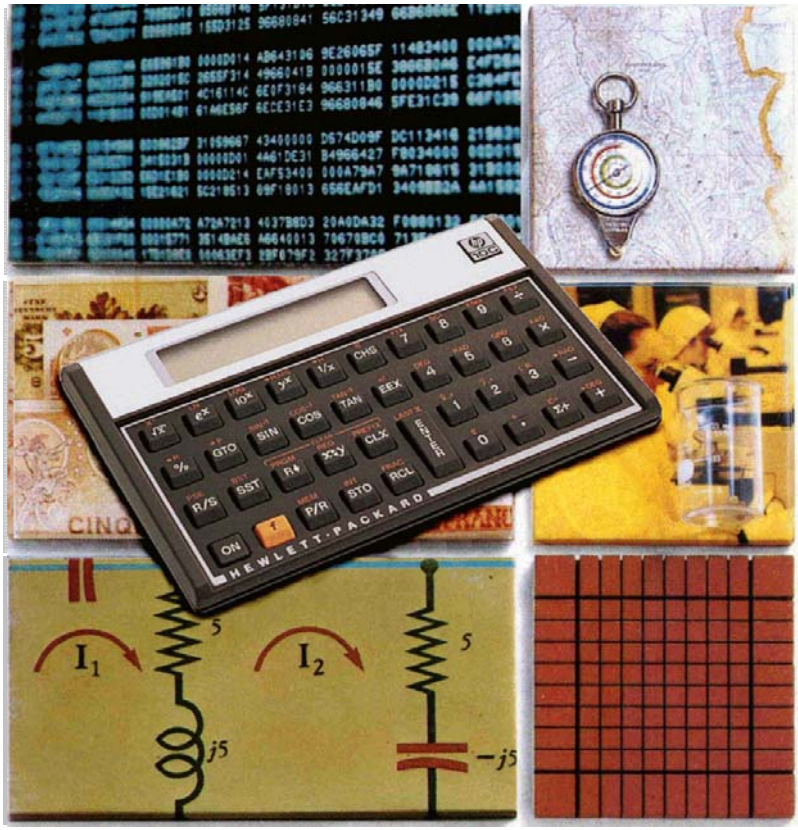
Museum of
Hewlett-Packard
Calculators

VOYAGERS

IN BRIEF



Luiz C. Vieira (Organizer)

DISCLAIMER

THIS MANUAL AND ANY EXAMPLES CONTAINED HEREIN ARE PROVIDED “AS IS” AND ARE SUBJECT TO CHANGE WITHOUT NOTICE. THE MUSEUM OF HEWLETT-PACKARD CALCULATORS OR ITS CURATOR MAKE NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. THE MUSEUM OF HEWLETT-PACKARD CALCULATORS SHALL NOT BE LIABLE FOR ANY ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MANUAL OR THE EXAMPLES CONTAINED HEREIN.

MoHPC

Voyagers in Brief

September 2009

THIS PRINTED COPY BELONGS TO:

1st. Edition - Rev. 2 - September 2009

Hello, Voyager Owner and/or User!

I will not congratulate you for buying or using one of the most interesting series of calculators ever offered by Hewlett-Packard. HP surely has already done that!

I found some spare time to put together the information herein and offer it as a book, a personal homage to both Hewlett-Packard design team and you, final user. Please, if you find the time to share additional wisdom (and to correct my Latin-flavored English...), feel free to drop me an e-mail at:

linux_lcv@yahoo.com.br

The book has been written with you in mind, so you are the first one to tell if it fits your needs or not. Your suggestions, corrections and any kind of contribution will be welcome.

As a matter of fact, some of what is written was *copied-&-pasted* from original HP manuals. I do not deny this fact, and because the book is not for commercial purposes (if you bought this book, sue someone!) and it addresses HP users, I took it as harmless. Any lawyers reading, please?

Maybe you already have read some of my posts at the MoHPC, and if you do not know about this yet, you are probably wondering if I am a native English speaker. Well, I'm not. I'm a native Portuguese speaker, and if you complain for studying Spanish, you don't know how lucky you are! Portuguese is a pain in the...

Sorry! So please, forgive me if whatever you read is awkward or hard to understand. I did not mean to be mean! If you are convinced my English writing is awkward that's because you have not heard me speaking yet...d8^D

'Nough said, I really hope you enjoy reading this book as much as I enjoyed writing it. Or even more! I really had a good time writing it, I actually would like knowing English a bit more so I could be sure I wrote what I meant.

Thanks and best regards.

Luiz Cláudio Vieira

Araguari - MG, Brazil. September 2009.

INDEX

Read this first, please.

This book is intended to show the many resources available with the HP Voyagers HP10C, HP11C, HP12C, HP15C and HP16C. These calculators are state-of-the-art products made by Hewlett-Packard Company and are meant to be precise, long lasting, high-quality tools for personal computation. If you do not have access to or do not know any of these calculators, this is the very moment to get acquainted to their powerful resources.

We hope you enjoy reading this book, at the same time it is our wish that you find all of the information you need here. If you have comments, suggestions or simply want to share your experience with us, feel free to add your post at:

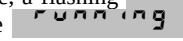
www.hpmuseum.com

Testing your Calculator

If you already know how to perform the tests described below, just skip to the next subject: Setting Up the Calculator. If you want to test your calculator and do not know how to do it, go ahead reading. The execution of these tests does not change memory contents.

Please, locate the [ON] key (lower-left corner) and check if your calculator can be turned on and off. Now, locate the [×] key and:

1) make sure that the calculator is turned off;
2) press *and hold down* the [×] key;
3) while [×] is held down, turn the calculator on;
4) release the [×] key (this test is also known as [ON]/[×] test)

If everything is fine, a flashing  message is shown for a few seconds. If the  message is not shown, please start the procedure again from step # 1. The average time spent for the test is 10 seconds for the HP10C, 25 seconds for the HP15C and 13 seconds for the others.

	HP10C	HP11C	HP12C	HP15C	HP16C
[ON]/[x] time spent:	~10s	~13s	~13s	~25s	~13s

While the test is "PUMA ING", do not press any key unless the expected time to resume is exceeded. In some rare situations, the test falls into an endless loop and, in this case, it may be necessary to remove the batteries. If you believe you have to remove the batteries, keep in mind that the memory contents may be lost.

The [ON]/[x] test normally resumes in one of two ways:

- the display shows 

which means the calculator internals are working fine;
- the display shows 

which means the calculator needs service.

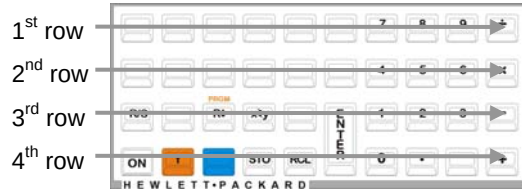
The [ON]/[x] test may also be performed with the sequence [ON]/[+] with a small difference: [ON]/[+] starts an endless loop that must be interrupted by pressing any key but the [ON] key. If you press the [ON] key in this case, the calculator is turned off immediately after the test finishes and you will not be able to see how the test ends.

To test the keyboard, locate the [÷] key and:

1) turn the calculator off;
2) press *and hold down* the [÷] key;
3) while [÷] is held down, turn the calculator on;
4) release the [÷] key.

At this point, the user must press all keys in the keyboard, following a predefined sequence. The drawing below represents this sequence: the four arrows over the keys show the keystroke sequence that should be followed.

Note that the [ENTER] *key must be pressed twice, one time for the third row and another time for the fourth row.* To help the user, a different pattern is shown in the display each time a different key is pressed.



If the test succeeds, the two digits that identify the calculator model are shown in the middle of the LCD: 10, 11, etc.

If any key repeats or fails when pressed, Error 9 is shown in the display. This message is also shown if you inadvertently press the same key twice or skip one of them. In the case Error 9 is shown in the display, you should press any key to clear the message out. A number identifying the failing key is placed in the display (maintenance purposes only).

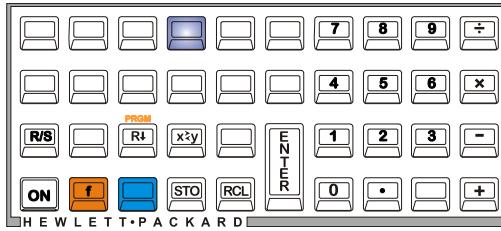
If the calculator does not properly respond to keystrokes, a *machine reset* may be performed. To perform the machine reset, please locate the shaded key shown in the drawing below (the key is shaded for location purposes). Although performing the same task in the machine reset, the shaded key shows a different label for each calculator:

- [y^x] in the HP10C, HP11C and HP15C
- [PMT] in the HP12C
- [D] in the HP16C

Calculator	Key	Calculator	Key	Calculator	Key
10C, 11C, 15C	[y^x]	12C	[PMT]	16C	[D]

The machine-reset procedure is the same to all of them:

- press and release both [ON] and the equivalent shaded key for your calculator at the same time (see the keyboard diagram below).



The expected reaction is a change in the display contents, hence you should clear the display contents with [CLx] after a machine reset. Normally, the machine reset does not cause the calculator to lose memory contents.

Reset the calculator to the default state

All Voyagers offer an easy and fast way to clear all memory contents and reset all settings to their default state. For each model, this default state has some particular references, and they are summarized in the table below.

Once you decide to ***reset calculator memory*** to the default state, keep in mind that this action ***cannot be undone***, meaning that ***there is no way to restore memory contents once the procedure is completed***. To perform the so called "master clear" or "master reset", locate the [−] key and:

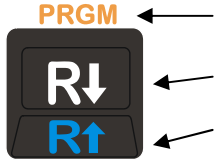
1) turn the calculator off;
2) press *and hold down* the [−] key;
3) while [−] is held down, turn the calculator on;
4) release the [−] key

The display should read Pr Error, which means all memory contents are lost and all settings are in their default condition, according to the following table.

Default after reset	Display Mode	Available Memory		Memory contents	Flags status
		Program	Registers		
HP10C	FIX 4	9 steps	10	All cleared	
HP11C	FIX 4	63 steps	21		cleared
HP12C	FIX 2	8 steps	25[1]		
HP15C	FIX 4	46 regs [2]	21		cleared
HP16C[3]	HEX	0	201		cleared

Setting Up the Calculator

The keyboard of each Voyager has two prefix keys: f and g (except for the HP10C, that has only the f prefix key). So, many keys in the keyboard allow either a second or even a third action to be performed.



To perform this action, press f first then press the key

To perform this action, simply press the key

To perform this action, press g first then press the key (g prefix key is not available in the HP10C)

In order to "see" the same things in any of the Voyagers while following the examples, it is necessary to set the display format accordingly. To clear the display contents to zero we use CLx, located at the left of the ENTER key. Press the CLx key if CLx is printed in the top of the key with white

[1] The available memory for the HP12C takes into account the financial registers

[2] The available memory for program in the HP15C is defined by the movable partition, explained in the HP15C's specific chapter

[3] In the HP16C, the default condition also comprises word size of 16 bits and 2's complement mode

letters (HP10C, HP12C), or press the g key prior to press CLx if CLx is printed in blue, in the slanted face of the key (HP11C, HP15C, HP16C). The number you see now in the display should be zero. If the display does not read 0.0000, please set it according to the instructions below.

If you are using an HP10C, 11C or 15C, locate the 7 key and check for FIX printed over it, in the keyboard faceplate. Press the following keys in this sequence:

f FIX 4

If you have an HP12C, you may notice that the default display is 0.00 (financial intended) instead of 0.0000. If you want to change it to four decimal places, simply press:

f 4

If your calculator is an HP16C, then the default display after g CLx is 0 h instead of 0.0000. To change the current number representation to a representation with four decimal places, first locate the RCL key and verify that FLOAT is printed over it, in the keyboard faceplate. Then press:

ENTER[4] f FLOAT 4

You can choose either a dot as radix mark and comma for thousands separator or the opposite. If you want to change the current radix mark setting in your calculator, locate the . key and:

1) turn the calculator off;
2) press *and hold down* the . key;
3) while . is held down, turn the calculator on;
4) release the . key (this sequence is also referred to as ON/.)

You can repeat this procedure as many times as you want to, and each time you do it, the separators will be exchanged with each other.

[4] ENTER is needed because the HP16C will try to compute a floating-point equivalent.

Annunciators

If your calculator is working fine, the following display is shown after the internal circuitry test is performed (either ON/× or ON/+ sequence):

The set of characters that appear in the bottom of the display are named annunciators, and they signalize (announce) a particular operating condition at the moment you are using the calculator. Their meaning is summarized in the table below.

Annunciator	Meaning
*	When this annunciator flashes, batteries should be changed for new, fresh ones. In normal operation, it is off.
f	f key pressed, prefix action active
g	g key pressed, prefix action active (except HP10C)
USER	User keyboard active (HP11C/HP15C only)
BEGIN	Payment in advance (HP12C only)
G	Out-of-Range condition (HP16C only)
GRAD	grads angle mode set (HP10C/HP11C/HP15C)
RAD	radians angle mode set (HP10C/HP11C/HP15C)
D.MY	dd.mmyyyy data mode selected (HP12C only)
C	HP12C: perform compound interest in odd period HP15C: flag 8 is set, complex mode active HP16C: flag 5 is set, carry/borrow condition occurred
PRGM	program mode active

If there is no specific mention, the annunciator has the same meaning in all models.

