PROGRAMMABLE CALCULATOR

rPn- 1250

User Manual

Programmable calculator RPN-1250

User manual

Calculator version 4.1 © Benoit Maag

June 2024

HHC 2018 : Repurposing Old TI Calculators

https://www.youtube.com/watch?v=mxwn67G2P60

Benoit Maag :

Repurposing a TI-1250 to create an RPN-1250 calculator

Sommaire

1. Keyboard layout

The **RPN-1250** keyboard has 24 keys.
The basic functions of each key are written in white below the key concerned.
The functions writed in orange above each key, on the left, are activated by preceding the press of the key concerned by a press of the **F** key.
The functions writed in blue above each key, on the right, are activated by preceding the press of the key concerned with two presses of the **F** key, i.e. the equivalent of **G**.

GSB RTN GTO	LBL SST BST	SV LD SST	PSE VIEW R/S
F G	IP FP PRGM	EEX NOP CHS	FIX SCI ÷
SIN -1 7	COS -1 8	TAN -1 9	STO RCL X
ln e^x 4	LOG 10^x 5	√x X^2 6	x⇆y R↓ −
1/x y^x 1	in → ←mm 2	mi → ←km 3	lb → ←kg +
CLR ALL ⇦	π →Rad 0	°F → ←°C .	LASTX *Esc.* ENTER

2. Overview

Based on a Texas Instruments **TI-1250** calculator, the **RPN-1250** calculator has a 64 KB Microchip ***PIC 18F2680*** flash chip with an 8-digit, 7-segment ***MAX7219*** LED driver. The software is programmed in **C** with the Microchip ***MPLAB X IDE***.
The **RPN-1250** calculator has 98 program steps and 20 registers in volatile memory. These programs and registers can be saved, in programming mode, in three different constant memory areas.

Features :

- 4-level scientific RPN stack with conversions,
- 20 memories (0 to 9 and .0 to .9) with arithmetic store and recall,
- Possible backup of the stack and registers in the "constant" flash memory of the PIC,
- 98 programmable steps (like the HP-29C) with alphanumeric display of program,
- 3 program saving areas in the "constant" flash memory of the PIC,
- 20 labels (0 to 9 and .0 to .9),
- subroutines (GSB, RTN),
- 12 possible tests (X=0, X<>0, X=y, X<>y, ...),
- PAUSE and VIEW functions,
- step by step execution (SST),
- conversions (in<>mm, mi<>km, lb<>kg, °F<>°C),
- Speed ??approximately 8 times higher than an HP-41C...

Deviations from the HP-29C calculator :

- No increment or decrement instructions (ISZ, DSZ)
- No indirect addressing
- No absolute value (ABS) [replaceable by x<0 ? CHS]
- No polar/rectangular conversions

4. How to use the calculator

The **RPN-1250** calculator is equipped with a single-line, 8-character (alpha)numeric LED display.
Power is provided by a 9 Volt 6LR61 battery and the calculator turns on and off using the switch located on its left :

- **ON** : positioned at the top,
- **OFF** : positioned at the bottom.

<u>3 modes of use are possible :</u>

- **"Execution" mode**

is the mode in which the calculator is used to make calculations and conversions or launch the execution of the program loaded in volatile memory.
The stack and the used registers, being in volatile memory, are reset each time the calculator is turned on, but can be kept in constant memory.

[F] [SV] allows saving of the stack and registers in constant memory.

[G] [LD] allows you to reload the stack and registers from constant memory.

The execution of a program loaded in volatile memory is done
- either by positioning at the start of memory (step 00) via [RTN] then [R/S] to launch execution (after entering the data required by the program)
- either by positioning at the starting label of the program via [GTO] following the label concerned (0 to 9 or .0 to .9) then [R/S] to launch the execution (after entering the data required by the program)
- either by directly launching the program via [GSB] followed by the label concerned (0 à 9 ou .0 à .9) (after entering the data required by the program).

A program can also be executed step by step to check that it is functioning correctly.
The launch is done in this case by positioning in the program either by [RTN] or by [GTO] and using [SST] to advance step by step.

- **"Program" mode**

is the mode in which programs are entered and can be edited.

- **"Esc." mode**

⇒ is the mode in which the brightness can be adjusted. [+] [−]
⇒ a display test can also be performed. [÷]
⇒ and all alphanumeric characters can be viewed. [CHS]

5. Programming

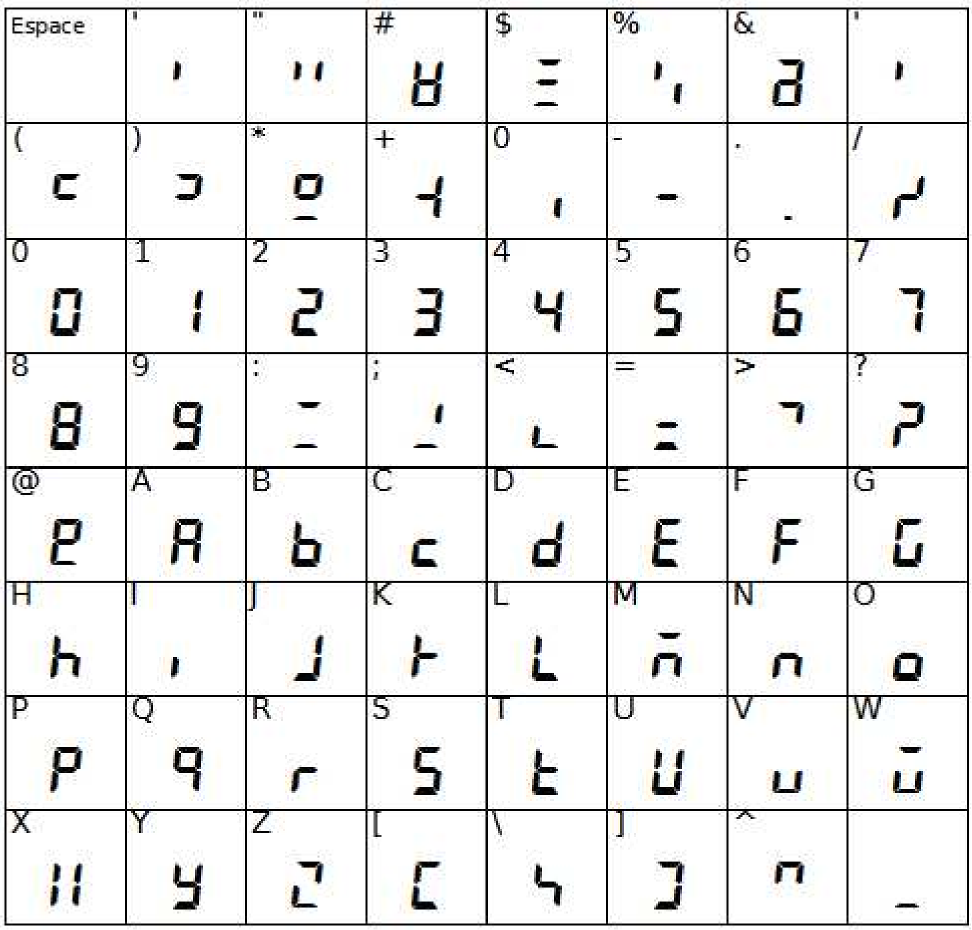
Writing a sequence of keys into the program memory is called a program.
The program turns the calculator into a powerful tool.

