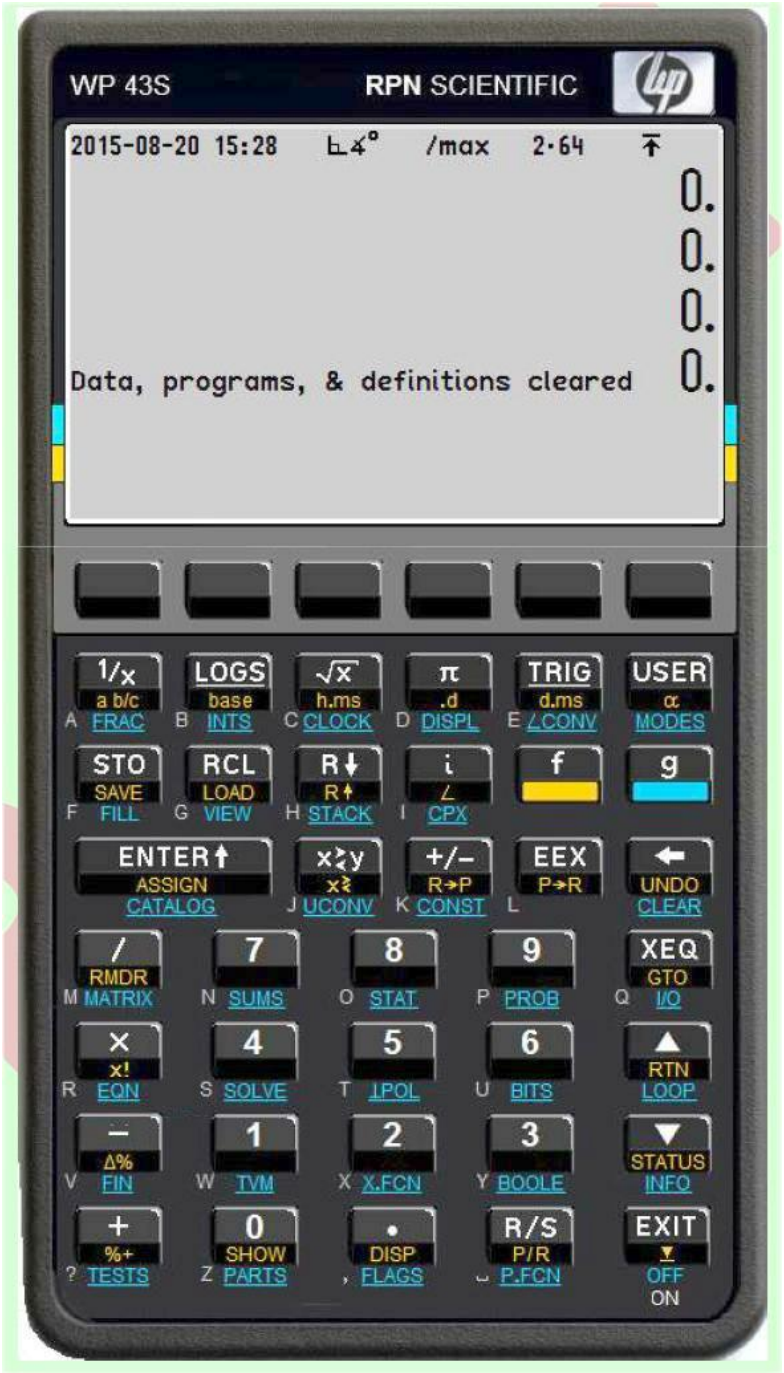
Reply | Forward | Archive | Junk | Delete | More

5/24/2015 6:15 AM

Hi Jake,

Since I don't know about any news of the reptiles I'd like to send you what's the current status on my side. Maybe it helps motivating some people to further progress. It's simply very silent else.

Best regards,
Walter




# Current WP 43S Layout

# Rundown of WP 43S Basic Features

- A solver (root[2] finder) that can solve for any variable in an arbitrary equation.
- A numeric integrator for calculating definite integrals.
- Numeric derivation, programmable sums and products.
- Support of real and complex numbers, fractions, integers, and text strings.
- Matrix and vector operations, including a comfortable Matrix Editor, a solver for simultaneous linear equations, and many other useful matrix functions in real and complex domain.
- Statistical operations, including probability distributions, curve fitting, and forecasting.
- Base conversions and integer arithmetic in fifteen bases from binary to hexadecimal.
- Bit manipulations in words of up to 64 *bits*.
- A stopwatch based on a real-time clock.
- An easy-to-use *menu* system that uses the bottom part of the display to label the top row of keys according to your needs.
- A keyboard layout and *menus* that can be customized by you.
- A *catalog* for reviewing all items stored in memory – be they provided by us or programmed by you.
- Keystroke programming including branching, looping, tests, flags, subroutines, and local data.
- The ability to run programs written for the *HP-41Cx*, *HP-42S*, and *WP 34S* calculators.
- Battery-fail-safe on-board backup memory for all your data.
- An *SD* card slot (allowing e.g. to transfer your programs to a computer, so you can edit and test them there, and return them).
- An infrared port for immediate printing results, calculations, programs, and data using an *HP 82240A/B Infrared Printer*.