The HP15C and the Complex Mode

If you decided not to perform the master clear ([ON]/[−]), chances are that some of the annunciators mentioned above are active in the display. There is no need to change any of them for now because the relevant conditions will be set in the examples. The only remark goes to the HP15C. If you are using an HP15C and it is set to work with complex numbers, the small c annunciator is visible, and some functions that would produce error messages in the other calculators may produce complex results in the HP15C in complex mode, instead. To change this, locate the [5] key and verify that [CF] is printed in the front, slanted face of it. To turn the complex mode off, please use the following keystroke sequence:

[g] [CF] 8

Contents of this Book, Credits and Copyright

Most of the contents of this book were taken from the original Voyagers manuals, with the sole purpose of keeping original texts. This book is not credited to a particular author, it is copyrighted to the MoHPC, Museum of Hewlett-Packard Calculators, and its curator, Dave Hicks. The compilation of data, its organization in chapters and additional, original texts provided by Luiz Cláudio Vieira.

Fundamentals

Introduction

Hewlett-Packard introduced the first Voyager Series in 1981 (HP11C and HP12C) and the remaining ones in 1982 (HP10C, HP15C and HP16C).

Model	10C	11C	12C	15C	16C
Intro-Disc	1982-84	1981-89	1981-(*)	1982-89	1982-89

(*) still in production

The reference made to each Voyager on their particular Owner's Handbook suggests that they are "specialized" on specific subjects:

HP10C / 11C: *Programmable Scientific Calculator*;
HP12C: *Programmable Financial Calculator*;
HP15C: *Advanced Programmable Scientific Calculator*; and
HP16C: *Computer Scientist*

How Voyagers' features are treated in this book

The many features common to all Voyagers are shown and discussed in this book at the same time. The features that are common *but* equal are also shown, and any necessary remarks were added so that all differences for each model are discussed, too. The specific features for the three most specialized models are shown and discussed in their specific chapters.

Function Tables

The two tables below summarize all functions in each model, represented by their corresponding key label. The first table is focused on groups of functions available in all five models, except for Statistics, not available in the HP16C, and trigonometry-related functions, not available in both HP12C and HP16C. The second table shows particular resources available with the more specialized models: HP12C, HP15C and HP16C.

Model	10C	11C	12C	15C	16C
Number input	0…9 . CHS EEX CLx	0…9 . CHS EEX CLx ←	0…9 . CHS EEX CLx	0…9 . CHS EEX CLx ←	0…F . CHS EEX CLx BSP
Stack Manipulation	ENTER R↓ LSTx x≷y	ENTER R↓ LSTx x≷y R↑	ENTER R↓ LSTx x≷y	ENTER R↓ LSTx x≷y R↑	ENTER R↓ LSTx x≷y R↑
Display Control	FIX SCI ENG 0 … 9	FIX SCI ENG 0 … 9	f 0 … 9 .	FIX SCI ENG 0 … 9	FLOAT 0 … 9 .
Data clearing	CLEAR PRGM CLEAR REG	CLEAR Σ CLEAR PRGM CLEAR REG	CLEAR Σ CLEAR PRGM CLEAR FIN CLEAR REG	CLEAR Σ CLEAR PRGM CLEAR REG	CLEAR PRGM CLEAR REG
Register access and arithmetic	STO RCL STO + − × ÷	STO RCL STO + − × ÷ x≷I x≷(i)	STO RCL STO + − × ÷	STO RCL STO + − × ÷ RCL × ÷ x≷	STO RCL x≷I x≷(i)
Arithmetic	+ − × ÷	+ − × ÷	+ − × ÷	+ − × ÷	+ − × ÷
Basic Math and number functions	√x x² 1/x y^x LN e^x LOG 10^x % INT FRAC	√x x² 1/x y^x LN e^x LOG 10^x % Δ% RND INT FRAC	√x 1/x y^x LN e^x % Δ% %T RND INTG FRAC	√x x² 1/x y^x LN e^x LOG 10^x % Δ% RND INT FRAC	√x 1/x
Program	SST BST R/S PSE P/R GTO x=0 x≤y	SST BST R/S PSE GTO LBL GSB RTN ISG DSE x=0 x=y x≠0 x≠y x>0 x>y x<0 x≤y USER P/R	SST BST R/S PSE P/R GTO x=0 x≤y	SST BST R/S PSE GTO LBL GSB RTN ISG DSE x=0 x≤y TEST 0 … TEST 9 USER P/R	SST BST R/S PSE GTO LBL GSB RTN ISZ DSZ x=0 x=y x≠0 x≠y x>0 x>y x<0 x≤y P/R
Misc.	CLR PREFIX f MEM	CLR PREFIX f g MEM	CLR PREFIX f g MEM	CLR PREFIX f g MEM	CLR PREFIX f g MEM
Statistics	n! Σ+ Σ− s x̄ L.R. ŷ,r x̂,r	Σ+ Σ− s x̄ L.R. ŷ,r x! RAN# Py,x Cy,x	Σ+ Σ− s x̄ n! x̄w ŷ,r x̂,r	Σ+ Σ− s x̄ L.R. ŷ,r x! RAN# Py,x Cy,x	

Model	10C	11C	12C	15C	16C
Trigonometry, Hyperbolic and related functions	SIN⁻¹ COS⁻¹ TAN⁻¹ SIN COS TAN →H.MS →H →P →DEG →R →RAD DEG RAD GRD π	SIN⁻¹ COS⁻¹ TAN⁻¹ SIN COS TAN →H.MS →H →P →DEG →R →RAD DEG RAD GRD π HYP HYP⁻¹		SIN⁻¹ COS⁻¹ TAN⁻¹ SIN COS TAN →H.MS →H →P →DEG →R →RAD DEG RAD GRD π HYP HYP⁻¹	

The table below summarizes the specific functions available with the HP12C, HP15C and HP16C.

HP12C	HP15C	HP16C
n i PV PMT FV AMORT INT NPV IRR 12× 12÷ CFo CFj Nj BOND YTM SL SOYD DB ΔDYS DATE BEG END D.MY M.DY	MATRIX RESULT DIM (i) I Re≷Im SOLVE ∫	HEX DEC OCT BIN XOR AND NOT OR SL SR RL RR RLn RRn LJ ASR RLC RRC RLCn RRCn MASKL MASKR RMD DBLR DBL÷ DBL× #B SB CB B? WINDOW 1'S 2'S UNSGN WSIZE FLOAT STATUS

Some important points:

- the HP16C's exclusive BSP key (Back SPace) acts exactly the same as ← available in the HP11C/15C;
- both HP12C and HP16C have no SCI neither ENG keys, but either f • (HP12C) or FLOAT • (HP16C) causes the numbers in the display to be shown the same way as SCI 7 in the other three models
- register arithmetic in the HP12C is defined only in the range R_0 to R_4
- there are 12 conditional tests in the HP15C, being x=0 and x≤y directly available in the keyboard (common to all Voyagers). TEST 0 to TEST 9 provide the other ten conditional tests available in the HP15C, described in the chapter about programming.

Keyboard Resources

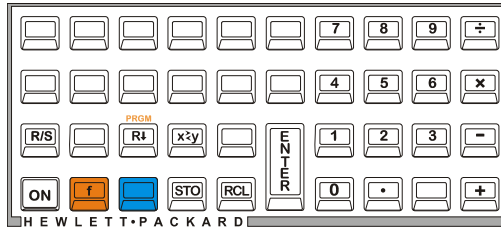
As mentioned previously, except for the HP10C that has only the f prefix key, each Voyager offers two prefix keys: f and g. This means that many keys in the keyboard allow not only the primary action to be performed for each key, but also a second or even a third one. The term *function* is used in this text as a general reference to the action performed by the calculator when a key is pressed. The term *instruction* may also be found.

To execute this function, press f first, then press the key

To execute this function, simply press the key: R↓

To execute this function, press g first, then press the key; commonly referred to as g R↑ (except the HP10C)

The previous function tables allow an easy identification of the common functions to all Voyagers. The few ones that share the same location in the keyboard are shown in the picture below[5].



Amongst the many functions common to all Voyagers, some actually share the same location in a few models. As an example, 1/x is available in all of the Voyagers and shares the same location in the keyboard of the HP10C, HP11C and HP15C. In the HP12C it has a different location, and in the HP16C it is a prefixed function: g 1/x. In order to keep a shorter main text, prefix keys are suppressed unless their use is necessary or clarifying.

[5] Although the HP10C has no g prefix key, the color was kept for practical purposes only

Getting Started

Power On and Off

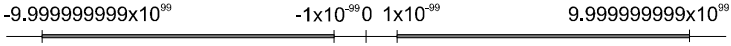
Press [ON] to turn the calculator on and off. The calculator automatically turns itself off to conserve battery power if left inactive for about ten minutes.

Keying In Numbers

The real numbers your calculator handles as data can be represented as:

$$\pm(mantissa) \times 10^{(exponent)}$$

where *mantissa* is a ten-digit real number[6] in the range {$1 \le mantissa < 10$}, and *exponent* (also: '*exponent of ten*') is an integer in the range {$-99 \le exponent \le 99$}. The gray areas in the diagram below represent the working range in all Voyagers:



The keys used to key in numbers in your calculator (*digit entry keys*.) are:

[0]...[9][7] [•] [CHS] [EEX][8] [←][9]

To key in any number, use the digit keys ([0] to [9]). After keying in the integer part of the mantissa, press [•] to enter the fraction part of it.

---

[6] Some discussions about the HP15C lead to the conclusion that it uses an internal representation of a 13-digit mantissa, instead

[7] When using an HP16C in hexadecimal, integer mode, [A] to [F] are used to key in numbers.

[8] The keystroke sequence to enter an exponent of ten in the HP16C is [f] [EEX]

[9] HP11C and HP15C only

To enter an exponent of ten, press [EEX] right after keying in the mantissa, then enter the exponent of ten. Note that [EEX] is ignored if pressed after keying in a mantissa that has more than seven digits in the integer part.

Use [CHS] to change the signal of either the mantissa or the exponent of ten while it's been keyed in. If [CHS] is pressed after digit entry is finished, it changes only the sign of the mantissa.

While keying in numbers, the HP11C, HP15C and HP16C allow backspacing. The backspacing key for the HP11C or HP15C is [←]. The HP16C has the [BSP] key, instead. Neither the HP10C nor the HP12C have backspacing capability.

Some important facts:

- Any other keys but the digit entry keys terminate digit entry.
- None of the Voyagers allow editing numbers already entered.
- Use [CLx] to replace the display contents to 0.0000.

Display Control

All voyagers allow numbers already keyed in to be showed either with ***n*** fixed decimal places or in scientific notation. The HP10C, HP11C and HP15C also allow numbers to be shown in engineering notation. The following tables show the keystrokes to do so.

Fixed Decimal Places

The fraction part of a number can be shown in the display with a fixed number of places, from zero to a maximum of nine. To define the number of places to be shown in the fraction part, use one of the following sequences:

HP10C, HP11C, HP15C	HP12C	HP16C
[f] [FIX] (0 to 9)	[f] (0 to 9)	[f] [FLOAT] (0 to 9)

When set to show numbers with fixed decimal places, the calculator will automatically switch the representation to scientific notation if the magnitude of the number does not allow it to be correctly shown.

Scientific Notation

Numbers in scientific notation are shown as a mantissa and an exponent of ten, in the range previously discussed in the beginning of this topic. The HP10C, HP11C[10] and HP15C allow choosing how many significant digits to show in the mantissa. The HP12C and the HP16C allow only setting scientific notation, and in their case, the mantissa is always shown with seven digits.

HP10C, HP11C, HP15C	HP12C	HP16C
f SCI 0 to 9	f .	f FLOAT .

Engineering Notation (HP10C, HP11C, HP15C)

Engineering notation can be seen as a modified scientific notation where the exponent of ten is the closest multiple of three that ensures the mantissa is shown in the range $0<|M|<1000$ (M is the mantissa). The argument for engineering means how many .

f ENG 0 to 9

RPN

RPN (short for **R**everse **P**olish **N**otation) is the core of most HP calculators, all Voyagers included. After setting up the calculator the way shown above, exploring the available resources through RPN becomes an easy task.

[10] Although the sequences f SCI 8, f SCI 9, f ENG 8 and f ENG 9 are valid for the HP11C, they are replaced for f SCI 7 or f ENG 7 if keyed in while loading programs.

One- and Two-number Functions and the [ENTER] key.

Some functions available in your calculator operate only in the number shown in the display. These are called one-number functions. Other functions need two numbers to generate results, and they are consequently called two-number functions. If you need to sequentially enter two numbers in order to perform a two-number calculation, use [ENTER] to indicate that the following digit keys are part of a new number.