The program memory consists of 4 independent program areas :

- Working memory which is memory for input and program execution.
  This memory is volatile and is erased when the calculator is turned off..
- 3 backup memories in which working memory can be saved.
  These 3 memories are constant and preserved when the calculator is turned off.

Each program area contains 98 program steps, so a total of 294 program steps are available in reserve.
20 labels, numbered 0 to 9 and .0 to .9, can be used.

Programming mode is activated with the key PRGM.
The content of the program is displayed alphanumerically on the display line in the form of step number followed by the text of the instruction.



Keys and functions useful for programming :

[SST] (*Single Step*) Increments the program pointer by 1 ("next step").

The **SST** key can also be used in “execution” mode.

In this mode, after pressing **SST**, the code of the current instruction is executed and the program pointer is positioned on the next step. (step-by-step testing of a program)

[BST] (*Back Step*) Decrements the program pointer by 1 ("previous step").

[←] (*Delete*) Deletes the instruction at the current position of the program and displays the next part of the program.

no key (*Insert*) The insertion of an instruction is done automatically **after** the current instruction with offset of the following instructions and therefore does not require a specific insertion key.

[PRGM] (*Program*) In “execution” mode **PRGM** switches to programming mode.

In programming mode, **PRGM** exits program edit mode and returns the calculator to “execution” mode.

[GTO] (*Go To*) The **GTO** instruction is normally used to move the program pointer to the specified label 0 to 9 or .0 to .9.

In “programming” mode, the instruction cannot be used to move the pointer, because the corresponding code would be stored in the program.

You must therefore exit programming mode by pressing [PRGM] , perform a [GTO] jump to the specified label 0 to 9 or .0 to .9 and return to programming mode by pressing [PRGM] .

[RTN] (*Return*) Return order after the execution of a subroutine called by **GSB**.

RTN is placed at the end of the subprogram.

But **RTN** can be used in "execution" mode to return the program pointer to address 0.

[R/S] (*Run/Stop*) Starts the program or stops the program (used in "execution" mode).

Means “end of program” in programming mode.

6. Keys and instructions

Each instruction has a step address followed by a title corresponding to a sequence of presses on one or more keys to express it.

00…09 [0] [9] 0…9 - Numbers

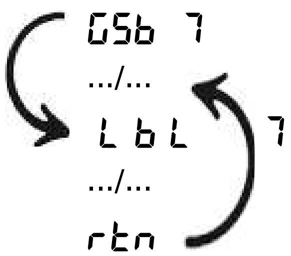
Base digits in the range 0 to 9 are used to enter numbers.
They are also used to enter the mantissa of a number, enter the exponent, a memory register number, or a tag number.

The numbers are stored in the program with a code from 0 to 9 without leading zero.

1.1 [F] [GSB] GSb - Subroutine call

The **GSB** (GoSuB) key is used to call a subroutine using as parameter the numerical code from 0 to 9 or .0 to .9 of the called label.
If the **GSB** instruction is used in "execution" mode, the subroutine is executed immediately.
A subroutine ends with the **RTN** instruction to ensure the subroutine returns to the calling program.

Example :



[G] [RTN] rtn - Return from subroutine

A subroutine called via **GSB** ends with the **RTN** instruction which ensures the subroutine returns just after the calling **GSB** instruction.
In "execution" mode, **RTN** positions the program pointer on step 0.

GTO — Gto - Go to a label

GTO allows you to perform an unconditional jump in a program. It has as parameter a numerical code from 0 to 9 or .0 to .9 corresponding to a label (**Lbl**) of the program.
When the **GTO** instruction is used in "execution" mode, the program pointer is positioned on the corresponding label.

Example :

Gto .8
.../...
LbL .8

1.2 F LBL — LbL - Label

The **Lbl** instruction can be used to mark the start of a sequence in the program as a label.
20 labels can be used from **Lbl** 0 to **Lbl** 9 and from **Lbl** .0 to **Lbl** .9.
The label number is specified as numeric parameter 0 to 9 or .0 to .9 of the **Lbl** instruction.
You can jump to the labeled location in the program using the **GTO** jump instruction or the **GSB** subroutine call instruction..

G TST — 11=0 ... 11L=y - Tests

The test instructions allow you to compare the **X** register (display contents) with either the value zero (0) or with the **Y** register.
If the test is satisfied, the instruction following the test instruction is executed; otherwise, the command following the test is ignored and execution continues after.

The comparative tests are :

comparison between X and zero

G TST 7	**X=0**	11=0
G TST 8	**X<>0**	11L 70
G TST 4	**X>0**	1170
G TST 5	**X>=0**	117=0
G TST 1	**X<0**	11L0
G TST 2	**X<=0**	11L=0

comparison between X and **Y**

G TST . 7	**X=Y**	11=y
G TST . 8	**X<>Y**	11L 7y
G TST . 4	**X>Y**	117y
G TST . 5	**X>=Y**	117=y
G TST . 1	**X<Y**	11Ly
G TST . 2	**X<=Y**	11L=y

BST bSt - Back step

The BST (Back Step) key in programming mode returns to the previous step.