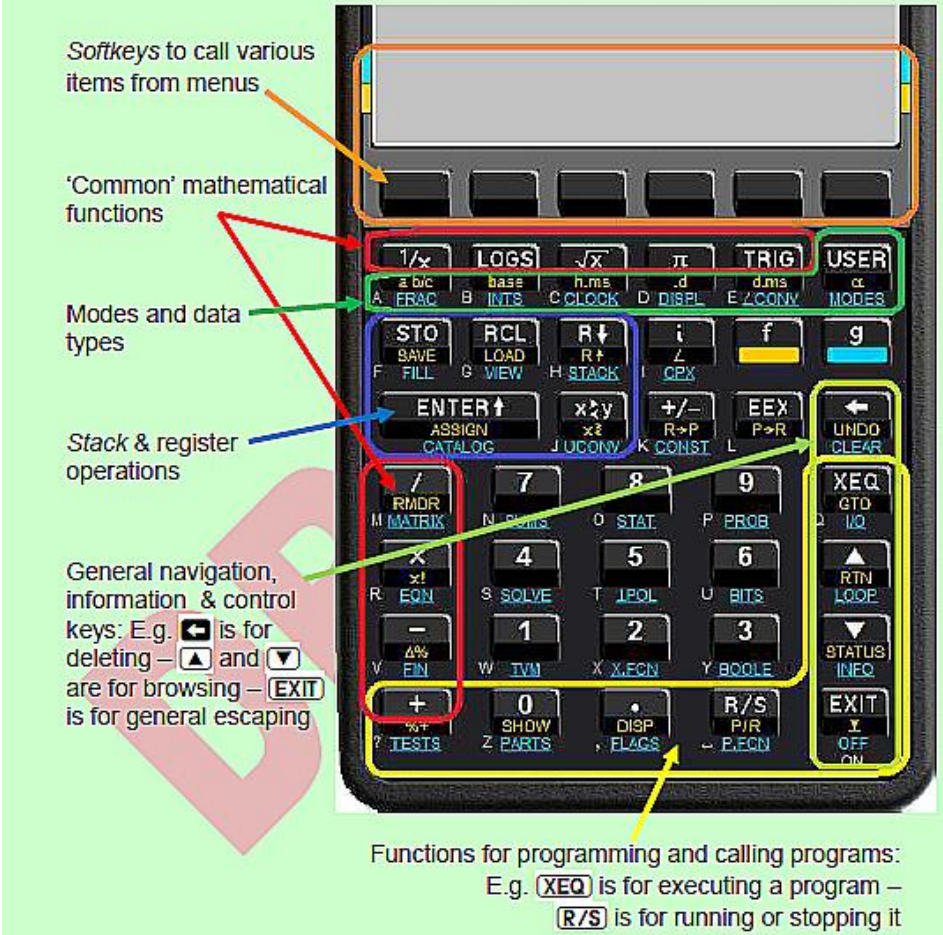
# WP 43S Basic Features, continued

- Full set of scientific functions, including *Euler's Beta* and *Riemann's Zeta*, *Lambert's W*, the *error function*, *Bessel functions* of first kind, *Bernoulli* and *Fibonacci numbers*, as well as the *Chebyshev*, *Hermite*, *Laguerre*, and *Legendre* orthogonal polynomials (no more need to carry heavy printed tables).
- Probability distributions like standard *normal*, *Fisher's F*, *Student's t*, *chi-square*, *Poisson*, *binomial*, *geometric*, *hypergeometric*, *Cauchy-Lorentz*, *exponential*, *logistic*, *Weibull*, *log-normal*, and *Gaussian*.
- Over 50 fundamental physical constants stored as accurate as used today by national standards institutes such as *NIST* or *PTB*, plus a selection of important constants from mathematics, astronomy, and surveying.
- More than 90 conversions, mainly from old *British Imperial* to universal *SI* units and vice versa.
- 4 or 8 *stack* levels and up to 107 global general purpose registers, each taking one object of arbitrary data type.
- As many named variables as memory can hold.
- 112 global user flags.
- Up to 10 000 program steps in RAM, up to 20 000 program steps in flash memory.
- 16 local flags and up to 888 local registers per program allowing for recursive programming.
- A multi-line, high-resolution, alphanumeric display with adjustable contrast, allowing for *menus*, *softkeys*, mathematical symbols, matrix display, Greek and extended Latin letters.

# WP 43S Keyboard

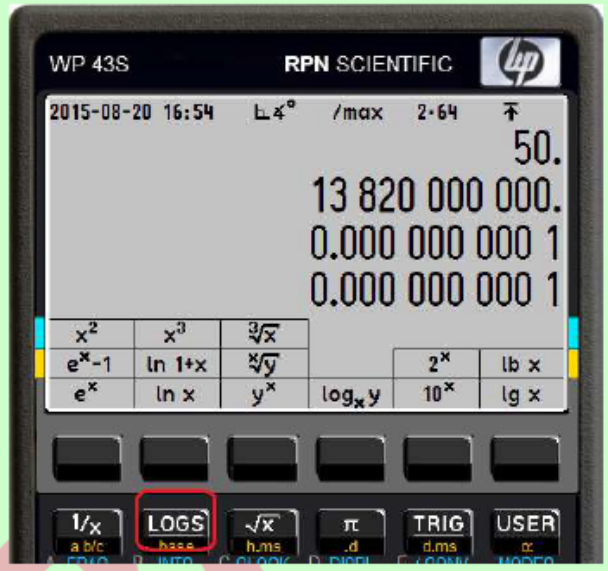
**How the Keyboard is Organized**

You might have recognized the labels on your *WP 43S* are grouped according to their purposes. There are six larger groups:



# Menus and Soft Keys

**Example:**
Pressing [LOGS] will modify the *menu section* (i.e. the lower third of the display) showing the following *menu view*:




As long as this *menu view* above is displayed, simply press

- the first *softkey* for calling [$e^x$], for instance, or
- [f] plus the rightmost *softkey* for [lb x], the inverse of $2^x$.