Clearing Functions

Whenever a series of calculations are about to start, it is a good idea to understand how do clearing functions work, so only unnecessary data will be cleared, whenever needed. All voyagers have their clearing functions, grouped under a large CLEAR brace (shortly noted as CLR) at the left side of the [ENTER] key. The clearing functions that are common to all models are:

[f] CLR [PREFIX] - clears partial keystroke sequences, including f and g annunciators, if applicable. [f] CLR [PREFIX] is not programmable.

[f] CLR [REG] - sets the contents of all available registers to 0.0000. In the HP12C, the registers for grouped cash flow [Nj] are all set to 0 1. [f] CLR [REG] ***cannot be part of a program*** in the HP12C.

[f] CLR [PRGM] - if performed while in program mode, clears all program memory (erases all existing programs); in normal, run mode, simply sets program memory to step # 00 (HP10C, HP12C) or 000 (HP11C, HP15C, HP16C). [f] CLR [PRGM] is not programmable.

The other clearing functions are:

[f] CLR [Σ] - HP11C, HP12C and HP15C. Clears accumulated data related to statistics, stored in memory registers designated to store them. Please, check the label in the back of your calculator for their locations. Also clears the contents of stack registers X, Y, Z and T to 0.0000.

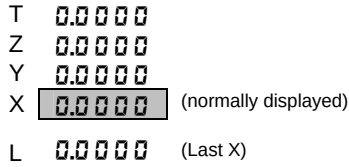
[f] CLR [FIN] - ***HP12C only*** - Sets financial registers [n], [i], [PV], [PMT] and [FV] to zero; all other registers remain intact.

Memory in General

The best way to start using any Voyagers' features is by understanding the Automatic Memory Stack. The secret of RPN efficient usage is the best understanding of the stack register manipulation. Because of its man-machine interface specially developed to financial computations, the HP12C's manual leaves stack register manipulation to Appendix A. All other Voyagers' manuals deal with the Automatic Memory Stack in their first chapters.

Stack Registers

All Hewlett-Packard calculators with RPN interface use the basic 4-level automatic memory stack. The stack itself is a dedicated memory structure over which contents almost all keyboard operations are performed when the calculator is in Run mode (that is, when the **PRGM** annunciator is not displayed). The number that appears in the display is the number in the X-register.

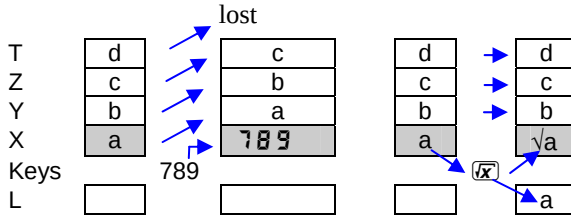


The Automatic Memory Stack Registers and their names

Complex calculations are easily performed because the stack registers automatically retain and return intermediate results. These features are based on the automatic memory stack and mostly the [ENTER] key.

Numbers in the stack are available on a last-in, first-out basis. Any number keyed in or resulting from the execution of a numeric function is placed in the display (X-register). Executing a function or keying in a number will cause stack registers contents either to lift, remain in the same register or drop, depending upon the type of operation being performed. If, as a result

of any applied function, the X-register contents are changed, its previous contents are saved in LSTx (Last X) register. Last X contents are retrievable through g LSTx. Consider that the stack is loaded with four arbitrary numbers a, b, c and d, as shown below. Pressing the indicated keys would result in the stack rearrangement shown on the right of each illustration.



Notice that the square-root is located in different key locations, according to your calculator model. Also, notice the contents of the T-register being lost when new numbers are 'pulled' into the automatic memory stack. This is called 'automatic stack lift'.

Functions and Keys for Stack Manipulation

ENTER - ENTER actually performs three operations:

- lifts the stack contents up.
- copies the number in the display (X-register) into the Y-register.
- disables automatic stack lift; if a number is keyed in right after ENTER, X-register contents are overwritten and stack contents do not lift.

ENTER is mostly used for separating numbers that are keyed in one after the other, mostly when performing two-number functions.

R↓ - Roll-down - rotates stack-contents down, placing X-register current contents into T-register while other contents 'drop' down one register.

g R↑[11] - Roll up - opposed to R↓, rotates stack contents up.

x≷y - X exchanges Y - exchanges contents of registers X and Y in automatic stack

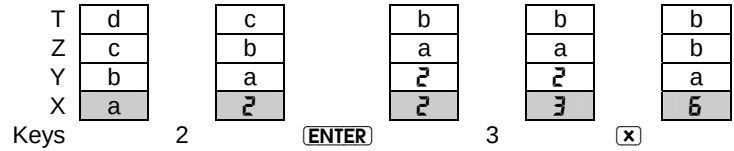
f x≷[12] - X exchanges register contents - exchanges contents of stack register X and one amongst some possible registers (discussed later)

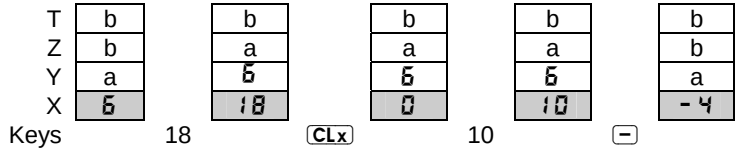
CLx[13] - Clear X-register contents - Sets X-register contents to 0.0000; also disables stack lift.

←[14]/BSP[15] - back space key - while keying in numbers, allows last digit to be cleared; works as CLx if the number has already been entered. Also used for editing programs.

g LSTx[16] - Last X - copies Last X register contents into the X-register

The diagrams below show automatic stack contents when performing several different calculations.





[11] Not available with the HP10C, nor with the HP12C

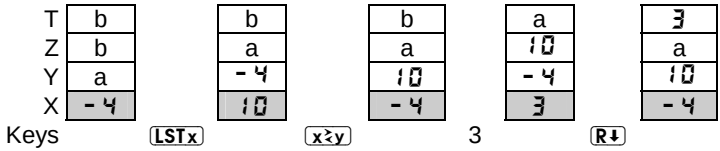
[12] Available only in the HP15C

[13] In the HP11C, HP15C and HP16C, CLx is a prefixed key: g CLx

[14] Available only in the HP11C and HP15C

[15] Available only in the HP16C

[16] f LSTx in the HP10C



Numbered Registers

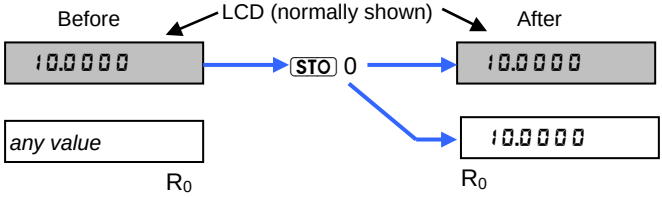
Voyagers' memory is organized in such a way that numeric data and program lines share the same memory structures without conflicting with each other. Please, refer to the tables in page 56, under Programming, for a detailed and complete view of memory organization in each Voyager.

Storing and retrieving (recalling) data is particularly useful when dealing either with large database or intermediate results. All voyagers use registers to store user data, and these registers are identified by numbers: register 0, register 1, and so on. Except for the HP10C, existing registers above register 9 are identified with the use of the dot (•) key preceding their identifying numbers: register • 0, register • 1, etc. The use of numbered data registers makes it easy to perform many tasks, and except for the HP16C, all Voyagers also allow arithmetic operations with registers contents.

Storing data

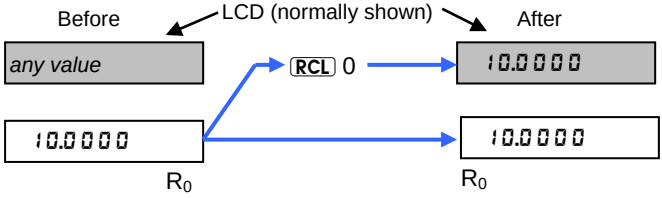
Provided that a number to be stored is in the display (thus stored in the X-register), pressing STO followed by a register identification[17] creates a copy of this number into the designated register. Current X-register contents remain and previous register contents are lost. The diagram below illustrates the action of storing number 10 in register 0 (R_0).

[17] Both the HP12C and the HP15C allow the use of specific register identifiers. The HP12C has financial keys n, i, PV, PMT and FV, each one associated with the respective financial register. The HP15C allows storing and recalling of matrix elements through A to E keys.



Retrieving data (recall)

Provided that a number is stored in a known register, pressing [RCL] followed by the register identification[18] creates a copy of this number into the X-register. Current X-register contents may either be replaced by the retrieved number or pushed into the stack, according to previous status. Current R_0 contents remain. The diagram below illustrates the action of recalling R_0 contents to the stack.



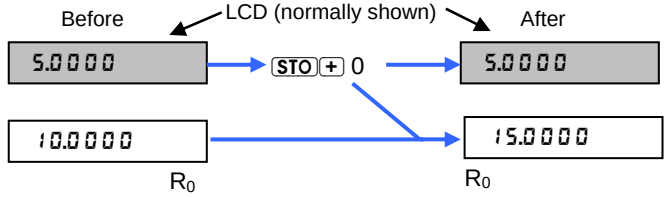
Register Arithmetic [19]

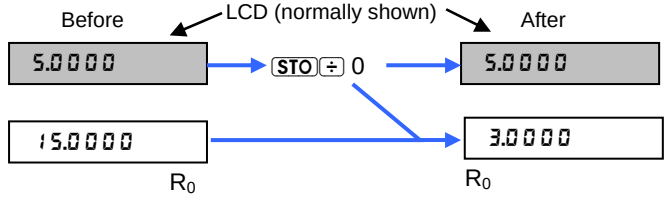
Instead of replacing the contents of a numbered register, it can be added, subtracted, multiplied or divided by the contents of the X-register. X-register contents remain, and register contents changes according to the selected operation. The diagrams below illustrate the registers contents for register addition and division. Consider that 5 has been keyed in.

---

[18] Both the HP12C and the HP15C allow the use of specific register identifiers. The HP12C has financial keys [n], [i], [PV], [PMT] and [FV], each one associated with the respective financial register. The HP15C allows storing and recalling of matrix elements through [A] to [E] keys.

[19] All voyagers but the HP16C





Keep in mind that trying STO ÷ to any register when zero (0.0000) is in the display always return Error 0 message (division-by-0 error) and no operation is performed.

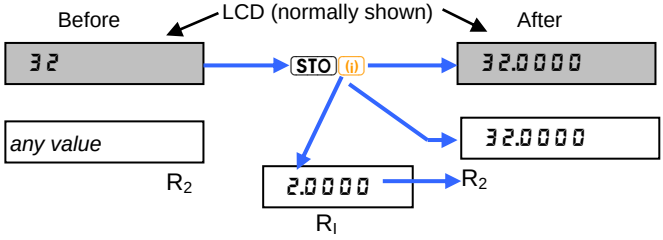
Remarks

The HP11C restricts register arithmetic to the first 10 numbered registers (R_0 to R_9), while the HP12C restricts it to the first five (R_0 to R_4). The HP15C also allows register arithmetic with RCL, and this will be covered in its specific chapter. Also, because of its extended functions and the way they apply to registers, the HP15C provides extra register handling that will be explored in its specific chapter.

Indexed (indirect) Register Access[20]

Either the HP11C, the HP15C or the HP16C allows indirect access to registers contents through a special index register, the I-register (I). Indexed storing, retrieving or arithmetic[21] is possible. Although the indirect access to registers contents may seem meaningless when direct addressing is possible, it proves to be useful in programs and necessary to access registers out of the range of the direct addressing[22].

Indexed access to registers implies the use of the index register contents to point to the register which contents will be 1) replaced by the number in the X-register or 2) retrieved to the X-register. Note that altering the index register contents is as easy as altering any other register's because the I-register behaves as an ordinary register for storing and recalling data. In order to use its contents as an index, it is necessary to use the (i) identifier. Only the integer part of the number stored in the index register is used to indirectly access other registers. The diagrams below illustrate what happens when using indirect addressing to store the number 32 in R_2 and to retrieve the number previously stored in R_0 to the X-register. Make sure to set the correct data by pressing 2 STO I 32:



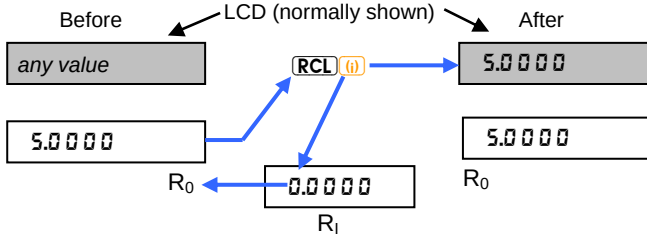
Now press 0 STO I and follow the next diagram to recall through the index register:[23]

[20] HP11C, HP15C and HP16C only

[21] Indexed register arithmetic is not available for the HP16C as well

[22] HP15C and HP16C only

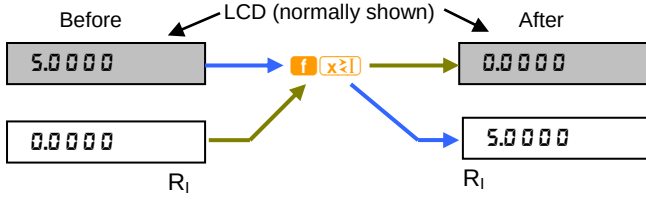
[23] The HP11C and the HP16C allow f I (f (i)) as shortcut for RCL I (RCL (i))



In order to extend indirect access functionality, g ISG and g DSE[24] allow incrementing and decrementing the contents of the index register[25] without disturbing stack registers contents. They are also design to control loops in programs.