F SV Su - Save in constant memory

In "execution" mode the **SV** (Save) function allows saving in constant memory (or continuous memory) of registers and the stack.
In programming mode, programs can be saved in 3 constant memory zones of your choice by specifying 1, 2 or 3 behind the **SV** command.

G LD Ld - Load from constant memory

In “execution” mode, the **LD** (Load) function allows you to reload the registers and the stack saved in constant memory (or continuous memory).
In programming mode, a program can be reloaded from one of the 3 constant memory zones of your choice by specifying 1, 2 or 3 behind the **LD** command..

SST SSt - Step forward

The **SST** (Single Step) key advances one step in programming mode.

In "execution" mode, the program instruction, on which the pointer is positioned, is executed, allowing the program to be executed step by step for debugging purposes.

Warning: in this case of step-by-step testing of the program if a subroutine is called, the return of the subprogram (**RTN**) works as in “execution” mode and returns to step 0.

1.4 F PSE PAUSE - Pause

The **Pause** command stops program execution briefly and displays the contents of the **X** register for the duration of the pause.

G VIEW ui E - View register

In a program, the **VIEW** command displays the contents of a register without stopping program execution.

R/S rrS - Run / stop program

The **R/S** (Run/Stop) key can be used to start or stop a running program.

At startup, the program starts executing from the current program step (the current address can be found by switching to **PRGM** programming mode).

In programming mode **R/S** indicates stopping of the program.

2.1 [F] Shift key F

The **F** key is used to change the meaning of the next key to an alternate function.
After pressing **F**, the alternative function (in orange) of the next key is then executed.
A second press on **F** activates the alternative function **G**.
A third press of **F** cancels the previous presses of **F**.

The title of the **F** key is not recorded in the program; it is the alternative title of the following key which is then displayed.

Example:

[ln] is actually obtained by pressing [F] [4] and displays Ln

[G] Shift key G

The **G** key is used to change the meaning of the next key to an alternate function.
This **G** function is already an alternative function of the **F** key.
After two successive presses of **G**, the alternative function (in blue) of the next key is then executed.
A third press on **G** cancels the previous presses on **G**.

The name of the **G** key is not recorded in the program, it is the alternative name of the next key which is then displayed.

Example:
[e^x] is obtained by [G] [4] but in reality by pressing on [F] [F] [4]

2.2 [F] [IP] int - Integer part

The **IP** (INTeger Part) key is used to remove digits after the decimal point from the number and to reduce the number to an integer.
The function has the same meaning as rounding to zero.

Example :

[2] [.] [3] [IP] ... integer part of 2.3 [2]

[2] [.] [3] [CHS] [IP] ... integer part of -2.3 [-2]

G FP FrAc - Fractional part

The **FP** (FRACtional Part) key is used to remove the digits before the decimal point from the number and to reduce the number to a fractional number.

Example :

2 . 3 FP ... partie fractionnaire de 2.3 [.3]

2 . 3 CHS FP ... partie fractionnaire de -2.3 [-.3]

PRGM PrGm - Programming

PRGM enables or disables programming mode.

2.3 F EEX EEx - Entering an exponent of ten

The **EEX** function allows you to enter a number multiplied by a power of 10.
If the key is pressed while entering a number, that entry displays the exponent at 00 while awaiting entry.
If the **X** register (display) is at zero, pressing **EEX** gives 1 as the **X** register value and displays the exponent at 00 while waiting for its entry.
The exponent can be negative using the **CHS** function.

In the event of overflow, the calculator displays ouErFLow

G NOP noP - No operation

The **Nop** (No Operation) command is an “empty” command that does not perform any operations. It is only used to fill an unused step in the program.

CHS chS - Change of sign

The **CHS** key changes the sign of the number on the display.
Its use while entering the exponent of a number (power of ten) changes the sign of this exponent.

2.4 [F] [FIX] Fix - Number of decimal places

Using the **Fix** key, the number displayed on the screen is rounded to the specified number of decimal places.

The number 0 to 6 is entered as a parameter, representing the number of decimal places after the decimal point : 0 to 6.

In rounding mode, the number is padded from the right with zeros, up to the specified number of decimal places.

Rounding only affects the number display. Internally, the number (**X** register) continues to be memorized in full.

The rounding mode set also affects how very small numbers are displayed.

If the number of decimal places to display only concerns zero decimal places, the number is displayed in negative powers of 10.

Examples :

Keys		Display
[0] [.] [0] [0] [1] [2] [3] [FIX] [6]	displays	0.001230
[FIX] [3]		0.001
[FIX] [2]		1.23 -03
[FIX] [5]		0.00123

[G] [SCI] Sci - Scientific notation

Using the **SCI** key, the number displayed on the screen is displayed as a power of ten rounded to the specified number of decimal places. The number 0 to 3 is entered as a parameter, representing the number of decimal places after the decimal point 0 to 3.

In rounding mode, the number is padded from the right with zeros, up to the specified number of decimal places.

Rounding only affects the number display. Internally, the number (**X** register) continues to be memorized in full.

Examples :

Keys		Display
[0] [.] [0] [0] [1] [2] [3] [FIX] [6]	displays	0.001230
[SCI] [3]		1.230-03
[SCI] [0]		1. -03
[SCI] [2]		1.23 -03
[FIX] [5]		0.00123

÷ r - Division

The sign ÷ allows you to divide the first operand (in the stack) by the second operand (in register **X**) according to the principle of Reverse Polish Notation (RPN).

Example :

Division of 2.2 by 0.5

[2] [.] [2] [ENTER] [0] [.] [5] [÷] gives 4.4

3.1 [F] [SIN] Sin - Sinus

The **sin** function calculates the sine of an angle in radians.
The angle must be entered in radian.
If the angle is in degrees it must first be converted into radians using the function **Deg-Rad**.

Example :

sin(9°) [9] [→Rad] [SIN] gives 0.156434

[G] [ASIN] ASin - Arcsinus

The **sin-1** (Arcsine) function calculates the arcsine of an angle in radians.
The angle must be entered in radians.
If the angle is in degrees it must first be converted into radians using the function **Deg-Rad**.