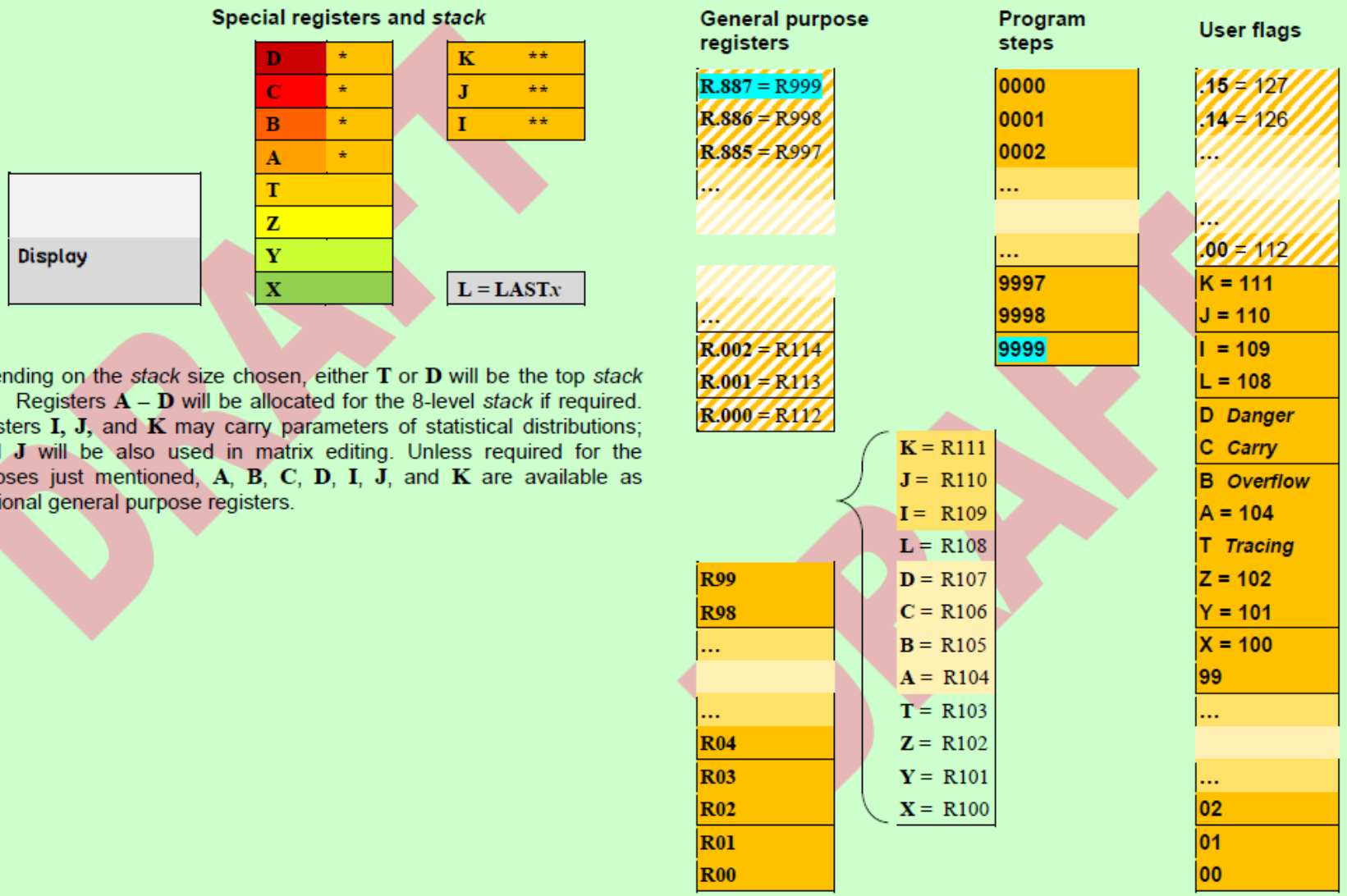
Thus, we may also print **lb x** to refer to [lb x] if we want to emphasize the entire access path to this function in a compact way. In analogy, a blue background may be printed for a function found in the [g]-shifted row (like **∛x**), and a grey background for an unshifted *menu* function (like **ln x**).

Generally, whenever a *menu* is called by accessing a label printed underlined, its top *view* will be displayed in the bottom third of the screen (called the *menu section*). Any such *view* may contain up to 18 *items*: up to six assigned to the unshifted, six to the [f]-shifted, and six to the [g]-shifted top row of keys – thus these keys are called *softkeys*. Note the golden marks on either side of the *LCD* indicating the [f]-shifted row of labels and the blue marks for the [g]-shifted row. For calling a specific *item* contained in such a *view*, use the corresponding key preceded by a *prefix* if applicable.

A predefined *menu* may contain more than just one *view*, i.e. more than 18 *items*. This will be indicated by a dashed line on top of the *menu section* on the screen. Whenever such a larger *menu* shows up, [▼] will advance to the next *view* and [▲] will return to the previous *view* changing the labels displayed.

On the other hand, the *menu section* will stay constant – granting easy access to the functions displayed therein – until this *view* is left (via [▼] or [▲]), this *menu* is left (via [EXIT]), or another *menu* is called. To indicate the access path via a *menu* and the corresponding *softkey*, we will print the formats as explained on previous page in this manual from here on. Note that *submenus* contained in a *menu* will be displayed **inverted** (and are thus printed this way in this manual as well).

# WP 43S RAM Layout




Depending on the *stack* size chosen, either **T** or **D** will be the top *stack* level. Registers **A** – **D** will be allocated for the 8-level *stack* if required. Registers **I, J,** and **K** may carry parameters of statistical distributions; **I** and **J** will be also used in matrix editing. Unless required for the purposes just mentioned, **A**, **B**, **C**, **D**, **I**, **J**, and **K** are available as additional general purpose registers.

# Data Input and Objects

## SECTION 2: DEALING WITH VARIOUS OBJECTS

Your *WP 43S* can do more than just calculating with real numbers. It can deal with integers, fractions, angles, times and dates, complex numbers, vectors, matrices, and even with text strings.

But how shall your *WP 43S* learn about the particular meaning of your input? Some **examples** will explain (showing the lowest numerical display line in the startup default display format after that input):

| Input | Display | Meaning |
|---|---|---|
| 123456 ENTER↑ | $123\,456_t$ | Integer number, see pp. 102ff |
| 123456 base 16 | $12\,34\,56_{16}$ | |
| 1.23.456 ENTER↑ | $1\ ^{23}/_{456}$ [=] | Fraction, see pp. 114ff |
| 12340.56 ENTER↑ | 12 340.56 | Real number, see pp. 80ff |
| 123.4 EEX 567 ENTER↑ | $123.4\times10^{567}$ | |
| 12.3 i 4.56 ENTER↑ | $12.3+i\times4.56$ | Complex number, see pp. 117f |
| 1.23 ∠ .456 ENTER↑ | $1.23\ \angle\ 0.456^r$ | |
| 1.23456 d.ms | 1°23' 45.60" | Sexagesimal angle, s. pp. 97ff |
| 1.23456 h.ms | 1:23:45.6 | Sexagesimal time, see pp. 99f |
| 1234.0506 .d | 1234-05-06 | Date, see pp. 100f |

# Data Objects and Memory Size

## APPENDIX B: MEMORY MANAGEMENT

| | Data type and meaning | Size in *bits* |
|---|---|---|
| **1** | ℤ Integer number of infinite precision | ≥ 64 |
| **2** | ℚ Rational number[132] | ≥ 64 × 2 |
| **3** | ℝ Real number | 64 [133] |
| **4** | ℂ Complex number | 2 × 64 |
| **5** | Angle | 64 |
| **6** | Time | 64 |
| **7** | Date | 64 |
| **8** | Alpha string | M [134] |
| **9** | Real matrix / vector | M |
| **10** | Complex matrix / vector | M |
| **11** | Integer number of finite precision | ≤ 64 |
| **12** | Double precision real number | 2 × 64 |
| **13** | Mode | M |
| **14** | Label | 6 × 16 |

There are twelve data types you know from *Section 2*. Four of them are of 'infinite' size limited by available memory only.

Two more data types are defined for internal use:

- one to store modes and user assignments (see STOM on p. 74),
- one for six-character strings for all kinds of 'labels'.