Interchanging X-register contents with other registers

The HP11C, HP15C and HP16C offer extra functionality with the use of the index register. With f x≷I[26], X-register contents are exchanged with the index registers contents, and with f x≷(i)[27], X-register contents are indirectly exchanged with the register pointed by R_I. The following diagrams illustrate these features.



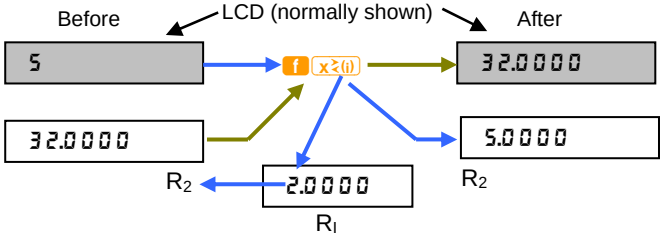
[24] g ISZ and g DSZ in the HP16C

[25] g ISG I and g DSE I in the HP15C; applies to any numbered register, too.

[26] f x≷ I in the HP15C

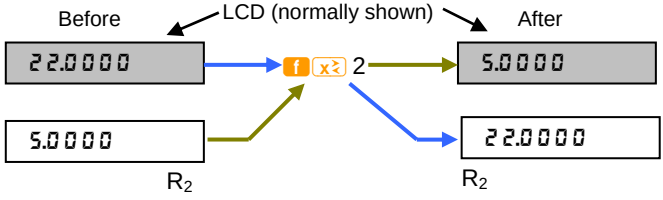
[27] f x≷ (i) in the HP15C

In order to follow the next diagram accordingly, please press 2 STO I, forcing the index register to point to R_2. Now enter 5 in the display:



Remark: HP15C's f x≷ function

The HP15C expands register interchange even more. Because it has f x≷ as a separate keystroke sequence, X-register contents can be exchanged with any of the 20 low-numbered registers directly. This means that keystrokes between f x≷ 0 and f x≷ .9 are valid keystrokes, and can also be part of any program. For instance, the HP15C allows performing the following operation without the use of the index register:

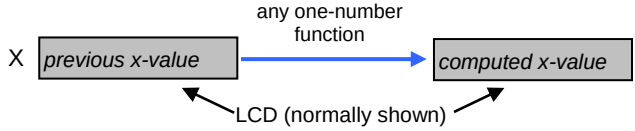


Numeric Functions

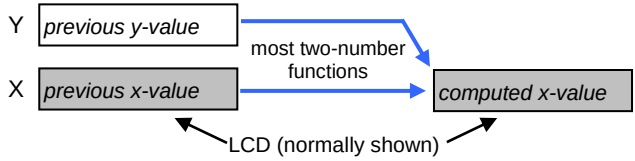
This chapter relates all numeric functions available in all voyagers, grouped into a few specific areas when necessary.

One- and Two-Number Functions

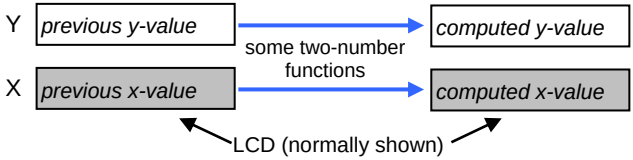
The basic concept of one- and two-number functions easily applies to any of the numeric functions viewed here, just keep in mind the general rule: executing any function (pressing its related key) immediately applies it to the arguments that must be previously placed in the relevant stack registers. The diagram below illustrates what happens when a one-number function is performed.



The diagram below illustrates what happens when the most common two-number functions are performed ([+], [−], [y^x] etc.).



For some 2-number functions, like the coordinate conversions ([→P] and [→R]) and other specific functions, the following diagram illustrates what happens.



All voyagers share the following group of six numeric functions:

[+] [−] [÷] [×]	addition, subtraction, division and multiplication
[1/x]	reciprocal
[√x]	square root

These six functions are available in all Voyagers but the HP16C:

[%]	Basic percentage	[y^x]	Generic exponential
[e^x]	Natural exponential	[LN]	Natural Logarithm
[FRAC]	Fractional part of the number	[INT][28]	Integer part of the number

[Δ%] - percent change. (HP11C, HP12C, HP15C)
[%T] - percent of a total. (only HP12C).
[RND] - rounds the internal representation to match the display. (HP11C, HP12C, HP15C)

Hyperbolic Functions (HP11C and HP15C)

Both the HP11C and HP15C compute hyperbolic functions and their inverses. Notice that the [GTO] key has [HYP] for [f]-prefix function and [HYP^{-1}] for [g]-prefix function. So:

to calculate any hyperbolic: [f] [HYP] { [SIN] [COS] [TAN] }

to calculate their inverses: [g] [HYP^{-1}] { [SIN] [COS] [TAN] }

[28] [g] [INTG] in the HP12C, to avoid confusion with financial [f] [INT]: compute interest rate. Also, the HP12C Platinum was added the [g] [x^2], available neither in the HP12C nor in the HP16C.

Exponential, Logarithmic and Trigonometric (HP10C, HP11C, HP15C)

The remaining exponential and logarithmic functions are:

[LOG] [10^x] [x^2] Logarithm base 10, generic exponent of ten and square

The Voyagers also allow trigonometric operations in any of the three basic angle modes.

[RAD] [GRAD] [DEG] Select radians, grads and degrees mode, respectively. One, and only one, of these modes is active at any time.

Either radians (RAD) or grads (GRAD) mode is announced in the calculator display. Degrees mode has no specific annunciator.

When trigonometric operations are applied, X-register contents are taken as angles in their decimal representation. Also, their inverses return angles in decimal representation.

[π] constant Pi: 'keys in' 3.14159265 4 in the X-register

[→RAD] converts from decimal degrees to radians

[→DEG] converts from radians to decimal degrees

Keep in mind that current angle mode is not affected by neither affects the calculations performed by [→RAD] or [→DEG].

Any decimal representation of angles in degrees can be converted to its degrees, minutes, seconds equivalent, and any valid degrees, minutes, seconds representation can be converted to its decimal equivalent.

[→H.MS] converts from decimal degrees to DDD.MMSSs representation

[→HR] converts from DDD.MMSSs representation to decimal degrees

These functions apply to decimal hour and hour, minute, second as well. Representation of degrees, minutes, seconds follows the ±DDD.MMSSs standard, where:

DDD integer part of the angle

MM minutes, from 00 to 59

SS seconds, from 00 to 59

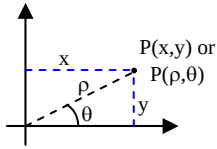
s tens of seconds, from 0 to 9

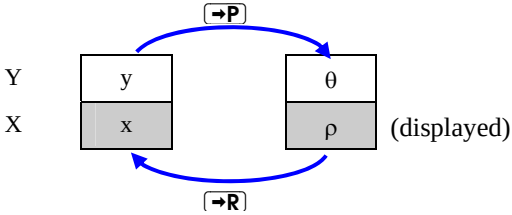
[SIN] [COS] [TAN] Respectively compute sine, cosine and tangent of the angle stored in the X-register; current angle mode defines how the argument in the X-register is interpreted.

[SIN^{-1}] [COS^{-1}] [TAN^{-1}] Respectively compute the inverses of sine, cosine and tangent of the argument in the X-register; current angle mode defines how the result should be interpreted.

Coordinate Conversions

The conversion between rectangular and polar representations depends on current angular mode. The following diagram illustrates these operations.





Make sure that the initial values are in their correct stack registers before converting (conversions exclusively apply to stack register contents).Also, keep in mind that the angular mode affects the way the angle (θ) in polar representation is either interpreted ([→R]) or computed ([→P]). If the current mode is degrees, θ is always in decimal degrees.

Statistics (all but the HP16C)

Except for the HP16C, all Voyagers deal with grouped statistics where a single or paired statistic sample can be handled.

How do Voyagers Deal With Statistical Data

These Voyagers can perform one- or two-variable statistical calculations based on accumulating statistics model. The data is entered into the calculator using the Σ+ key, which automatically calculates and stores the summations related to the input data into storage registers designated to hold them, according to the table below.

Location of the Statistics Registers

The summations of single or paired statistic data are stored in each Voyager according to a pre-defined set of registers. These locations are printed in the back of each calculator, and are summarized in the following table.

	HP10C HP11C	HP12C	HP15C
n	R_0	R_1	R_2
Σx	R_1	R_2	R_3
Σx^2	R_2	R_3	R_4
Σy	R_3	R_4	R_5
Σy^2	R_4	R_5	R_6
Σxy	R_5	R_6	R_7

Before beginning to accumulate statistics for a new set of data, the statistics registers must be cleared by pressing f CLR Σ [29] This operation also clears the contents of the stack registers. Also, keep in mind that the correct use of statistics functions is based on an understanding of memory and stack operation. The order of entry is important for most statistics calculations.

Entering statistics data

In one-variable (single) statistical calculations, enter each data point (*x*-value) into the display, then press Σ+

In two-variable (paired) statistical calculations, enter data like this:

[29] Not available in the HP10C; use f CLR REG instead, or store 0 (zero) in registers R_0 to R_5

1. Key in the *y*-value and press [ENTER];
2. Key in the *x*-value and press [Σ+].

Each time [Σ+] is pressed, the calculator does the following:

- The register holding n has its contents increased by `1`, and the result is copied into the display;
- The value in the display (*x*-value) is added to register that holds Σx;
- The value in the display is squared and added to register that holds Σx^2.
- The value in the Y-register (*y*-value) is added to register that holds Σy;
- The value in the Y-register is squared and added to register that holds Σy^2;
- The product of the *x*- and *y*-values is added to the contents of the register that holds Σxy.

Correcting Accumulated Statistics

The accumulated statistics can easily be corrected if any data is entered incorrectly. Simply key in the incorrect data point or data pair again as if it were new and press [g][Σ−] ([f][Σ−] in the HP10C) instead of [Σ+]. Then enter the correct data point or data pair and press [Σ+]. If the incorrect data has just been entered, the following sequence applies:

- Immediately after [Σ+], press [g][LSTx] ([f][LSTx] in the HP10C);
- Press [g][Σ−] ([f][Σ−] in the HP10C);
- Enter correct data;
- Press [Σ+].

Statistics Functions

Once the statistics data has been accumulated in calculator memory, many available functions can be applied to analyze the sample. Some of the statistics functions are available in all Voyagers, while others are available in one or two models.

All four models above: [Σ+] [Σ−] [x̄] [s] [ŷ,r] [n!] ([x!])[30]

These functions compute:

[x̄] - Mean of the accumulated statistics.

[s] - Standard Deviation of the accumulated statistics.

[ŷ,r] - *y*-value forecast based in linear estimation. It also computes the correlation coefficient **r**.

[n!] (HP10C, HP12C) or [f] [x!] (HP11C, HP15C)- Computes the factorial or Gamma of the value in the display.

Mean and Standard Deviation

[x̄] causes the simultaneous calculation of both means, for *x*- and *y*-values (arithmetic averages). After [x̄] is pressed, *x*-values` mean appears in the display; press [x⇄y] to display *y*-values` mean.

The same procedure above can be followed to obtain standard deviation, by simply pressing [s] instead of [x̄].

Forecasting and correlation coefficient

To preview *y*-values for a given sample in the *x*-values, simply enter the corresponding *x*-value and press [ŷ,r]. The forecasting *y*-value is returned to the display, and the correlation coefficient for the sample is returned to the Y-register (press [x⇄y] to see it)

[30] Factorial function n! is replaced by [f] [x!] (gamma function: Γ) in the HP11C and HP15C.

Factorial/Gamma Function

[n!] / [f] [x!] - Computes the factorial of the integer value in the display. The integer must not exceed 69. In the HP11C or HP15C, if the number in the display is a non-integer, i.e., has a fractional part, the Gamma function is automatically applied. Non-integers and negative numbers cause `Error 0` in the HP10C and HP12C. In the HP11C and HP15C, negative numbers cause negative overflow (`-9.9999999 99`).