3.2 [F] [COS] coS - Cosine

The **cos** function calculates the cosine of an angle in radians.
The angle must be entered in radian.
If the angle is in degrees it must first be converted into radians using the function **Deg-Rad**.

[G] [ACOS] AcoS - Arccosine

The **cos-1** (Arccosine) function calculates the arccosine of an angle in radians.
The angle must be entered in radians.
If the angle is in degrees it must first be converted into radians using the function **Deg-Rad**.

3.3 [F] [TAN] tAn - Tangent

The **tan** function calculates the tangent of an angle in radians.
The angle must be entered in radian.
If the angle is in degrees it must first be converted into radians using the function **Deg-Rad**.

[G] [ATAN] AtAn - Arctangent

The **tan-1** (Arctangent) function calculates the arctangent of an angle in radians.
The angle must be entered in radians.
If the angle is in degrees it must first be converted into radians using the function **Deg-Rad**.

3.4 [F] [STO] Sto - Store in a registry

STO (*Store*) allows to store the displayed number in the data register 0 to 9 or .0 to .9. Register number 0 to 9 or .0 to .9 is entered as an instruction parameter.

STO (*Store*) can also be used “arithmetically” by inserting an operator before the register number (0 to 9 or .0 to .9) entered as a parameter.

Example :

[2] [STO] [+] [7] adds 2 to the contents of register 7

[4] [STO] [x] [.] [1] multiplies the contents of register .1 by 4

STO (*Store*) allows you to modify the contents of the stack registers (**X**, **Y**, **Z** ou **T**).
Warning: entering a number before the STO shifts the stack !
Arithmetic operators can also be used.

[STO] [GTO] affects the **X** register in the stack.

[STO] [BST] affects the **Y** register in the stack.

[STO] [SST] affects the **Z** register in the stack.

[STO] [R/S] affects the **T** register in the stack.

In addition to direct storage functions in registers 0 to 9, .0 to .9 and stack, **STO** (Store) can also store data indirectly in registers 1 to 29, register 0 being used as an index.
These 29 registers are the “standard” or “primary” registers for the first 15 which therefore correspond to registers R0 to R.5, and indirect registers for the following 14 which can only be used in indirect addressing.

The key to signify “Index” is the key [ENTER]

Primary registers	
R 1	1
R 2	2
R 3	3
R 4	4
R 5	5
R 6	6
R 7	7
R 8	8
R 9	9
R .0	10
R .1	11
R .2	12
R .3	13
R .4	14
R .5	15

Indirect registers	
R(16)	16
R(17)	17
R(18)	18
R(19)	19
R(20)	20
R(21)	21
R(22)	22
R(23)	23
R(24)	24
R(25)	25
R(26)	26
R(27)	27
R(28)	28
R(29)	29

Example :

[7] [STO] [0] stores 7 as index value in register 0,

[2] [4] [STO] [ENTER] then stores the value 24 in the indirect register (R7)

[1] [0] [STO] [x] [ENTER] multiplies by ten the contents of the register whose index is stored in register 0.

[G] [RCL] rcL - Recall from a registry

RCL (Recall) is used to recall a number from data register 0 to 9 or .0 to .9 to the display.
Register number 0 to 9 or .0 to .9 is entered as an instruction parameter.

RCL (*Recall*) can also be used “arithmetically” by inserting an operator before the register number (0 to 9 or .0 to .9) entered as a parameter.
Example :

[2] [RCL] [+] [7] recalls the contents of register 7 and adds 2

[4] [RCL] [×] [.] [1] recalls the contents of register .1 and multiplies by 4

Note : this arithmetic operation does not affect the contents of the register but only the displayed value.

RCL (*Recall*) allows you to recall the contents of a register from the stack **X**, **Y**, **Z** or **T**.
Note : the number displayed shifts the stack !
Arithmetic operators can also be used.

[RCL] [GTO] recalls the register **X** from the stack.

[RCL] [BST] recalls the register **Y** from the stack.

[RCL] [SST] recalls the register **Z** from the stack.

[RCL] [R/S] recalls the register **T** from the stack.

In addition to the direct recall functions from registers 0 to 9, .0 to .9 and from the stack, **RCL** (Recall) can also recall data indirectly from registers 1 to 29, register 0 being used as index.
These 29 registers are the “standard” or “primary” registers for the first 15 which therefore correspond to registers R0 to R.5, and indirect registers for the following 14 which can only be used in indirect addressing.

The key to signify “Index” is the key [ENTER]

Primary registers	
R 1	1
R 2	2
R 3	3
R 4	4
R 5	5
R 6	6
R 7	7
R 8	8
R 9	9
R .0	10
R .1	11
R .2	12
R .3	13
R .4	14
R .5	15

Indirect registers	
R(16)	16
R(17)	17
R(18)	18
R(19)	19
R(20)	20
R(21)	21
R(22)	22
R(23)	23
R(24)	24
R(25)	25
R(26)	26
R(27)	27
R(28)	28
R(29)	29

Example :

[7] [STO] [0] stores 7 as index value in register 0,
[RCL] [ENTER] then recalls the value contained in the indirect register (R7)

[1] [0] [RCL] [×] [ENTER] recalls the contents of the register whose index is stored in register 0 and multiplies the display by 10.

X - Multiplication

The sign **X** allows you to multiply the first operand (in the stack) by the second operand (in register **X**) according to the principle of Reverse Polish Notation (RPN).

Example :

Multiplication of 2.2 by 0.5

[2] [.] [2] [ENTER] [0] [.] [5] [X] gives 1.1

4.1 [F] [ln] Ln - Natural logarithm

ln calculates the natural logarithm of the displayed number.
This natural logarithm uses Euler's constant as a base with the value 2.718281828459 to be calculated.
The argument of the **ln** function must be a non-zero positive number.

In case of zero or negative number, the display will show the value dt Error (*data error*), as error indication.

Example:

[5] [ln] calculates the natural logarithm of 5 = 1.609438

[G] [eˣ] EXP - Natural exponent

The natural exponent is calculated from Euler's constant (value 2.718281828459) raised to the power X.

Example:

[5] [eˣ] calculates the natural exponent of 5 = 148.4131

4.2 [F] [LOG] LoG - Decimal logarithm

LOG calculates the decimal logarithm of the displayed number.
The argument of the **LOG** function must be a non-zero positive number.