As mentioned above, any object of any data type will take one storage space only: one register, one *stack* level, or one variable. In consequence, register lengths in your *WP 43S* may vary considerably. You do not have to bother – your *WP 43S* will take care of all the necessary administration.



[132] A rational number consists of two integers of data type 1.
[133] As in the *WP 34S*, standard real numbers feature 64 *bits* and 16 digits precision.
[134] A string of $n$ characters needs $n \times 16$ *bits*.
A vector or matrix needs $n \times 64$ *bits*, with $n$ being the number of elements of it – a complex vector or matrix needs $2n \times 64$ *bits*.
The size of an object of data type 13 will vary according to the number of user assignments being part of it (see *Section 7*).

# Combining data types (add/subtract and multiply)

The matrix overleaf lists in column 1 all the twelve data types your *WP 43S* supports. Furthermore, this matrix shows what will happen if you combine objects of such data types: an object of the data type as indicated in one of the lean columns at right ($y$) plus or minus an object of the data type in column 1 ($x$) will result in an object of the data type at the intersection (thus, wherever a resulting type number is printed at the intersection, the corresponding combination is legal with this operation).

| Data type and meaning $x$ | | $y$ 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ℤ Integer number of infinite precision | 12 | 1 | - | - | 8 | 7 | - | - | 4 | 3 | 2 | 1 |
| 2 | ℚ Rational number (ratio of two integers of data type 1) | 12 | 2 | - | - | 8 | - | - | - | 4 | 3 | 2 | 2 |
| 3 | ℝ Real number | 12 | 3 | - | - | 8 | - | - | - | 4 | 3 | 3 | 3 |
| 4 | ℂ Complex number (in Cartesian or polar coordinates) | 4 | 4 | - | - | 8 | - | - | - | 4 | 4 | 4 | 4 |
| 5 | Angle (in various units) | - | - | - | - | 8 | - | - | 5 | - | - | - | - |
| 6 | Time (in H.MS) | - | - | - | - | 8 | - | 6 | - | - | - | - | - |
| 7 | Date | - | 7 | - | - | 8 | 1[43] | - | - | - | - | - | 7 |
| 8 | Alpha string [44] | 8 | 8 | - | - | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 9 | Real matrix / vector | - | - | 10 | 9 | - | - | - | - | - | - | - | - |
| 10 | Complex matrix / vector | - | - | 10 | 10 | - | - | - | - | - | - | - | - |
| 11 | Integer number of finite precision | 12 | 11 | - | - | 8 | 7 | - | - | 4 | 3 | 2 | 1 |
| 12 | *DP* real number | 12 | 12 | - | - | 8 | - | - | - | 4 | 12 | 12 | 12 |

[43] A date minus a date will result in an integer number of days. Other arithmetic operations on dates are illegal.

[44] In operations on alpha strings, adding corresponds to appending $x$ (converted to a string according to the display format set if necessary) to string $y$. Other arithmetic operations with alpha strings are illegal.

The matrix below shows the data types of products in the same way:

| … times an object $x$ of the data type below returns a product of the data type printed at the intersection. | An object $y$ of data type … 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 ℤ Integer of inf. prec. | 12 | 1 | 10 | 9 | - | - | 6 | 5 | 4 | 3 | 2 | 1 |
| 2 ℚ Rational number | 12 | 2 | 10 | 9 | - | - | 6 | 5 | 4 | 3 | 2 | 2 |
| 3 ℝ Real number | 12 | 3 | 10 | 9 | - | - | 6 | 5 | 4 | 3 | 3 | 3 |
| 4 ℂ Complex number | 4 | 4 | 10 | 10 | - | - | - | - | 4 | 4 | 4 | 4 |
| 5 Angle | 5 | 5 | - | - | - | - | - | - | - | 5 | 5 | 5 |
| 6 Time | 6 | 6 | - | - | - | - | - | - | - | 6 | 6 | 6 |
| 7 Date | - | - | - | - | - | - | - | - | - | - | - | - |
| 8 Alpha string | - | - | - | - | - | - | - | - | - | - | - | - |
| 9 Real matrix / vector | 9 | 9 | 10 | 9 | - | - | - | - | 10 | 9 | 9 | 9 |
| 10 Complex matrix / vector | 10 | 10 | 10 | 10 | - | - | - | - | 10 | 10 | 10 | 10 |
| 11 Integer of finite prec. | 12 | 11 | 10 | 9 | - | - | 6 | 5 | 4 | 3 | 2 | 1 |
| 12 *DP* real number | 12 | 12 | 10 | 9 | - | - | 6 | 5 | 4 | 12 | 12 | 12 |

Please note you have to use DOUBLE to convert an existing number into a *DP* real number (of data type 12) - you cannot enter *DP* numbers directly. *DP* numbers are displayed like

123.456 789 or 123,456 789

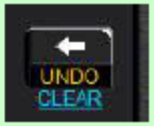
– watch the radix marks. Single precision can be regained letting →REAL operate on a *DP* number.

# Stack UNDO like the WP 31S

**Error Recovery: ← and UNDO**

Nobody is perfect – errors will happen. Stay cool – your *WP 43S* allows you to undo the last command executed, restoring the *stack* exactly as it was before.

1. If you get an **error message** in response to your function call, press ← or EXIT to erase that *temporary message* (cf. p.75), and return to the state before that error happened. Now, do it right!




2. If you have erroneously called a **wrong function**, just press UNDO to undo it immediately. This recalls the *stack* as it was before the last operation was executed.[32] Then resume calculating where you were interrupted.

**Example:**
Assume – while you were watching an attractive fellow student or collaborator – you pressed × inadvertently instead of / in the second last step solving the lengthy formula on p. 48. Murphy's Law! But there is absolutely no need to start that calculation all over again – that error is easily undone as follows:

| | | | | | |
|---|---|---|---|---|---|
| Z | | | | | |
| Y | *numerator* | | *num* | | |
| X | *denominator* | *num × den* | *den* | *num / den* | *correct result* |
| | | × | UNDO | / | √x |
| | Fine so far. | Oops! | | Resume | |

UNDO works for an 8-level *stack* as well. So don't worry – be happy!