The remaining statistics functions are distributed according to the following table:

	HP10C	HP11C	HP12C	HP15C
[x̂,r]	yes	-	yes	-
[x̄w]	-	-	yes	-
[L.R.]	yes	yes	-	yes
[Py,x] [Cy,x]	-	yes	-	yes
[RAN#]	-	yes	-	yes

These keys relate to:

[x̂,r] - This function is the counterpart of [ŷ,r]. It forecasts a *x*-value for a given *y*-value based on linear estimation. It also computes the correlation coefficient **r**.

[x̄w] - Weighted Mean of the X- values in current accumulated statistics.

[L.R.] - Linear Regression of the accumulated statistics.

[Py,x] and [Cy,x] - Permutation and Combination

[RAN#] - Random number register and computing

Weighted Mean (HP12C only)

To compute the weighted mean in the HP12C, the statistics registers must contain a different set of data. The weighted mean is calculated upon the number of times one condition (a number) repeats in the same sample. In this case, the more the number repeats, the closer to it the mean will be.

In order to compute the weighted mean, the input sequence of the statistical data should be:

1. key in the *number of times* a particular x-value repeats;
2. press [ENTER];
3. key in the *x*-value and press [Σ+];
4. repeat the steps above till all different X- values have been input;
5. press [g] [x̄w]

Linear Regression (HP10C, HP11C, HP15C)

Linear regression is a statistical method for finding a straight line that best fits a set of two or more data pairs, thus providing a relationship between two variables. By the method of least squares, [f] [L.R.] returns the linear coefficients *a* (slope) and *b* (Y-intercept) of the linear equation:

$$y = a \cdot x + b$$

Considering that the accumulated data has already been provided, simply press [f] [L.R.] and the Y-intercept value (*b*) is returned to the display (X-register). Press [x⇄y] and the slope (*a*) will be shown.

Probability Calculations (HP11C and HP15C)

These functions are two-number functions and they cause the stack to drop as the result is placed in the X-register. The input for permutation and combination calculations is restricted to nonnegative integers. The order of entering data must be the *y*-value before the *x*-value.

Permutations - Pressing [f] [Py,x] calculates the number of possible different arrangements of *y* different items taken in quantities of *x* items at a time. No item occurs more than once in an arrangement, and different orders of the same *x* items in an arrangement are counted separately. The expression to compute permutations is:

$$P_{y,x} = \frac{y!}{(x - y)!}$$

Combinations - Pressing g Cy,x calculates the number of possible sets of *y* different items taken in quantities of *x* items at a time. No item occurs more than once in a set, and different orders of the same *x* items in a set are not counted separately. The expression to compute combinations is:

$$C_{y,x} = \frac{y!}{x!\cdot(x-y)!}$$

The execution for these functions may last several seconds, depending on the magnitude of *x* and *y*. A flashing running is shown while computing.

Notice that even if the arguments for *x* and *y* exceed the limit of 69, the final result will be calculated and shown if it lies within the limits of the calculator representation. In any case, neither *x* nor *y* should be bigger than 9,999,999,999.

Random Number Generator (HP11C and HP15C)

f RAN# (random number) generates a random number in the range $0 \le r < 1$. This number is part of a uniformly distributed pseudo-random number sequence that uses a "seed" to initiate its random number sequence. The default value for this seed is zero, and every time a random number is generated, that number becomes the seed. A different seed can be provided at any time by storing a new seed for the random number generator with STO RAN#, thus causing a new random number sequence. Repetition of a random number seed will produce repetition of the entire random number sequence.

Notice that only the HP15C allows recalling the current seed as a normalized, positive real number with zeroed exponent of ten with RCL RAN#.

Programming Basics

What Is a Program?[31]

A program is a group of codes the calculator 'remembers' (is stored in memory) that corresponds to a sequence of keystrokes. Once stored in memory, any given program can be executed as often as needed by simply pressing one or two keystrokes. The automatic stack responds to instructions in a running program in exactly the same way it responds to an identical set of instructions executed from the keyboard. The answer displayed at the end of program execution is likewise the same as the one you would obtain by executing the instructions from the keyboard. Some powerful functions allow program control, with conditional jump and loop.

No prior programming experience is necessary to learn how to program your Voyager.

Why Write Programs?

Programs save you time on repetitive calculations. Once you have written the keystroke procedure for solving a particular problem and recorded it in the calculator, you need no longer devote attention to the individual keystrokes that make up the procedure. You can let the calculator solve each problem for you.

And because you can easily check the procedure in your program, you have more confidence in your final answer since you don't have to worry each time about whether or not you have pressed an incorrect key. The following information covers the operation of programming features available with your voyager.

Memory Resources

Each Voyager has its own amount of memory available for the user. Basically, this memory is divided into two blocks: one is dedicated to store numbers, organized in numbered registers[32], and the other is dedicated to store program instructions, organized in program steps. These blocks partially share the available memory. Right after the table below, that summa-

[31] Original text: *HP11C Owner's Handbook and Programming Guide*, with some modifications

[32] the HP12C has the Financial Registers (n, i, PV, PMT and FV) and cash flow frequency (Nj), not numbered

rizes the memory resources for each Voyager, there are a few notes that describe some particular memory arrangements.

	10C	11C	12C	15C	16C
Initial display in PRGM mode	00-	000-	00-	000-	000-
Last possible step #	79-	203-	99-	448-	203-
Permanent program steps [33]	9	63	8	0	0
Label definition	prgm steps	0→9 A→E	prgm steps	0→9 .0→.9 A→E	0→9 A→F
Index register I	no	yes	no	yes	yes
Registers (max.) [34]	10	20	20	66	25→406 [35]
Registers' numbers	$R_0 \rightarrow R_9$	$R_0 \rightarrow R_9$, $R_{.0} \rightarrow R_{.9}$	$R_0 \rightarrow R_9$, $R_{.0} \rightarrow R_{.9}$	$R_0 \rightarrow R_9$, $R_{.0} \rightarrow R_{.9}$, $R_{20} \rightarrow R_{65}$ [36]	$R_0 \rightarrow R_F$, $R_{.0} \rightarrow R_{.F}$, $R_{33} \rightarrow R_{406}$

For each model, g MEM (f MEM in the HP10C and HP16C) shows current memory organization.

	MEM default LCD	Means that available resources are:
HP10C	P-09 r-10	9 program lines, 10 registers
HP11C	P-63 r-.9	63 program lines, up to register $R_{.9}$
HP12C	P-08 r-20	8 program lines, 20 registers
HP15C	19 46 0-0	20 registers (up to $R_{.9}$) 45 registers in *common-pool*, no program lines used
HP16C	P-0 r-025	no program lines used, 25 registers[37]

[33] Steps available before the first upper-numbered register is used; this condition is described in details for the HP15C because it has a user-defined movable partition, explained in the HP15C's specific chapter
[34] Index register I is *not* taken into account for the 11C, 15C and 16C
[35] the number of available registers for the HP16C depends on the word size: 25 if WS = 64, 406 if WS ≤ 4
[36] R20 to R65 are accessible only indirectly, with the Index register I

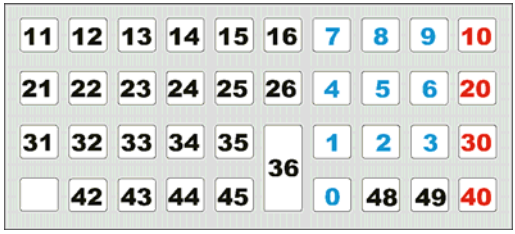
The HP15C particular memory organization is explained in details in its specific chapter.

Keycodes

Once dealing with programs and program lines, mastering keycodes enhances user's program writing and debugging skills.

Each keycode identifies the location of a key in the keyboard in terms of a row-column, two-number code. Hence, dot key [.] is coded 48 because it is located in the fourth row (first digit: 4), eighth column (second digit: 8). The following keyboard representation shows the keys with functions that are found in the same position in all Voyagers[38]. This means that the keycodes for these keys represent the same function. For instance, keycode 33 refers to [R↓] key in any Voyager.

The diagram bellow illustrates all existing keycodes associated to any Voyager. Compare it to the keyboard in your calculator and locate the corresponding key to each keycode.



The [ON] key has no associated keycode because it cannot be part of a program. Number keys [0] to [9] use a single-number code to make their identification easier. Also, in the HP16C, keycodes for the first-row, [A] to [F] keys, are respectively single-number codes A to F [39] instead of the two-number codes 11 to 16, common to the other Voyagers. Note that

---

[37] Still considers the HP16C is in floating point mode.

[38] except for [g], that does not exist in the HP10C

[39] A, b, C, d, E and F are hexadecimal numbers in the HP16C

[ENTER] has a unique two-number code: 36. Many of these keycodes may be found together in one single program line, separated or not by commas (or dots). For instance, [STO] 0 is coded 44 0 in any Voyager, but depending on which function and calculator model you are dealing with, the combined codes may vary. Consider the square-root function $\sqrt{x}$, its keystroke sequence and keycode for each model:

Model	keystroke	keycode
HP10C, HP11C, HP15C	[√x]	15
HP12C	[g] [√x]	43 21
HP16C	[g] [√x]	43 25

Simple programming

Storing programs in your calculator simply demands the knowledge of which keystrokes compose these programs. The sequence used to compute the solution manually is mostly part of the program itself.

Consider that you are given the task to compute $1/(2a+3)$ for any given value of *a*. The problem is that you must apply $1/(2a+3)$ to a table of about 100 values of *a*. Writing and using a program would surely allow this task to be easily completed.

The direct keystroke solution would be:

- Key in *a*;
- Press [ENTER];
- Key in 2;
- Press [×];
- Key in 3;
- Press [+];
- Press [1/x]

As a matter of fact, the program has actually been written! If we remove the first line ("Key in *a*;"), all of the remaining steps *are* the program itself. The first line is the only one to be performed by the user prior to execute the program for each different value of *a*.

Storing the Program (Loading)

To store/load the program, the calculator must be set to program mode and the corresponding keystrokes must be pressed in the same sequence they were pressed to solve the problem manually. Then, the calculator is set back to normal operation mode, or RUN mode, and the program is ready to be used. The following sequence shows how to store the program above in your calculator memory.

Keys	Display	Comments
P/R [40]	000 - [41]	Sets program mode; PRGM annunciator is on
f CLR PRGM	000 -	Clears all program memory(*)
ENTER	001 - 36	program's first step is entered: key-code for ENTER is 36
2	002 - 2	second step; keycode for 2 is 2
×	003 - 40	third step: keycode for + is 40
3	004 - 3	fourth step
+	005 - 20	fifth step
1/x	006 - 15	HP10C, HP11C, HP15C
or	006 - 22	HP12C
g 1/x	006 - 43 26	HP16C — last step
P/R	0.0000	Sets run mode; PRGM annunciator is off

(*) only if needed

[40] Each type has its location and keycode: HP10C, P/R; HP12C, f P/R; others, g P/R

[41] HP10C and HP12C listings show program steps with two digits (0 is a reminder)

Running the program

Make sure that each key was pressed in the given sequence and that all keycodes were generated according to the program listing above, specially the keycode for the last step. Now, with the calculator back in RUN mode, press f CLR PRGM. Remember that f CLR PRGM acts differently in program and run mode: in run mode it only sets program memory to default step 000, and in program mode it entirely clears program memory. Now enter a value for *a* and press R/S

Keys	Display	Comments
4	4	Enters testing value
R/S	0.0909	Computes $1/(2\cdot 4+3)$ with a single keystroke

Now you can easily fill the table with the remaining 99 values.

The following example was taken from the HP11C´s user manual.

Most conventional home water heaters are cylindrical in shape, and their heat loss can be easily calculated using the formula

$$q = h \cdot A \cdot T, \quad \text{where:}$$

q is the heat loss from the water heater (Btu per hour).

h is the heat-transfer coefficient.

A is the total surface area of the cylinder.

T is the temperature difference between the cylinder and the surrounding air.

Given a 52-gallon cylindrical water heater with a surface area of 30 square feet, a heat transfer coefficient of approximately 0.47 and an average temperature difference between the heater surface and surrounding air of 15 degrees Fahrenheit, how much energy is being lost because of poor insulation?

To directly calculate the heat loss of the water heater based on the temperature difference, simply press the following keys in a sequence.

- Key in 15; (temperature difference T)
- Press [ENTER];
- Key in 30; (surface area A)
- Press [x];
- Key in [•]47; (heat-transfer coefficient h)
- Press [x];

The display shows 2 1 1.5 0 0 0, which is the heat loss in BTU/hour.

The heat loss for the water heater in the preceding example was calculated for a 15° temperature difference. But what if you want to calculate the heat loss for several temperature differences? Again, you may want to perform each heat loss calculation manually. But now we know how to write a program to calculate the heat loss for any given temperature difference.

Writing the Program

The program has already been written! It comprises the same series of keystrokes you executed to solve the problem manually but the first three ones.

Loading the Program.

To load the instructions of the program into the calculator, press the following keys in the order they appear. As we know, the calculator records (remembers) the instructions as you key them in.