In case of zero or negative number, the display will show the value dt Error (*data error*), as error indication. .

Example:

[5] [LOG] calculates the natural logarithm of 5 = 0.698970

[G] [10^x] ALoG - Decimal exponent

10^x calculates the decimal exponent of the number **X** displayed, i.e. 10 raised to the power of **X**.

Example:

[5] [10^x] calculates the decimal exponent of 5 = 100000.

4.3 [F] [√x] SqrT - Square root

√x (*SQRT*) allows you to calculate the square root of a number. The number must not be negative.

In the case of a negative number, the display will show the value dt Error (*data error*), as error indication.

[G] [X^2] Sqr - Square of a number

The **X^2** function calculates the square of a number, or the multiple of a number by itself.

4.4 [F] [x⇆y] SWAP - Exchange of X and Y registers

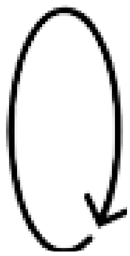
With the **x<>y** key, it is possible to swap the **X** and **Y** registers.
The **X** register is the working register and also the display contents.
The **Y** register is the register preceding the X register in the stack RPN.

[G] [R↓] rdoWn - Scrolling the stack down

With the **R down** key, it is possible to scroll through the registers of the stack by permuting them in cascade.

Example :

Stack		R↓	R↓	R↓
T	1	4	3	2
Z	2	1	4	3
Y	3	2	1	4
X	4	3	2	1



[−] - - Substraction

The sign **-** allows you to subtract the second operand (in register X) from the first operand (in the stack) according to the principle of Reverse Polish Notation (RPN).

Example :

Subtraction of 0.5 from 2.2

[2] [.] [2] [ENTER] [0] [.] [5] [−] yes 1.7

5.1 [F] [1/x] 1/X - Multiplicative inverse

The **1/x** function allows you to calculate the inverse of a number.

If the number is zero, the display will show the value dt Error (*data error*), as error indication.

[G] [yx] n - Exponentiation

The **y^x** instruction raises the first operand **Y** (in the stack) to the power expressed by the second operand **X** (displayed in the **X** register)

Example:

[3] [ENTER] [7] [yx] elevation of 3 to the power of 7 = 2187

5.2 [F] [in→] in-mm - Convert Inches to millimeters

The **IN-MM** function converts inches to millimeters.
1 " = 25.4 mm

The inch is a unit of length used in the Anglo-Saxon system of measurement units, representing 1/12 of a foot.

[G] [←mm] mm-in - Convert Millimeters to Inches

The **MM-IN** function converts millimeters to inches.
1 mm = 0.039370 "

The millimeter is a unit of length used in the metric system, equivalent to one thousandth of a meter.

5.3 [F] [mi→] mi-km - Convert Miles to kilometers

The **MI-KM** function allows you to convert miles to kilometers.
1 mi = 1.60934 km

The mile is a unit of length used in the Anglo-Saxon system of measurement units, equivalent to 5,280 feet or 1,760 yards.

[G] [←km] km-mi - Convert Kilometers to miles

The **KM-MI** function converts kilometers to miles.
1 km = 0.62137 mi

The kilometer is a unit of length used in the metric system, equivalent to 1000 meters.

5.4 [F] [lb→] Lb-KG - Convert Pounds to kilograms

The **LB-KG** function converts pounds to kilograms.
1 lb = 0.45359 kg

The pound is a unit of weight used in the Anglo-Saxon system of measurement units, equivalent to 16 Ounces (OZ).

G ←kg kG-Lb - Convert Kilograms to pounds

The **KG-LB** function converts kilograms to pounds.
1 kg = 2.20462 lb

The kilogram is a unit of weight used in the metric system, equivalent to 1000 grams.

+ + - Addition

The sign **+** allows you to add the second operand (in register X) to the first operand (in the stack) according to the principle of Reverse Polish Notation (RPN).

Example :

Addition of 2.2 and 0.5

2 . 2 ENTER 0 . 5 + gives 2.7

6.1 F CLR cLr - Clearing the display

CLR clears the **X** register so the display.

G ALL cLrALL - Clearing all Data

ALL erase all data :

- the RPN stack (X, Y, Z, T),
- all registers (0 to 9 and .0 to .9)

⇦ Input correction (backspace)

- In "execution" mode the “Back” function erases the last digit entered.
- In programming mode the “Back” function erases the current step.

6.2 F π Pi - PI

The **π** (*Pi*) key is used to enter the Archimedes constant, the value of 3,141592.

G →Rad dEG-rA - Convert Degrees to Radians

The **Deg-Rad** function converts an angle value in degrees into an angle value in Radian.

1° × π / 180 = 0,017453 rad

6.3 [F] [°F→] F-c - Convert Farenheit to Celsius

The **°F-°C** function converts degrees Farenheit to degrees Celsius.
1 °F = -17.222 °C

On the Fahrenheit scale, primarily used in the United States, the freezing point of water is set at 32 degrees, while the boiling point of water is set at 212 degrees (scale divided into 180 intervals).

[G] [←°C] c-F - Convert Celsius to Farenheit

The **°C-°F** function converts degrees Celsius to degrees Farenheit.
1 °C = 33.800 °F

On the Celsius (centigrade) scale, used in most countries as the standard unit of measurement for temperature, the freezing point of water is set at 0 degrees, and the boiling point of water is set at 100 degrees (scale divided into 100 intervals).

[.] . - Decimal point

The period (**.**) is the separator of whole digits and decimal digits of a number.
It is also used to prefix registers .0 to .9 and labels .0 to .9

6.4 [F] [LASTX] LAStX - Last X

The **LastX** function allows you to recall the last known operand in register **X**.

Example :

[2] [4] [ENTER] [3] [5] [X] gives 840

[LASTX] recall 35

[ENTER] EntEr - Enter a number

The **Enter** key validates the entry of a number and copies it into the Y register by shifting the stack (Z into T, Y into Z, X into Y) while keeping this number in the register **X** (display) until the introduction of a new number.

7. Example programs

1. Forensics

Classic calculator test to test calculation accuracy.