# The CATALOG

**One to Rule Them All – the CATALOG**

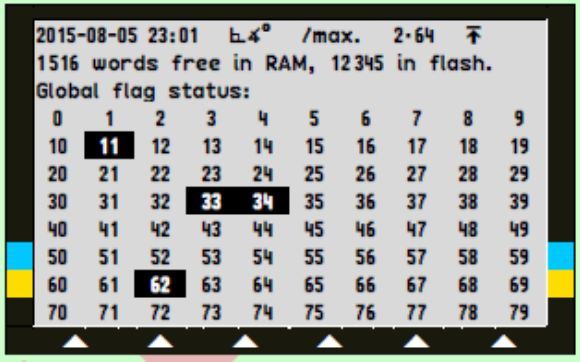
CATALOG calls a very particular *menu*: CATALOG contains all the *items* defined on your *WP 43S* and visible for the user, sorted alphabetically in different branches. Note the contents of the various *submenus* of CATALOG are presented below in reverse order compared to your screen, taking care of your reading habits:

| | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | Remarks |
|---|---|---|---|---|---|---|---|
| CATALOG: | FCNS | PROGS | DIGITS | CHARS | VARS | MENUS | see below |
| FCNS: | °C→°F | °F→°C | $10^x$ | 12h | 1COMPL | 1/x | functions defined, sorted alphabetically |
| | 24h | $2^x$ | 2COMPL | $\sqrt[3]{x}$ | ABS | ACOS | |
| | ACOSH | ac→ha | $ac_{us}$→ha | AGM | ALL | AND | |
| | ... | | | | | | |
| | ... | #B | | | | | |
| PROGS: | RAM | | | | | FLASH | programs (actually global labels) defined |
| RAM: | ... | | | | | | |
| FLASH: | ... | | | | | | |
| DIGITS: | 0 | 1 | 2 | 3 | 4 | 5 | digits defined |
| | 6 | 7 | 8 | 9 | A | B | |
| | C | D | E | F | i | | |
| CHARS: | A...Z | A...Ω | INTα | MATα | Myα | •α | |
| A...Z: | A | B | C | ... | | | plain Latin letters |
| | | | | | | | |
| | ... | Z | | | | | |
| A...Ω: | ... | | | | | | Greek letters, see p. 138 |
| INTLα: | ... | | | | | | extended Latin letters, see p. 139 |
| MATα: | ... | | | | | | mathematical operators and symbols, see p. 140 |
| Myα: | ... | | | | | | see pp. 136f |
| •α: | ... | | | | | | punctuation marks, see p. 140 |

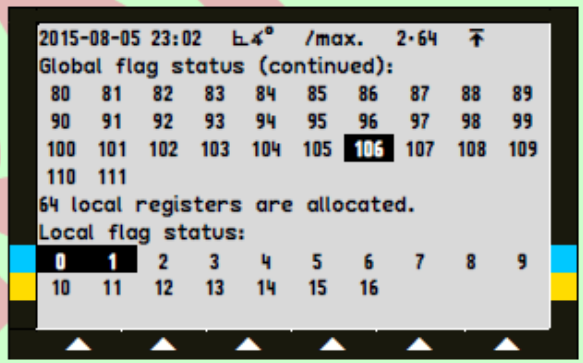
| | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | Remarks |
|---|---|---|---|---|---|---|---|
| VARS: | INTEGS | FRACS | REALS | CPXS | STRGS | MATRS | variables defined of various data types |
| | FINTS | ANGLES | DATES | TIMES | | | |
| ANGLES: | ... | | | | | | |
| CPXS: | ... | | | | | | |
| DATES: | ... | | | | | | |
| FINTS: | ... | | | | | | |
| FRACS: | ... | | | | | | |
| INTEGS: | ... | | | | | | |
| MATRS: | ... | | | | | | |
| REALS: | ... | | | | | | |
| STRGS: | ... | | | | | | |
| TIMES: | ... | | | | | | |
| MENUS: | abcd | A...F | A...Z | A...Ω | A: | Binom: | *menus* defined, sorted alphabetically. See above and below for their contents. |
| | BITS | BOOLE | Cauch: | CFIT | CHARS | CLEAR | |
| | CLOCK | CONV | CONST | CPX | CPXS | DIGITS | |
| | ... | | | | | | |

# Free Memory / Flag Browser

| Keys to press | Contents and special remarks |
|---|---|
| f STATUS | Displays the amount of free memory available and the status of all user flags (inspired by STATUS on *HP-16C* and *WP 34S*) in two views: |




and:

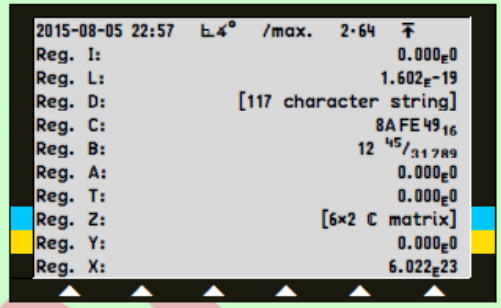



Flags set are displayed inverted. The last four rows of view 2 will only be displayed if local registers are allocated at all.

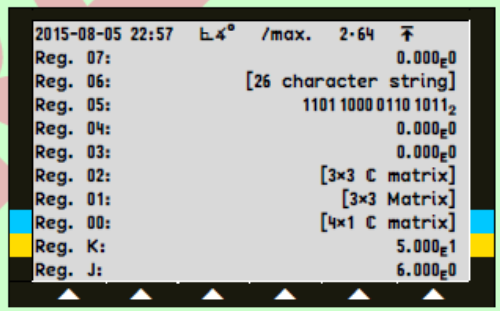
▲ or ▼ toggle between these two views.

# Register Browser

| Keys to press | Contents and special remarks |
|---|---|
| f SHOW | Browses all allocated registers showing their contents. SHOW operates in $\alpha_T$ mode (see pp. 59ff). The first screen you see covers **X** through **I** (the register contents may deviate on your screen): |




| | |
|---|---|
| ▲ | goes up the *stack*, continuing with the other lettered registers, then with **R00**, **R01**, etc. like shown here: |




| Keys to press | Contents and special remarks |
|---|---|
| ▼ | browses the registers going down (if starting with the screen on p. 279) from **R99** to **R00**; then continues with **K, J**, etc. Cf. pp. 57f. |
| . | turns to local registers if allocated, starting with **R.00**. Then, ▲ and ▼ browse local registers up and down – until another . returns to the first screen above. |
| ***nn*** or ***a*** | Input of any legal letter or two-digit number jumps to the corresponding register (see p. 60). |
| ENTER↑ or RCL | recall the register displayed in the lowest line. In *program-entry mode*, they enter a corresponding step RCL … |

# Vector/Matrix Handling pt. 1

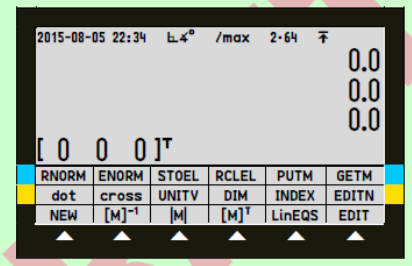
**Example:**

A vector $\begin{bmatrix}4\\-5\\6.7\end{bmatrix}$ and a matrix $\begin{bmatrix}-1 & 12 & 7\\25 & 0 & 3\end{bmatrix}$ shall be entered subsequently. The *stack* shall be clear at beginning.