Keys	Display	Comments
[P/R]	0 0 0 -	Program mode. (PRGM appears.)
[f] CLR [PRGM]	0 0 0 -	Clears program memory.
[ENTER]	0 0 1 - 3 6	
3	0 0 2 - 3	The same keys
0	0 0 3 - 0	pressed
[x]	0 0 4 - 2 0	to solve

[.]	005- 48	the problem
4	006- 4	manually
7	007- 7	
[×]	008- 20	
[P/R]	211.5000	Back to Run mode

To use the program, key in the temperature difference and press [R/S]

Common Resources for Programs

Storage Registers

Either [STO] or [RCL] and related register arithmetic act in program lines as they act in keystroke sequences in run mode.

Pause and Stop

In a program line, [R/S] halts program execution and [PSE] causes the program to briefly pause for about 1 second while showing X-register contents in the display. If included as program steps, all showing operations in the HP16C ([f] SHOW [HEX], [f] SHOW [DEC], etc.) and [g] [DATE] in the HP12C also pause a running program for about 1 second to show related data.

Program Memory Navigation: [SST] , [BST] and [GTO] [.] *nnn*

[SST] (Single Step) and [BST] (Back Step) allow the user to navigate into program memory. [SST] steps to the next program line and [BST] moves back to the previous program line. In program mode, holding [SST] or [BST] down causes program lines to step automatically. In run mode, each program line and its keycode are shown for as long as you hold them down; when [SST] is released, the action associated to the program line being shown is also performed (step-by-step execution). To position program memory into line *nnn* (or *nn*), use [GTO] [.] *nnn*[42] (or *nn*), either in program or in run mode.

[42] [GTO] [CHS] nnn in the HP15C

Conditional Tests

All Voyagers offer conditional tests that allow X-register contents to be compared to zero or to Y-register contents. Conditional tests are useful only in programs because the result of their execution defines if the next program line will be executed or skipped.

Conditional tests and their availability in each Voyager are as follows:

Test	HP10C	HP11C	HP12C	HP15C	HP16C
x=0	yes	yes	yes	yes	yes
x≠0	-	yes	-	TEST 0	yes
x>0	-	yes	-	TEST 1	yes
x<0	-	yes	-	TEST 2	yes
x≥0	-	-	-	TEST 3	-
x≤0	-	-	-	TEST 4	-
x=y	-	yes	-	TEST 5	yes
x≠y	-	yes	-	TEST 6	yes
x>y	-	yes	-	TEST 7	yes
x<y	-	-	-	TEST 8	-
x≥y	-	-	-	TEST 9	-
x≤y	yes	yes	yes	yes	yes

Program Branching

GTO instruction must always be completed with an indication of 'where to go', meaning that after executing a GTO function, program execution moves away to a different program line. In the HP10C and HP12C, the complement after GTO is the number of the program line where the program is transferred to. In all of the others, a label must be placed somewhere in the program, where the user wants the execution to continue.

Conditional tests and branching become a powerful tool when writing programs with decision-level structures. Look at the fragment of a program listing below.

...	05-	instruction
...	06-	instruction
x=0	07- 42 20	*if x is equal to zero...*
GTO 18	08- 22 18	*...then go to program line 18*
	09-	*...else continue here*

If this fragment comes from an HP11C, HP15C or HP16C it would be more like this:

...	005-	instruction
...	006-	instruction
g x=0	007- 43 20	*if x is equal to zero...*(HP15C)
GTO 8	008- 22 8	*...then go to label 8*
	009-	*...else continue here*

In this example, a LBL 8 must exist somewhere in the program.

The top of program memory is signalized by program line 00 (or 000). It cannot be executed, nor contain a keycode.

HP10C and HP12C Additional Program Information

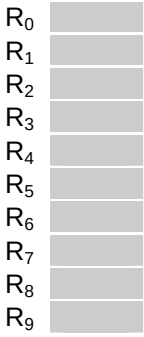
The HP10C and the HP12C have a different amount of memory and a different distribution of program and register memory. This distribution is illustrated in the diagrams below.

HP10C Memory Distribution

Initial Program Steps: 9

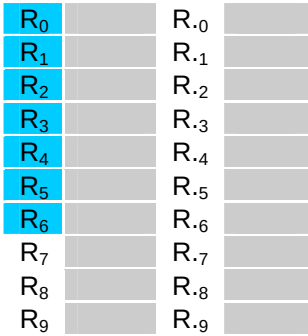
HP12C Memory Distribution

Initial Program Steps:8



Initial Registers

Permanent



Initial Registers

As program lines are added after the last available program line, each higher-numbered register is converted into seven new program lines, one at a time. In the HP10C all registers can be converted to a maximum of 79 program lines, and In the HP12C, only registers from $R_{.9}$ to R_7 can be converted to a maximum of 99 program lines. Registers R_6 to R_0 are permanent, they are never converted into program lines.

Branching in the HP10C and HP12C

In the HP10C and HP12C, [GTO] instruction is completed with a two-digit indication of the program line where program execution resumes. In certain circumstances, a [GTO] function pointing to a lower numbered line in program memory must be included. Such event may cause the program lines between the [GTO] and the lower numbered line to be executed indefinitely. To control such circumstance, conditional tests must be used. As a thumb rule, conditional tests should control [GTO] execution. Additionally, if a program line contains the keycode for [GTO] 00, program execution deviates to line 00 - and the program stops running.

Program Editing

Program lines' contents can be changed anytime you need. To replace a keycode in a program line for another:

1. Set the calculator to PRGM mode;
2. Use [SST], [BST] or [GTO][.] *nn* to set the calculator to the program line ***preceding*** the line with the keycode to be replaced
3. Now, press the correct keys to replace the old keycode; as a valid keystroke sequence is completed, the new code is shown in the correct program line.
4. Set the calculator back to run mode

If you need to add new program lines in the middle of a small program already in memory, it may be easier to enter the hole program again with the new program lines. But if you have a bigger program and want to add new lines in the middle of it, please refer to either the HP10C or the HP12C Owner's Handbook under Program Editing that the many possibilities are discussed there. In both HP10C and HP12C the same procedure applies.

Multiple Programs

It is possible to store more than one program in the HP10C and in the HP12C program memory. Provided that enough memory is available and one of the programs is already stored in memory, simply position the calculator in the last, highest-numbered program line with a valid keycode that is

used by the existing program and add a GTO 00 right after it. Now key in the second program in program memory provided that all GTO instructions point to the correct program line.

HP11C, 15C and 16C Additional Program Information

The HP11C and the HP16C have 203 bytes of program and numbered register memory for the user while the HP15C has 448 bytes. All three models also have the I register[43]. Their main differences in accessing numbered registers and program lines are explained below.

- The HP11C allows a maximum of 20 numbered registers, while the HP15C extends this limit to 64 numbered registers and the HP16C addresses a maximum of 406 4-bit registers in integer mode.
- If the HP16C is set to work with floating point numbers, memory organization for numbered registers matches the one in the HP11C.
- The HP16C addresses up to 32 numbered registers directly: $\mathbf{R_0}$ to $\mathbf{R_F}$, $\mathbf{R_{.0}}$ to $\mathbf{R_{.F}}$ (in the HP16C, A , B, C, D and F are hexadecimal numbers).
- The HP11C initial program memory allows 63 program lines to be entered prior to convert numbered registers to program lines; there is no such margin in the HP15C nor in the HP16C

Branching in the HP11C, HP15C and HP16C

In these models, GTO instruction is completed with a one-character indication (a number or a letter from A to E[44]) of the label where program execution resumes. In the HP15C, GTO and LBL also accept .0 to .9 as arguments. Not necessarily a demand, each possible definition for a GTO should have the equivalent LBL elsewhere in the program, otherwise Error 4 will be shown in any attempt of executing such 'labelless' GTO's.

Same as with the HP10C and HP12C, a GTO instruction in the HP11C, HP15C or HP16C pointing to a lower numbered line in program memory

[43] The HP11C and the HP15C also have a register to store the RAN# (Random number generator) seed. It is not suitable for storing data because its contents cannot be recalled accordingly.

[44] In the HP16C, F is also a valid reference

may cause some program lines to be executed indefinitely. Conditional tests must be used with the HP11C, HP15C and HP16C likewise.

Subroutines

When the same set of instructions occurs more than once in a program, memory space can be conserved by adding a *subroutine* containing these instructions. A subroutine is called (executed) by using a GSB function (Go to Subroutine) added by an identification of the subroutine to be executed. Subroutines start with an identification label and must have g RTN as its last step, so program execution returns to the correct program line.

GSB transfers execution to the specified label address in the same way as GTO, but when a GSB is executed in a running program, a pending *return* condition is set in the calculator. In this case, the first subsequent g RTN (return) instruction encountered by a running program will cause program execution to be transferred back to the program instruction immediately following the last GSB.

If the subroutine being executed contains a R/S instruction, the program stops and keyboard control returns to the user. If the user press R/S at this point, the program resumes till g RTN is found. Subroutines may add as many R/S as needed, but the first g RTN found on it will cause its return.

The Index register

The index register is a powerful programming tool available on these voyagers. In addition to simple storage and recall of data, the Index register can also be used for:

- Program loop counter and control functions.
- Indirectly addressing data storage registers, branches, and subroutines.

The index register control capabilities are performed through a control number that must be stored in the Index register itself.

Incrementing and Decrementing the Index Register.

In the HP11C, f ISG (increment, then skip if greater than) and f DSE (decrement, then skip if less than or equal to) will always act upon the

index register contents. In the HP15C, these same functions allow the user to specify which register they are going to act upon.

HP11C	HP15C
ISG	ISG {I, 0 to 9, .0 to .9, A to E)
DSE	DSE {I, 0 to 9, .0 to .9, A to E)

The HP16C provides g ISZ (increment and skip if zero) and g DSZ (decrement and skip if zero) to act upon the index register contents. Since f ISG and f DSE use the fraction part of the number in the index register, they would not work properly in a calculator that is mainly design to deal with integer numbers.

Both f ISG and f DSE use a control number to define their action. The integer portion of the control number is automatically altered after each loop iteration or indirect addressing operation. The decimal portion of the control number contains the parameters for altering and limiting the integer portion. The basic components of the loop control number have this format:

nnnnn.xxxyy where:

±nnnnn	is the current counter value
xxx	is the counter test value
yy	is the increment or decrement value

In yy is not explicitly defined, the calculator assumes it is 1

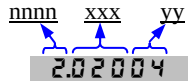
When f ISG is executed:

- nnnnn is added an amount equal to the increment (yy)
- updated nnnnn is compared to the counter test value (xxx)
- if updated nnnnn is bigger than xxx, next step is skipped; otherwise, it is executed.

When f DSE is executed:

- nnnnn is subtracted an amount equal to the increment (yy)
- updated nnnnn is compared to the counter test value (xxx)
- if updated nnnnn is less than or equal to xxx, next step is skipped; otherwise, it is executed.

As an example, consider that 2.02004 is a control number stored in the I register. This means that the current count is 2, the limit is 20 and it counts in increments of 4.



The program below is an example of using f ISG to control a loop in the HP11C (HP15C: step # 009 is f ISG I, display: 009-42, 6,25):

Keys	Display	Comments
P/R	000-	Program mode. (PRGM appears.)
f CLR PRGM	000-	Clears program memory.
f LBL A	001-42,21,11	Main start
.	002- 48	
0	003- 0	
1	004- 1	Control: 0.010
STO I	005- 44 25	stored in index register
f LBL 0	006-42,21, 0	Loop starts here
RCL I	007- 45 25	recalls index register contents
f PSE	008- 42 31	pauses to show X-register
f ISG	009- 42 6	Increment I and skip if greater
GTO 0	010- 22 0	Goes to LBL 0 (10 more times)
g P/R	0.0000	Back to Run mode

When this program is executed, I registers goes from 0 to 10 and stops looping . With these lines, it would not work properly in the HP16C. An equivalent listing for the HP16C would count from 10 to 1 and stops looping when I reaches 0 (zero). This is because the g ISZ function would always increment the integer part of the index register, compares it to zero and would never stop because I contents will never be equal to zero. With g DSZ the counting starts with 10 and stops in 1. The program below for the HP16C illustrates it.

Keys	Display	Comments
P/R	000-	Program mode. (PRGM appears.)
f CLR PRGM	000-	Clears program memory.
g LBL A	001-43,22, A	
1	002- 1	
0	003- 0	Control: 10
STO I	004- 44 32	stored in index register
g LBL 0	005-43,22, 0	Loop starts here
RCL I	006- 45 32	recalls index register contents
g PSE	007- 43 34	pauses to show X-register
g DSZ	008- 43 23	Decrement and skips if zero
GTO 0	009- 22 0	Goes to LBL 0 (10 more times)
g P/R	0 h	Back to Run mode

Taking again the example where 2.02004 is a control number stored in the index register, if f ISG is executed it updates the index register contents to 6.02004. If f DSE is executed it updates the index register contents to -2.02004.

Using the Index Register Contents

At any moment, the index register contents cam be used with some functions. They will use only the absolute value of the integer part of it.