This "forensics" algorithm invented by Mike Sebastian to quickly provide a comparison of the accuracy of scientific calculators applies the following calculation :

arcsin(arccos(arctan(tan(cos(sin(9))))))
or
9 sin cos tan atan acos asin

Use :

GSB 0 ... start the calculation

Program :

1	LbL 0	program start label
2	Fix 9	sets the decimal places to the maximum (6 in reality)
3	9	
4	dEG-rA	converts the angle expressed in degrees into radians
5	Sin	sinus
6	coS	cosine
7	tAn	tangent
8	AtAn	arctangent
9	AcoS	arccosine
10	ASin	arcsine
11	1	|
12	8	|
13	0	> convert the angle obtained into degrees for comparison
14	×	|
15	Pi	|
16	÷	|
17	rtn	

Result : 8.999961

2. Factorial

Calculating the factorial of a number

Use :

n number for which the factorial must be calculated

GSB 1 ... launches the factorial calculation

Program :

1	LbL 1	program start label
2	Fi 11 0	set decimal places to 0
3	Sto 1	stores the number n in register 1
4	1	stores the value 1
5	Sto 2	in register 2
6	LbL 9	label for iterative loop
7	rcL 1	rank of calculation
8	StΞ 2	which multiplies the result
9	1	1
10	St- 1	decrements the rank of the calculation
11	rcL 1	reminder of the rank value
12	IIL ¬0	if different from zero
13	Gto 9	then return to the start of the loop
14	rcL 2	reminder result
15	rtn	end of program

Result : n !

3. Fibonacci

Calculates a Fibonacci number of rank n

Use :

n rank for which we must find the Fibonacci number

GSB 2 ... start the search

Program :

1	Lbl 2	program start label
2	Sto .0	stores the number n in register .0
3	1	stores the value 1
4	Sto .1	in register .1
5	0	
6	Sto .2	
7	Lbl 7	label for iterative loop
8	rcL .0	
9	1	1
10	-	decrements the rank of the calculation
11	Sto .0	reminder of the rank value
12	x=0	if equal to zero
13	Gto 8	then number found therefore end
14	rcL .1	|
15	rcL .2	|
16	+	|
17	Sto .3	> calculates the number
18	rcL .1	|
19	Sto .2	|
20	rcL .3	|
21	Sto .1	|
22	Gto 7	next iteration
23	Lbl 8	
24	rcL .3	reminder of the result
25	Fix 0	set decimal places to 0
26	r/S	end of program

4. Circle

Calculates the perimeter and area of ??a circle from the radius

Use :

r	radius of the circle
GSB 5	calculates and displays the perimeter
R/S	calculates and displays the area

Program :

1	LbL 5	program start label
2	FiII 2	set decimal places to 2
3	Sto 0	stores the radius r in register .0
4	2	|
5	×	> calculates the perimeter r x 2 x π
6	Pi	|
7	×	|
8	r/S	
9	rcL .0	recall radius r from register .0
10	SQr	|
11	Pi	> calculates the area r^2 x π
12	×	|
13	r/S	
14	Gto 5	

5. Stirling

The Stirling formula (James Stirling, Scottish mathematician, born in May 1692 in Garden near Stirling and died on December 5, 1770 in Edinburgh) makes it possible to approach the factorial of a number.

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

This improved formula will provide a better approach :

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\frac{1}{12\,n}\right)$$

When the function ***n!*** *(factorial) does not exist on a calculator, this factorial calculation of a number is usually done on programmable calculators using an iterative loop.*
This kind of calculation can be very inexpensive in terms of number of steps but excessive in time for large numbers.
On the other hand, the Stirling formula gives an approximation of the result very quickly but costs a few program steps. (see factorial program page 24)

Use :

n	number for which the factorial must be calculated
GSB 4	... launches the factorial calculation

Program :

Step	Instruction	Step	Instruction
1	LbL 4	14	y^x
2	Sto .0	15	rcL .1
3	2	16	×
4	×	17	rcL .0
5	Pi	18	1/x
6	×	19	1
7	SqrT	20	2
8	Sto .1	21	÷
9	rcL .0	22	1
10	1	23	+
11	EXP	24	×
12	÷	25	FiX 0
13	rcL .0	26	r/S

6. Binet

Binet's formula (Jacques Philippe Marie Binet, French mathematician and astronomer, born in Rennes on February 2, 1786 and died in Paris on May 12, 1856) provides the nth term of the Fibonacci sequence.

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

The calculation of the nth term of the Fibonacci sequence is usually done on programmable calculators using a loop up to n. (see Fibonacci program page 25)
This kind of calculation can be very inexpensive in terms of number of steps but excessive in time for high values ??of n.
On the other hand, Binet's formula gives the result very quickly but costs many program steps.

Use :

n	rang pour lequel il faut rechercher le nombre de Fibonacci
GSB 3	... launches the calculation

Program :

```
1  LbL 3        17    1
2  Sto 1        18    -
3    5          19    2
4  Sqrt         20    ÷
5    1          21  rcL 1
6    +          22    ^
7    2          23    5
8    ÷          24  Sqrt
9  rcL 1        25    ÷
10   ^          26  Sto .2
11   5          27  rcL .1
12 Sqrt         28  rcL .2
13   ÷          29    -
14 Sto .1       30  Fix 0
15   5          31  r/S
16 Sqrt
```

7. GCD

One of the classic little programs for programming calculators...

Many programmers started with these small programs whose usefulness was to learn the language of the newly acquired calculator..

Use :

n1	first number
ENTER	
n2	second number
GSB 6	calculates and displays the GCD

Program :

1	LbL 6	13	×
2	Sto 2	14	chS
3	rdown	15	rcL 1
4	rdown	16	+
5	Sto 1	17	Sto 2
6	LbL 7	18	rcL 3
7	rcL 1	19	Sto 1
8	rcL 2	20	rcL 2
9	Sto 3	21	x≠0
10	÷	22	Gto 7
11	int	23	rcL 1
12	rcL 2	24	r/S

8. Birthday

The birthday paradox calculates the percentage chance of finding 2 people with the same birthday (not necessarily born in the same year) in a group of n people.

$$p(n) = 1 - \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \cdots \cdot \frac{365-n+1}{365}$$

To simplify, the formula chosen assumes that all years are non-leap years.
Considering leap years would change the results of the calculations little, but would make the programs more complicated.