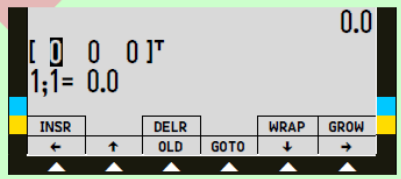
Enter DISPL FIX 1

3 ENTER↑ 1 MATRIX NEW

to initialize the 3D vector (i.e. a 3×1 matrix). You will see the top view of MATRIX displayed at the bottom of the screen now.



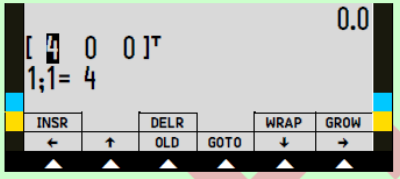

For reasons of space economy, your *WP 43S* displays each vector transposed (thus the superscript T trailing it), i.e. in one row instead of one column on the screen. The vector is initialized with all its components equaling zero. To enter the vector components, press EDIT (the rightmost *softkey*) and the Matrix Editor *menu* will appear at the bottom of the screen:




Note the first 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), so we need two lines for **X**.

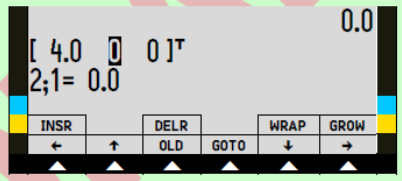
Now press the following key:

4



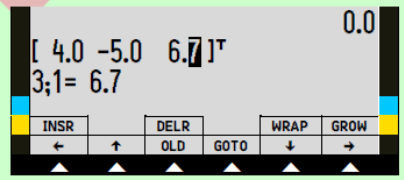

Move the cursor to the next element:

→

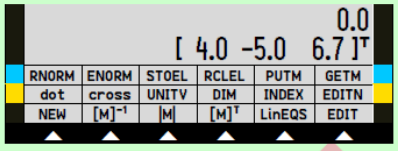



Continue editing:

5 +/- → 6 . 7




EXIT

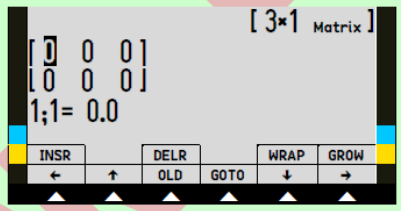



Note EXIT leaves the Matrix Editor returning to the top view of MATRIX, closes input for the object in **X**, and shifts it to the right.

Let us enter the 2×3 matrix now. Initialize it:

2 ENTER↑ 3 NEW

EDIT

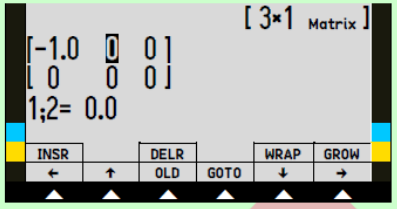



Three numeric lines are required for editing ***x*** now. The '3×1 matrix' in **Y** is the 3D vector we entered before.[82] Note any matrix is shown in this short form in any *stack* level but **X**.

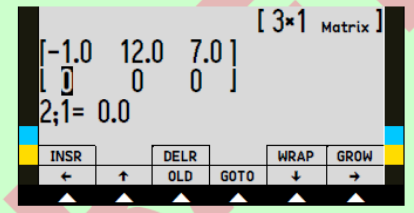
All elements of the new matrix start containing zero again. Its first element is displayed inverted as the first element of the vector was above. It will continue this way:

# Vector/Matrix Handling pt. 2

1 [+/-] [→]

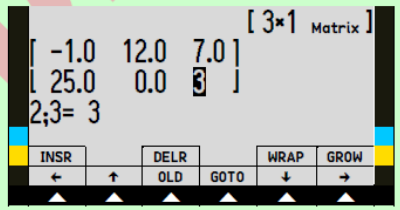



12 [→] 7 [→]

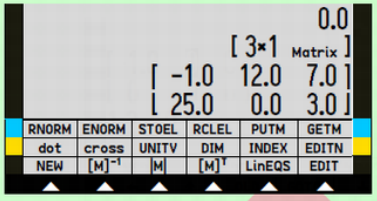



Entering the last [→] moved the cursor from the last element of row 1 to the first of row 2. So you can simply continue

25 [→] [→] 3

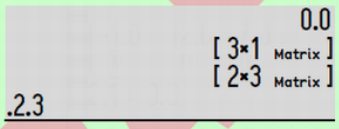



[EXIT]



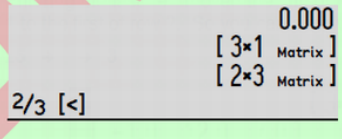

Now also this matrix is closed and ready for calculating. Assume you want to multiply it by $^2/_3$.

[.] 2 [.] 3

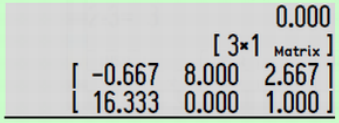



Remember the input line is evaluated not earlier than it is closed. But we want more than just a single decimal displayed in the result:

[DISP] 3




Now press [x] and you will get immediately




which are all matrix elements multiplied by $^2/_3$ at once.

You may store such matrices in any register or variable. So let us store our resulting matrix in **R00** – just press [STO] **00**

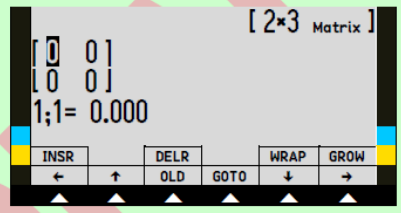
You can also create and fill a matrix directly in a variable (i.e. you do not have to create the matrix on the *stack* and store it afterwards).