STO I, RCL I, STO (i) , RCL (i), x≷ I and x≷(i) have been extensively discussed in this book under Fundamentals, Memory in General.

GTO I and GSB I - with these functions, the program execution is deviated to the label that matches I-register contents. Keep in mind that *there are no such things as* LBL I, GTO (i) or GSB (i)

The HP15C is the only Voyager that allows indexed control of flags.

Flags

g SF (set flag), g CF (clear flag) and g F? (flag test) are the three functions available for dealing with flags. Flags are single, numbered memory portions that can either hold a 1 (set) or a 0 (clear), or one single bit. To store a 1 in a flag, simply press g SF and add the flag number. To clear it, press g CF and specify the flag.

Testing a flag in a running program has the same effect of a conditional test, like x=0 or x>y. When g F?n is executed, where n is the number of the flag being tested, if the 'n' flag is set then next program line is executed; otherwise it is skipped. Testing a flag has no effect in run mode.

Flags are arranged like this in these Voyagerss:

	HP11C	HP15C	HP16C
All purpose, user flags	0 and 1	0 to 7	0, 1 and 2
Specific flags	none	8 - Complex mode 9 - Overflow	3 - leading zeroes 4 - carry / borrow 5 - out of range

Indexed access to flags (g SF I, g CF I and g F? I) is possible only in the HP15C.

The Specialists

The HP12C Special Resources

Calendar Functions

The HP12C provides handling for dates from October 15, 1582 through November 25, 4046 in one of two possible date formats: month-day-year[45] or day-month-year. The date format defines how numbers are interpreted when they are used in computations and displaying dates. Date format is also taken into account in bond calculations (f PRICE and f YTM). Check the table below to see how to choose the date format.

To choose...	Day-Month-Year	Month-Day-Year
...press	g D.MY	g M.DY

When the date format is set to day-month-year, the D.MY status indicator in the display is lit. If D.MY is not lit, the date format is set to month-day-year (default). The date format is reset to its default Month-Day-Year format only if Continuous Memory is reset, not each time the calculator is turned on.

Keying in dates

To key in a date with Month-Day-Year format in effect:

1. Key in the one or two digits of the ***month***.
2. Press . (decimal point key).
3. Key in the two digits of the ***day***.
4. Key in the four digits of the ***year***.

For example, April 7, 2004 is keyed in as 4.072004

[45] Month-Day-Year, M.DY, is the default date format

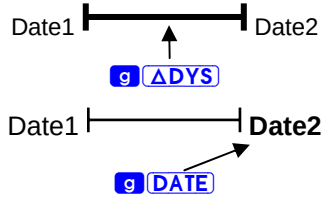
To key in a date with Day-Month-Year format in effect:

1. Key in the one or two digits of the ***day***.
2. Press [.] (decimal point key).
3. Key in the two digits of the ***month***.
4. Key in the four digits of the ***year***.

If you choose to work with Day-Month-Year format in effect, previous date (April 7, 2004) should be keyed in as 7.042004.

Calculating with Dates: g [ΔDYS] and g [DATE]

Both g [ΔDYS] and g [DATE] take into account the current date format to compute either the number of days between dates or a new date. In any case, current date format applies to both dates simultaneously. The diagrams below illustrate the calendar computations the HP12C performs.



To calculate the number of days between two given dates:

1. Key in the first date and press [ENTER].
2. Key in the second date and press g [ΔDYS].

The answer shown in the display is the actual number of days between the two dates, including leap days (the extra days occurring in leap years), if any. In addition, the HP12C also calculates the number of days on the basis of a 30-day month and stores it in the Y-register. So, you must press [x≷y] to show it and [x≷y] once again to return the original answer to the display. If the numbers are negative, then the second date occurred in the past related to the first date.

To determine the date and day that is a given number of days from a given date:

1. Key in the given date and press [ENTER].
2. Key in the number of days; (negative if the date to be computed is in the past)
3. Press [g] [DATE].

The answer calculated by the [g] [DATE] function is displayed in a special format. The numbers related to the month, day, and year are separated by current digit separators, and the sole digit at the right of the displayed answer indicates the day of the week: 1 for Monday through 7 for Sunday.

Monday	1	Wednesday	3	Friday	5	Sunday	7
Tuesday	2	Thursday	4	Saturday	6		

Resources for Financial Calculations

The HP12C has functions to compute single interest, compound interest and discounted cash flow. *Amortization and the additional functions for Bond and Depreciation Calculations will be covered in future editions of the Voyagers in Brief.*

Financial Registers

In finance, n, i, PV, PMT, and FV refer to the five TVM variables (Time Value of Money). The HP12C has five registers designated as n, i, PV, PMT, and FV, and their contents can be accessed through the first five keys on the top row of the keyboard: [n], [i], [PV], [PMT] and [FV]. These keys are the means to calculate an unknown financial value and to automatically store the result in the corresponding register.

Te following table summarizes the HP12C financial keys and their meaning

Keys	Meaning
n	stores or computes **n** (number of periods)
i	stores or computes **i** (**i**nterest rate)
PV	stores or computes **PV** (**P**resent **V**alue)
PMT	stores or computes **PMT** (**P**ay**M**en**T**)
FV	stores or computes **FV** (**F**uture **V**alue)
f AMORT	computes amortization
f INT	computes single interest
f NPV	computes net present value (discounted cash flow)
f IRR	computes internal rate of return (discounted cash flow)
g 12×	multiplies value in the display by 12 and stores in **n**.
g 12÷	divides value in the display by 12 and stores in **i**
g CFo	stores first cash flow (**C**ash **F**low$_0$)
g CFj	stores consecutive cash flows (**C**ash **F**low $_j$)
g Nj	stores number of consecutive occurrences of current cash flow
g BEG	payments in the beginning of compound periods - BEGIN active
g END	payments in the end of compound periods - BEGIN inactive

How to Use the TVM Keys

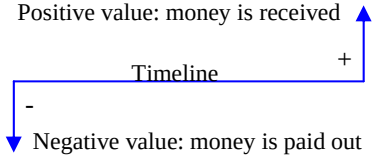
Whenever you need to use any of the five TVM keys, the same rule applies:

key in	Press	Calculator reaction
value	n, i, PV, PMT or FV	stores *value* in the corresponding variable
[nothing]	n, i, PV, PMT or FV	calculates and displays the corresponding value

A Word About the Cash Flow Diagram and Signal Convention

As a basic convention, money received is entered and displayed as a positive value, and money paid out is entered and displayed as a negative value.

Once a financial problem is taken from one of the sides of the equation - lender or borrower - it must be kept until the problem is solved. The following diagram - a Cash Flow Diagram - illustrates it.



Single Interest Calculations

The basic expression that is used to compute single interest is:

$$I_{\#days} = \frac{n}{\#\,days} \times PV \times i$$

To compute single interest calculations:

1. Enter the values for [n], [i] and [PV] in any order.
2. Press [f] [INT] ; the interest accrued on a 360-day basis is displayed.
3. To calculate the total of the principal with this interest, press [+].

Alternatively, after the second step above:

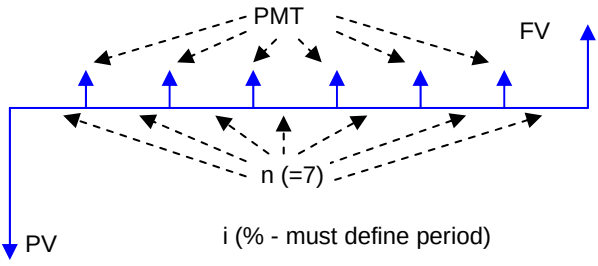
3. To display the interest accrued on a 365-day basis, pres [R↓] and [x≷y]
4. To calculate the total of the principal with this interest, press [+]

Compound Interest Calculations

The basic expression that is used to compute single interest is:

$$PV + (1 + i \times S) \times PMT \times \left[\frac{1 - (1+i)^{-n}}{i} \right] + FV \times (1+i)^{-n} = 0$$

The TVM keys are also related to the TVM variables in the generic cash flow diagram, as seen below:



Both the signal convention and the reaction to keystrokes mentioned before apply compound interest calculations. Some restrictions apply:

- **PMT** must have the same value in each occurrence
- The periods in time referred to **n** must be equal in extension.
- **n** must be positive[46]
- **i** should not be smaller than a minimum[47] to allow valid **n** calculations.

Any of the other registers associated to the variables in the above diagram may have negative, zero or positive values stored in it, but at least one of them must have its sign opposed to the others. As a reminder, the TVM keys work like this:

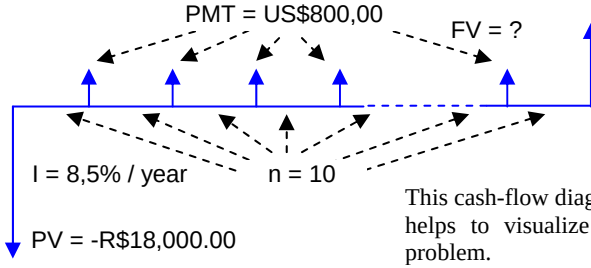
key in	Press	Calculator reaction
value	n, i, PV, PMT or FV	stores *value* in the corresponding variable
[nothing]	n, i, PV, PMT or FV	calculates and displays the corresponding value

[46] Actually, if **n** = 0, then **PV** = **-FV** regardless **i** and **PMT** values.

[47] In each valid **TVM** case, if **i** goes below a minimum value, **n** tends to infinite; if **i** is negative, this means deflation.

Having at least four of the five values as known quantities, just enter them and press the key of the fifth one so the HP12C calculates its value.

As an example, calculate how much of a US$18,000.00 loan remains after 10 months with a US$800,00 payment and an interest rate of 8.5% a year.



This cash-flow diagram helps to visualize the problem.

The following keystroke sequence calculates FV:

Keys	Display	Comments
f CLR FIN	unchanged	clears TVM registers
18000 CHS PV	- 18,000.00	Enters PV
10 n	10.00	Enters n
800 PMT	800.00	Enters PMT
8.5 g 12÷	0.7083	Enters i (% / n) [48]
FV	PUMT ING	Shown briefly (flashing)
	11,056.54	This is the remaining debit

Notice that the values for n, i, PV and PMT could be entered in any sequence.

[48] The interest rate must be related to **n**; in this case, **n** is in months and **i** is in % / year

The TVM What ... if ...'s

It is easy to test for other possibilities once you have the basic information already stored in the correct registers. If you want to go ahead in the same problem to test, say, for a different payment value, or to estimate an interest rate for a given remaining debit, the first rule is:

> *Never clear the financial registers contents unless you have concluded the current analyzes!*

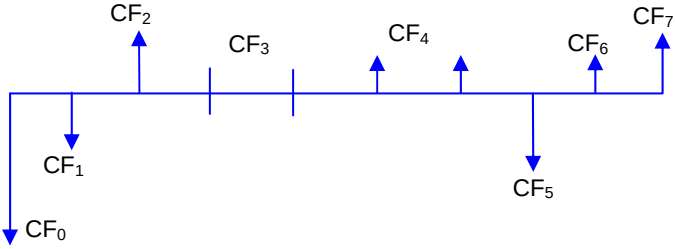
Once you press f CLR FIN, all TVM data must be entered again. So, without changing current financial registers contents, calculate the expected annual interest rate for a remaining debit of exactly R$10,500.00 in the previous example. Because all data is already there, simply enter the new value for FV and press i.

Keys	Display	Comments
10500 FV	10,500.00	Enters new value for FV
i	PMT -709	Shown briefly (flashing)
	0.34	the new interest rate (monthly)
12 x	4.10	the new annual interest rate

Discounted Cash Flow

The HP 12C memory structure allow the user to store and analyze samples with up to 21 non equal, consecutive cash flow values occurring at regular intervals. If two or more equal, consecutive cash flows occur, it is possible to solve problems involving more than 21 cash flows by storing these occurrences as grouped cash flows. The two functions provided by the HP12C for discounted cash flow analysis are NPV (Net Present Value) and IRR (Internal Rate of Return).

Discounted cash flow problems can also be represented by cash flow diagrams, like the one presented below.



The basic difference in this diagram compared to the ones for compound interest is the occurrences of different payments, now designated as CF_1, CF_2... (CF is short for Cash Flow). As a common reference, each cash flow is generally designated as CF_j. CF_0 is the initial cash flow which occurs only one time. The indexes for each cash flow relate only to their sequence in the cash flow diagram.

The HP12C stores each cash flow occurrence, from CF_0 CF_{19}, in one of the available storage registers, from R_0 to R_{19}. Provided that all numbered registers are available (i.e. no more than 8 program lines have been entered in program memory, if any), if CF_{20} occurs (last of 21 cash flows) it will be stored as FV (in FV Register). When grouped cash flow occur, the number of occurrences of each sequentially repeating cash flow (designated N_j, in the range from 1 to 99) is stored in a separated share of memory. This part of memory is ready to hold the 21 N_j's corresponding to each CF_j.