Use :

n	number of persons
GSB 0	... starts the percentage calculation

Program :

1	LbL 0		17	rcL 1
2	Sto 4		18	St- 2
3	3		19	rcL 2
4	6		20	St× 3
5	5		21	Gto 1
6	1/x		22	LbL 2
7	Sto 1		23	Fix 2
8	1		24	1
9	Sto 2		25	rcL 3
10	Sto 3		26	-
11	LbL 1		27	1
12	1		28	0
13	St- 4		29	0
14	rcL 4		30	×
15	x=0		31	r/S
16	Gto 2			

9. Ramanujan

Ramanujan's formula allows you to calculate the factorial of a number n.

$$n! \sim \sqrt{\pi}\left(\frac{n}{e}\right)^n \sqrt[6]{8n^3+4n^2+n+\frac{1}{30}}$$

(*Srinivasa Ramanujan, Indian mathematician, born December 22, 1887 in Erode and died April 26, 1920 in Kumbakonam*)

Use :

n number for which the factorial must be calculated

GSB 0 ... launches the factorial calculation

Program :

1	LbL 0	17	×
2	Sto .0	18	1
3	1	19	+
4	E	20	rcL .0
5	÷	21	×
6	rcL .0	22	3
7	y	23	0
8	Pi	24	1/x
9	SQrt	25	+
10	×	26	6
11	8	27	1/x
12	rcL .0	28	y
13	×	29	×
14	4	30	Fill 0
15	+	31	r/S
16	rcL .0		

10. Trigo

Calculation of the sines, cosines and tangent of an angle in degrees.
Does not use the SIN, COS, TAN, PI functions of the calculator.
The results are stored in registers .1, .2 and .3

Use :

n	angle in degrees
GSB 9	... launches the calculation

Program :

```
 1  LBL 9
 2  3      |
 3  5      |
 4  5      |
 5  ENTER  |
 6  1      > PI
 7  1      |
 8  3      |
 9  ÷      |
10  ×
11  1
12  8
13  0
14  ÷
15  STO .0
16  ENTER
17  ×
18  2
19  0
20  ÷
21  1
22  +
23  1/x
24  1
25  0
26  ×
27  7
28  -
29  RCL .0
30  3
31  ÷
32  ×
33  STO .1   SINUS
34  RCL .0
35  ENTER
36  ×
37  3
38  0
39  ÷
40  1
41  +
42  1/x
43  5
44  ×
45  3
46  -
47  RCL .0
48  RCL .0
49  ×
50  4
51  CHS
52  ÷
53  ×
54  1
55  +
56  STO .2   COSINUS
57  RCL .1
58  SWAP
59  ÷
60  STO .3   TANGENTE
61  CLR
62  R/S
```

11. Gravité

Calculation of fall time, in seconds, depending on height

Use :

h	height in meters
GSB 3	... launches the calculation

Program :

1	LbL 3
2	2
3	×
4	9
5	.
6	8
7	÷
8	SQrt
9	rtn

12. PI Day

Different approximations of PI...

Use :

GSB 1 ou 2 ou 3 ou 4 ou 5 ... calculation of PI

GSB 9 ... deviation from PI function

Program :

1	LbL 1	34	.	67	SqrT
2	7	35	9	68	SqrT
3	EntEr	36	9	69	SqrT
4	6	37	1	70	SqrT
5	EntEr	38	EntEr	71	SqrT
6	5	39	7	72	1
7	SqrT	40	÷	73	4
8	+	41	r/S	74	SqrT
9	SqrT	42	LbL 5	75	SqrT
10	+	43	8	76	SqrT
11	SqrT	44	SqrT	77	SqrT
12	r/S	45	SqrT	78	SqrT
13	LbL 2	46	SqrT	79	SqrT
14	1	47	SqrT	80	SqrT
15	.	48	SqrT	81	SqrT
16	8	49	SqrT	82	SqrT
17	SqrT	50	SqrT	83	SqrT
18	LAStx	51	SqrT	84	SqrT
19	+	52	SqrT	85	SqrT
20	r/S	53	SqrT	86	6
21	LbL 3	54	SqrT	87	8
22	3	55	SqrT	88	SqrT
23	5	56	SqrT	89	SqrT
24	5	57	SqrT	90	SqrT
25	EntEr	58	SqrT	91	SqrT
26	1	59	SqrT	92	SqrT
27	1	60	SqrT	93	+
28	3	61	SqrT	94	+
29	÷	62	SqrT	95	r/S
30	r/S	63	SqrT	96	LbL 9
31	LbL 4	64	SqrT	97	P
32	2	65	SqrT	98	-
33	1	66	SqrT		

13. Premier

Finding the prime number closest to the number n entered.

Use :

n	maximum number for search
GSB 0	... launches the search

Program :

1	LbL 0	24	Sto+ .1
2	FIX 0	25	rcL .1
3	1	26	rcL .3
4	+	27	SWAP
5	SQrt	28	x≤y
6	Sto .4	29	Gto 3
7	1	30	rcL .2
8	Sto .1	31	Sto .0
9	3	32	1
10	Sto .2	33	Sto .1
11	LbL 1	34	Gto 1
12	2	35	LbL 3
13	Sto+ .2	36	rcL .2
14	rcL .2	37	rcL .1
15	SQrt	38	÷
16	Sto .3	39	FrAc
17	rcL .4	40	x≠0
18	x>y	41	Gto 2
19	Gto 2	42	1
20	rcL .0	43	Sto .1
21	r/S	44	Gto 1
22	LbL 2	45	r/S
23	2		

14. Hilo

HILO game: you have to guess a number...
If the number proposed is less than the guess number, display of -1
If the number proposed is greater than the guess number, display of 1
If found display of 88888 then display of number of moves played.

Use :

xx.xxxx “seed” number...