**Example:**

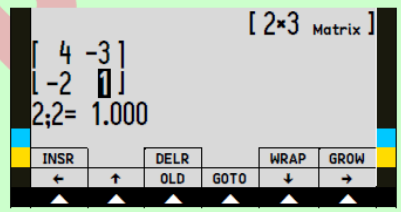
Create a matrix $MA = \begin{bmatrix} 4 & -3 \\ -2 & 1 \end{bmatrix}$ and fill it without recalling it to the *stack*.

2 [ENTER↑] [DIM] [α] **M A** [ENTER↑] Create **MA** as a 2×2 matrix.

[EDITN] MA Open it for editing.

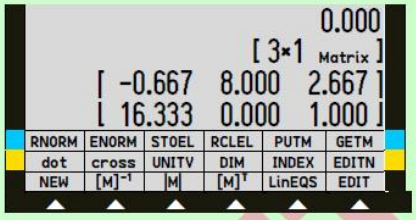



4 [→] 3 [+/-] [→] 2 [+/-] [→] 1



# Vector/Matrix Handling pt. 3

Now, press EXIT and you are done – while the screen looks as before again:




### Vectors and Matrices: Displaying and Editing Larger Objects

Whenever **X** contains a matrix, your *WP 43S* will try to show it completely (i.e. display all its elements in the format you chose for real numbers). Objects in higher *stack* levels will be indicated in a single line (abbreviated if necessary) or will be shifted out of the display window – but $x$ and $y$ will stay on the screen at least.

If space is insufficient for showing the complete matrix in the format chosen, your *WP 43S* will switch to the small font.

**Example** (continued):

RCL 00

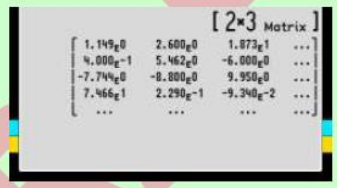
DISP 5


If font switching should not suffice, your *WP 43S* will turn to SCI **3** for the elements of the respective matrix. This allows for showing arbitrary 5×4 real matrices entirely (see *App. D*). If a real matrix exceeds 5 rows, its fifth row will be displayed filled with ellipses (...); if it exceeds 4 columns, its fourth column will be shown filled with ellipses.

**Example:**
Assume a 6×5 matrix

$$x = \begin{bmatrix} 1.1493 & 2.6 & 18.725 & 3 & 9.2 \\ 0.4 & 5.462 & -6 & 95.1 & 51.6 \\ -7.744 & -8.8 & 9.95 & 54.5 & 0.17 \\ 74.66 & 0.229 & -0.0934 & 2 & -3.829 \\ 33.9 & -79.4 & 3.436 & 9.08 & 4.256 \\ 0.0488 & 7 & 5.98 & -0.68 & -22.492 \end{bmatrix}$$

was entered on the present *stack* and is in **X** now. Then the screen will look like this to scale:




Editing such a large matrix will remove also $y$ from the screen until input is closed again. You can browse the entire matrix regardless of its size. 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 3×3 matrix elements including the one selected by the cursor can be seen always at least – this element is also displayed below of the matrix in the format you chose for real numbers. Since the indices of this element are shown there as well you always know where you are.

### Vectors and Matrices: Complex Stuff

Your *WP 43S* supports also complex vectors and matrices, i.e. matrices containing complex elements. They are created and initialized like real objects via NEW or DIM as explained above. Or you can recall a real matrix; then it may be filled with one or more complex numbers – thus becoming a complex matrix – and can be stored at the same or another place.

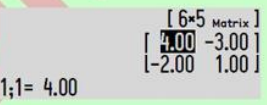
**Example** (continuation of p. 126):
Create and store a complex matrix $\begin{bmatrix} 5+8i & \pi i \\ -2 & 4-3i \end{bmatrix}$.

Remember we have created a 2×2 matrix just a few pages ago. So it is most easy to recall it for using it as a template:

RCL VARS MA

DISP 2 since we will not need five decimals for that.

MATRIX EDIT




We can now simply enter the new elements there as we have done before:

5 . 8 → . π → → 4 . 3 +/-

EXIT

STO 01

We can now simply enter the new elements there as we have done before:

5 . 8 → . π → → 4 . 3 +/-

EXIT

STO 01

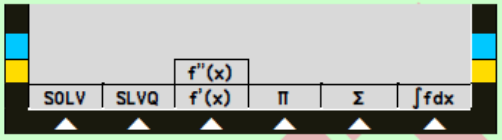

Since we stored the complex matrix at a new location, the real matrix **MA** is not affected at all.

Compare pp. 114f for the input and formatting of complex numbers. Everything else works as it does for real matrices. You see complex vectors and matrices are no complex topic for you with your *WP 43S* at all.

# Advanced Problem Solving

## SECTION 4: ADVANCED PROBLEM SOLVING

There are some powerful commands provided for computing programmable sums and products, for solving equations, for computing finite integrals as well as first and second derivatives. All are contained in SOLVE. Pressing SOLVE results in

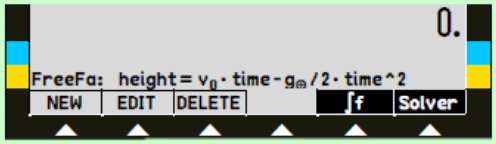



The quadratic solver SLVQ is described in the *IOI* on p. 224. The remaining commands Σ, Π, SLV, ∫, f′(x), and f″(x) are explained below. Integrating and solving equations may also be reached through EQN. See below for details.

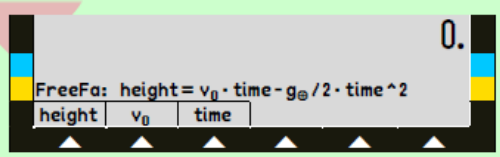
### Programmable Sums

### Programmable Products

### Equations




### The Interactive Solver



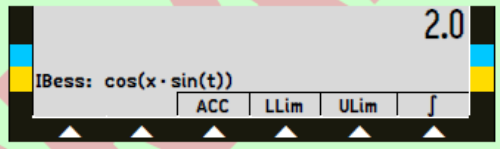

### Numeric Integration


You can enter values of any variables (i.e. integration constants) you already know by pressing the respective *softkeys* now, e.g.

**2** **x**

(For recalling such an integration constant, just press f before the respective *softkey*.)

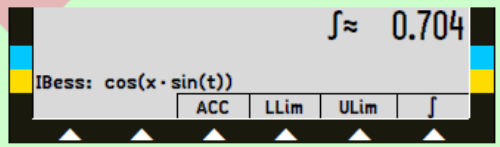
Then select the variable of integration by simply pressing **t** here (there must not be any numeric input heading **t** ). The *menu* will change:



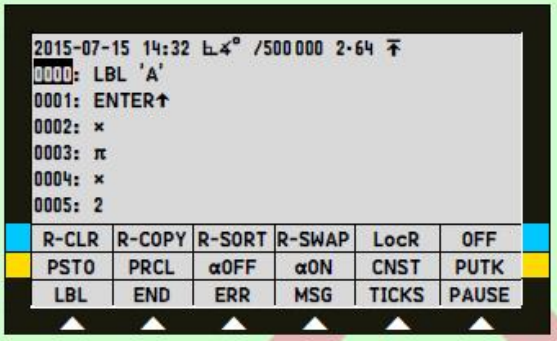

We want to see the result with three decimals. Thus we enter

DISP **3** **.01** **ACC** **0** **LLim** π **ULim**

and start integrating by pressing ∫ . Your *WP 43S* will return:



# Entering a Program




## Recording a New Routine

Whenever you want to enter a new routine, switch to *program-entry mode* (unless you are already in) using P/R and start with pressing GTO . . . These four 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 else.

Start with LBL giving your routine a name (it may be six characters long). Then press the keys as you would do in manual problem solving (cf. p. 23). Each new step will be inserted right after the *current step* as defined above.

You find

- XEQ for calling or executing a specific routine,
- GTO for Going TO it (i.e. positioning the program pointer to the respective LBL step),
- ▲ and ▼ for browsing program steps,
- RTN for returning from a routine called,
- R/S for Running or Stopping programs,
- P/R for toggling *Program-entry* and *Run mode*, and
- EXIT for leaving *program-entry mode* (returning to *run mode*)

bottom right on your *WP 43S* as shown here, together with the *menus* for *TESTs*, *FLAGS*, and *PARTS*. Further programming functions like LBL mentioned above are collected in *P.FCN*. Note that ▲ and ▼ are not programmable but useful nevertheless (see also p. 151). See the next paragraphs and the *IOI* (on pp. 171ff) for more about all these commands. And remember the final RTN in your routine.

# “Items” Index

## SECTION 5: INDEX OF *ITEMS (IOI)*

All the *items* provided on your *WP 43S* (more than 850) are listed below with their *names* and the keystrokes necessary to call them. Names printed in **bold** face in this table belong to operations directly accessible on the keyboard; the other *items* may be picked from *menus* (see pp. 255ff) or *catalogs*.[104]

Sorting in this index, in CATALOG, and CONST is case insensitive and works in the following order:

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

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

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

Operations working with the accumulated statistical data are marked light blue.

Some 300 functions featured in your *WP 43S* are new compared to *HP's RPN* calculators.[105] Operations carrying familiar names but deviating in their functionality from previous *HP RPN* calculators or the *WP 34S* are marked light red.

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

[105] We did not compare with the *RPL* calculators of the last decades or the *HP Prime*. They are exceeding the realm of shirt pocket calculators.

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

(0) represents functions without any effects on the *stack* (e.g. mode setting functions);

(1) is for *monadic functions*,

(2) for *dyadic functions*, and

(3) for *triadic functions* as defined on pp. 38ff;

(-1) stands for functions pushing one object and

(-2) for functions pushing two objects on the *stack*.

| Name | Keystrokes | Remarks (see pp. 171ff for general information) |
|---|---|---|
| °C→°F<br>°F→°C | g UCONV f °C→°F<br>etc. | See pp. 269ff. |
| $10^x$ | LOGS $10^x$ | (1) {1, 2, 3, 4, 9*, 10*, 11} Returns $10^x$. |
| 12h | g CLOCK ▼ f 12h | (0) Sets 12h time display mode: e.g. 1:23 will become 1:23 AM, 23:45 will become 11:45 PM. Cf. 24h. |
| 1COMPL | g INTS ▼ 1COMPL | (0) Sets 1's complement mode (see p. 99). |
| 1/2 | g CONST 1/2 | (-1) See pp. 260ff. |
| 1/x | 1/x | (1) {2, 3, 4, 9*, 10*} Inverts the number in $x$. |
| 24h | g CLOCK ▼ f 24h | (0) Sets 24h time display mode. Cf. 12h. |
| 2COMPL | g INTS ▼ 2COMPL | (0) Sets 2's complement mode (see p. 99). |
| $2^x$ | LOGS f $2^x$ | (1) {1, 2, 3, 4, 9*, 10*, 11} Returns $2^x$. |
| $\sqrt[3]{x}$ | LOGS g $\sqrt[3]{x}$ | (1) {1, 2, 3, 4, 9*, 10*, 11} Returns the cube root of $x$. |
| a | g CONST a | (-1) See pp. 260ff. |
| $a_0$ | g CONST $a_0$ | |
| ABC | g CATALOG CHARS ABC | *Submenu*. See p. 66. |

# Anticipated Prototype Hardware Specs

## APPENDIX A: HARDWARE

Technical Specifications:

| | |
|---|---|
| Overall dimensions: | 66 *mm* × 155 *mm* × 14 *mm* for the prototype mylar folded case |
| Mass with batteries: | 100 *g* |
| *LCD* dimensions: | about 58.8 *mm* × 35.3 *mm* visible area, 400 × 240 quadratic pixels monochrome (see App. D) |
| Processor: | Silicon Labs (formerly Energy Micro) EFM32GG380F1024 |
| Memory: | 1 *MB FM*, 128 *kB RAM* (see App. B), 128 *kB* internal non-volatile data memory. |
| Power supply: | 3 *V* by 2 *CR2032* coin cells or 2 AAA or 2 AA cells (to be decided). Expected battery life time exceeds 6 months of typical use. |
| I/O: | 1 card slot for a micro SD card (up to xxx *GB* can be addressed), infrared printer port. |
| Self-test: | initiated by xxx |
| Overlays: | For the prototype edition, you can install overlay sheets with your user keyboard layouts printed on them. The overlay slides into the case through the battery door. See the example drawing below for the dimensions of such sheets. |
| | On the final edition, slots are provided for fixing overlay sheets with your user keyboard layouts printed on them. |



| | | L, mm | W, mm | D, mm |
|---|---|---|---|---|
| hp 50g | | 184 | 88 | 25 |
| hp Prime | | 182 | 86 | 14 |
| WP 43S | | 155 | 66 | 14 |

The ’43S is not quite a “battleship”

# Overall Manual Contents (abridged)

## TABLE OF CONTENTS

# We Want Our 43S!!