[GSB] [0] ... launches the game

n *(between 0 and 1000)* [R/S] ... to repeat until the end of the game

Program :

1	LbL 8		24	LbL 3
2	SqrT		25	r/S
3	FrAc		26	Sto 6
4	3		27	1
5	ALoG		28	Sto+ 5
6	÷		29	rcL 6
7	int		30	rcL 1
8	Sto 1		31	II=Y
9	0		32	Gto 1
10	Sto 5		33	II⌝Y
11	Fix 0		34	Gto 2
12	8		35	rcL 4
13	8		36	Gto 3
14	8		37	LbL 1
15	8		38	rcL 2
16	8		39	PSE
17	Sto 2		40	PSE
18	1		41	PSE
19	chS		42	rcL 5
20	Sto 3		43	r/S
21	1		44	LbL 2
22	Sto 4		45	rcL 3
23	clr		46	Gto 3

15. Fraction

Returns a number to 2 decimal places as a fraction.

(Example : 12.48 ... 312/25)

Use :

nn.nn	number to transform into fraction (Example : 12.48)
GSB 0	... launches the calculation
	... then displays the numerator (Example : 312)
R/S	... displays the denominator (Example : 25)

Program :

1	LbL 0	19	×
2	FIX 2	20	chS
3	1	21	rcL 1
4	0	22	+
5	0	23	Sto 2
6	Sto 1	24	rcL 3
7	Sto .1	25	Sto 1
8	×	26	rcL 2
9	int	27	x≠0
10	Sto .2	28	Gto 7
11	Sto 2	29	rcL .2
12	LbL 7	30	rcL 1
13	rcL 1	31	÷
14	rcL 2	32	R/S
15	Sto 3	33	rcL .1
16	÷	34	rcL 1
17	int	35	÷
18	rcL 2	36	R/S

16. Convert

Decimal to binary or binary to decimal conversion.

Use :

	GSB 8	for decimal to binary	displays 10.2
OR	GSB . 8	for binary to decimal	displays 2.10
number	GSB 0	... launches the conversion	

Program :

1	LbL 0
2	Fill 0
3	Sto .0
4	Sto .4
5	rcL .1
6	rcL .2
7	-
8	Sto .3
9	LbL 1
10	rcL .4
11	rcL .2
12	÷
13	int
14	Sto .4
15	x=0
16	Gto 2
17	rcL .3
18	×
19	St+ .0
20	rcL .1
21	St× .3
22	Gto 1
23	LbL 2
24	rcL .0
25	r/S
26	LbL 8
27	1
28	0
29	Sto .1
30	2
31	Sto .2
32	1
33	0
34	÷
35	+
36	Fill 1
37	r/S
38	LbL .8
39	2
40	Sto .1
41	1
42	0
43	Sto .2
44	2
45	.
46	1
47	Fill 2
48	r/S

WELCOME TO THE RPN-1250 CALCULATOR

Benoit Maag © 2015

KEY FEATURES

Based on Microchip PIC 18F2680
(64k FLASH) at 8 MHz or PIC 18F26K80
(64k FLASH) at 16 MHz
MAX 7219 Display driver
8-digit 7-segment LED display
RTC (RPN-1250+)
Original TI-1250 enclosure
9V standard battery

Programmable – 98 steps
Save 3 programs to continuous memory
6 levels of subroutine
All 12 tests
Indirect addressing

Scientific Functions

Basic Conversions

Alphanumeric Display (brightness adjustable)

In RUN mode:
SV saves all memories
and stack into
continuous memory
LD restores all memories
and stack from
continuous memory

20 memories: 0~9 and .0 ~ .9 + 10
additional through indirect addressing
Store and Recall Arithmetic

RPN

RUN MODE

GTO 0~9 or .0~.9 goes to corresponding label
GSB 0~9 or .0~.9 execute program from corresponding label
RTN in run mode goes to program step 0

OFF/ON Switch

SHIFT KEY:
Press once for F
Press twice for G
Press a third time to go back to unshifted

Press once for program entry mode
Press again to go back to Run mode

TRIG functions : angle in RAD only

CLR: clears X Register
ALL: clears stack and all memories

In RUN mode:
SV saves all memories and stack into continuous memory
LD restores all memories and stack from continuous memory

IP: Integral Part
FP: Fractional Part

20 memories: 0~9 and .0 ~ .9
Store and Recall Arithmetic

PROGRAM MODE

20 labels 0~9 and .0~.9

OFF/ON Switch

TST: all 12 tests

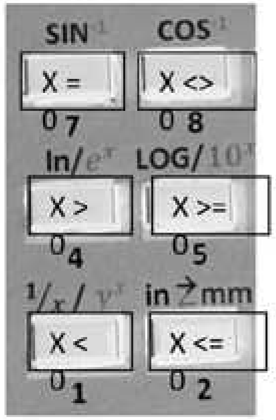
In Program mode:
SV followed by 1, 2 or 3 saves program into continuous memory (you can save up to 3 programs)

LD followed by 1, 2 or 3 restores program 1, 2 or 3 from continuous memory

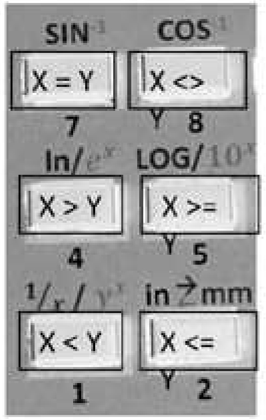
Run Program
VIEW 0~9 or .0~.9 to view a register without stopping program

Stops a running Program

After TST, press below key for test vs 0:



After TST, press . (dot) and then below key for test vs Y:



STACK REGISTERS
INDIRECT ADDRESSING

Store and Recall functions (including store and recall arithmetic) and VIEW can operate on stack registers, which are linked to the top row keys

As in the HP-29C the indirect addressing register is R0

Indirect addressing (i) works with Store, Recall (including store and recall arithmetic) and VIEW functions only (No fix (i), or GSB (i)) and is called with the [ENTER] key

REAL TIME CLOCK (RTC)
RPN-1250+ Only

Time, Date and Day Of Week are addressed as memory registers using the [PREFIX] [PRGM] and [CHS] Keys

Time format is hh.mmss (24 hour)

Date format is mm.ddyy

Day of Week is 1~7 for Mon-Sun

RTC functions are programmable

(*) DAY OF WEEK

ESC MODE


Displays Firmware Version
Shows all alphanumeric characters
Display test (lits up all display segments)
- to lower display brightness
+ to increase display brightness
G prefix + ENTER: to enter 'Esc mode'
Press ENTER to leave 'Esc Mode'