How to Store Cash Flow Values

The main keys to store cash flow values are g CFo, g CFj and g Nj. Once you know the variables:

1. Make sure there are enough registers to store cash flow data.
2. If any, identify grouped cash flow and check for their N_j.
3. Clear all data in registers with f CLR REG (strongly recommended)
4. Key in first cash flow (CF_0) and press g CFo
5. For each subsequent cash flow, key in its value and press g CFj
6. If grouped cash flow, key in its corresponding N_j and press g Nj
7. Repeat steps 5 and 6 until last cash flow

The HP12C keeps track of which cash flow will be stored through TVM register [n]. In fact, each time [g][CFj] is pressed the HP12C:

1. Increments current [n] contents in one unit;
2. Stores display contents in register pointed by [n] contents.

Neither [g][CFo] nor [g][Nj] change [n] register contents. At any moment, you can check for the last entered CF_j (or N_j) by recalling [n] contents with [RCL][n].

Editing Cash Flow Data

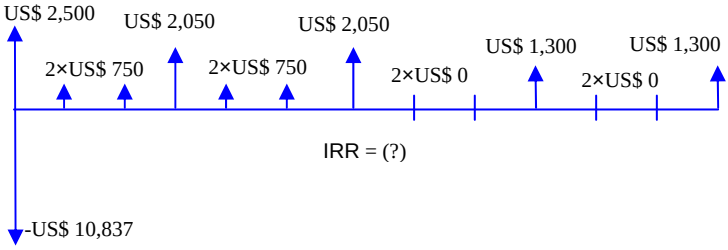
To correct a wrong CF_j entry:

1. If applicable, check [n] contents so you may continue entering cash flow data after updating the wrong entry.
2. Key in the index of the cash flow that immediately precedes the one to be corrected.
3. Press [n] to store it.
4. Key in the correct value to be stored.
5. Press [g][CFj]

In step 2 above, it is necessary to enter preceding index because the HP12C increases [n] contents prior to store the value in the display when [g][CFj] Is pressed. If you need to correct a wrong N_j entry:

1. If applicable, key in the index of the N_j to be corrected and press [n].
2. Key in the correct value to be stored.
3. Press [g][CFj]

If you are correcting the value of N_j for current CF_j, step 1 is not necessary.



Considering the cash flow diagram in previous page, the following key-stroke sequence enters all data and calculates the internal rate of return.

Keystroke	Display	Comments
f CLR REG	0.00	All registers = 0; N_j = 1
2500 ENTER	2500.00	
10873 −	-8,373.00	CF_0
g CF0	-8,373.00	CF_0 stored in R_0
750 g CFj	750.00	CF_1 stored in R_1
2 g Nj	2,00	N_1
2050 g CFj	2,050.00	CF_2 stored in R_2
750 g CFj	750.00	CF_3 stored in R_3
2 g Nj	2.00	N_3
2050 g CFj	2,050.00	CF_4 stored in R_4
0 g CFj	0.00	CF_5 stored in R_5
2 g Nj	2.00	N_5
1300 g CFj	1,300.00	CF_6 stored in R_6
0 g CFj	0.00	CF_7 stored in R_7
2 g Nj	2.00	N_7
1300 g CFj	1,300.00	CF_8 stored in R_8
f IRR	running	Wait for about 30 sec.
	2.71	IRR = 2.71 %

Viewing Cash Flow Data

At any moment you may check cash flow data. Just be aware of the fact that the fastest way to do so is viewing them from the last to the first one:

1. Make sure [n] contents points to the last cash flow you want to check.
2. If applicable, check grouped cash flow first with [RCL] [g] [Nj]
3. Press [RCL] [g] [CFj]
4. Repeat 2 then 3 (or only 3) until CF_0 is shown in the display

If you press [RCL] [g] [Nj] or [RCL] [g] [CFj] right after checking the contents of CF_0, Error 6 will be shown in the display. This is because when the calculators executes [RCL] [g] [CFj], a copy of the contents of CF_j that is pointed by **n** is brought to the display and [n] contents are diminished in one unit. This way, next time you press [RCL] [g] [CFj] the preceding CF_j will be brought to the display. [RCL] [g] [Nj] will always bring current N_j contents without changing [n] contents.

The following keystroke sequence shows all data from previous example

Keystroke	Display	Comments
[RCL] [n]	8.00	Last register used: j = 8
[RCL] [g] [Nj]	1.00	N_8
[RCL] [n]	8.00	Unchanged: j = 8
[RCL] [g] [CFj]	1,300.00	CF_8
[RCL] [n]	7.00	Automatic decrease: j = 7
[RCL] [g] [Nj]	2.00	N_7
[RCL] [n]	7.00	Unchanged: j = 7
[RCL] [g] [CFj]	0.00	CF_7
[RCL] [n]	6.00	Automatic decrease: j = 6
[RCL] [g] [Nj]	2.00	N_6
[RCL] [n]	6.00	Unchanged: j = 6
[RCL] [g] [CFj]	1,300.00	CF_6
…	…	…
[RCL] [n]	0.00	Automatic decrease: CF_0
[RCL] [g] [CFj]	-8,373.00	

Notice that the automatic decrease happens after RCL g CFj, and not after RCL n. The Internal Rate of Return (IRR) and the Net Present Value (NPV) are well known tools for financial analysis and will not be discussed here. The value of NPV indicates the result of the investment and IRR is the rate of return at which the discounted future cash flows equal the initial cash outlay. NPV equals zero when the discount rate matches IRR. Thus, if we calculate the NPV right now, we would find zero (0.00) or a value too close to that. In order to make sense of it, we could calculate the NPV for a given, different discount rate. So, let's consider an actual rate of 2,60% instead of the 2,71% found previously and then, we calculate the NPV.

Keystroke	Display	Comments
2.6 i	2.60	Known discount rate
f NPV		Wait for about 10 secs
	49.64	NPV=US$ 49.64

If the presented cash flow is observed, for a given discount rate of 2.60%, this analysis concludes for a barely profitable business.

The HP15C Special Resources

There are four major areas where the HP15C can be used to perform advanced calculations: complex numbers, matrix arithmetic, equation solve and numerical integration. Complex numbers and matrix arithmetic demand knowledge in stack and register manipulations. *Equation solve and numerical integration demand basic knowledge in programming and will be covered in future editions of the Voyagers in Brief.*

Memory Requirements

Unlike any other Voyager, the HP15C has part of its memory, grouped in registers, dedicated to temporarily store information related to its specific resources. This group of registers is named *common segment*, are the HP15C temporarily uses some registers of this segment when performing operations related to these resources. The HP15C memory can be mapped like this:

Numbered and Indirect Accessed Registers	Common Segment: imaginary stack, matrix elements, integral/solve temporary regs.	Program Lines

As shown under Programming Basics, memory resources in all Voyagers can be checked by pressing g MEM (f MEM in the HP10C and HP16C). In the HP15C, default memory status is:

These numbers directly reflect the HP15C´s memory map:

- 19 refers to the last available numbered register (from **R**$_0$ to **R**$_{.9}$)
- 46 means 46 registers available in the common segment
- 0-0 means no program line has been written in memory

The 0-0 has a particular interpretation. The first number tells how many registers are actually occupied with programs and the second number tells how many bytes are left till another register will be needed.

In all Voyagers, the HP15C included, the part of the memory occupied with programs (program memory) grows automatically as you insert new pro-

gram lines. This means that the number of available registers in memory is reduced as program memory uses more of it. In the HP15C, only registers from the common segment are reclaimed when program lines are added. The number of available numbered registers can only be changed by the user. To define how many numbered registers must exist in memory:

- Key in the number of the last register you need in the X-register;
- Press f DIM (i)

Next time you press g MEM, this is the number you will see first, and RCL DIM (i) returns this value to the X-register as a valid number. Remember that you cannot define more numbered registers than the available memory, because neither program lines nor registers already in use by special features will be available. Also, R_0 and R_1 are always available, so if you try 0 f DIM (i), the calculator will actually set it to 1 and give no error message.

The number of the last numbered register that can be defined at any moment is the sum of the first two numbers you see, in the leftmost side of the display, when you press g MEM. Also, if the common segment has not enough registers for a particular resource, you will not be able to use it. Check the table below:

Resource	Registers needed
Complex Mode	5
Matrix	1 for each element of each matrix
Solve	5
Integration or Solve + Integration	23

In some circumstances, Solve and Integration may be performed simultaneously. In these cases, they will both share five registers and will demand the same amount of registers that Integration would demand alone.

Complex Numbers

The HP15C handles complex numbers in the form $a+bi$, where a is the real part, b is the imaginary part of it and i is $\sqrt{-1}$. This is possible by using a complex stack, that is created when the complex mode is activated. The complex stack is composed of two parallel, four-register stacks (and two LASTX registers), one for the real parts and another for the imaginary parts of the numbers. It behaves the same way as the stack used in ordinary calculations.

The imaginary part of the stack is created once Complex mode is activated by doing one of the following:

- executing f [I] or f [Re≥Im]; or
- setting flag 8, the Complex Mode flag.

When the HP15C is in Complex mode:

- c annunciator is lit
- flag 8 is set
- five registers of the common segment are allocated to the imaginary stack

Complex Numbers and the Complex Stack

Both real and imaginary parts of a complex number must be defined, even if equals zero, prior to enter it in the stack. There are a few ways to enter a complex number through the keyboard:

1. Key in the *real part* of the number into the display.
2. Press [ENTER].
3. Key the *imaginary part* of the number into the display.
4. Press f [I].

This sequence is the most common and usual way to enter a complex number in the stack. If the HP15C is not in complex mode, it will be activated and the c annunciator will be displayed. Because it uses the (ENTER) key,

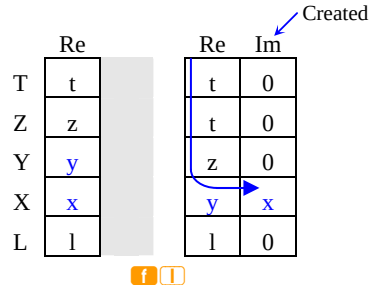
current Z- and T-register contents are lost. It is possible to preserve Z-register contents by following this keystroke sequence:

1. First, key in the *imaginary part* of the number into the display.
2. Press f [Re>Im].
3. Press ←.
4. Then, key the *real part* of the number into the display.

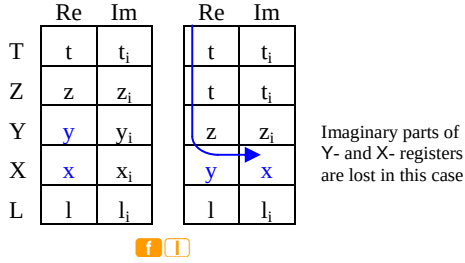
In this particular sequence , complex mode will be activated in step 2 if the HP15C is not yet in complex mode. In most cases, depending on previous stack contents and enabling state, the display shows 0.0000 after step 2. Because f [Re>Im] was executed, ← is used to disable stack lift and allow the real part to occupy the real X-register without lifting the stack.

If, by any reason, the contents of the stack registers must be preserved, it is possible to enter a complex number in the X-register and keep the rest of the stack unchanged. Simply press ← and follow the previous keystroke sequence from the first step and ahead.

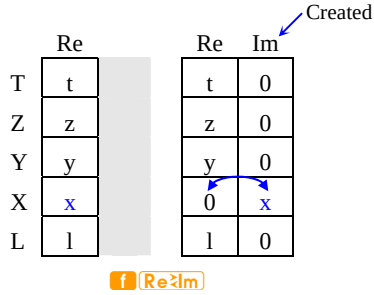
What happens if f [I] is executed when *complex mode is not active:*



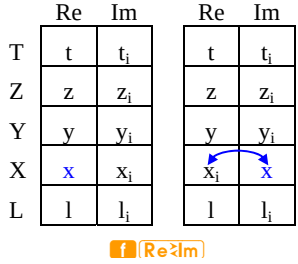
What happens if f [I] is executed when *complex mode is active:*



What happens if f Re≷Im is executed when *complex mode is not active:*



What happens if f Re≷Im is executed when *complex mode is active:*



The operations in the table below act only in the real part of the numbers in the stack, regardless the complex mode being activated or not:

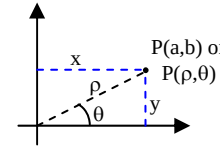
Stack	2-number	1-number
CLx ← CHS	Σ+ Σ−	% Δ% Cy,x Py,x DIM (i)

The functions in the table below act simultaneously in both real and imaginary parts of the numbers in the stack:

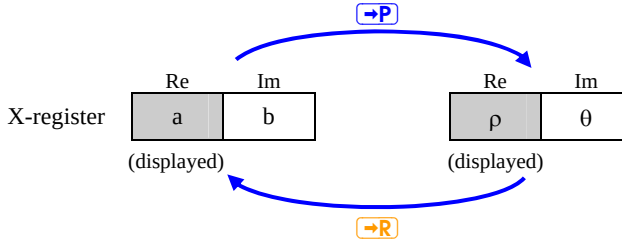
Stack	2-number	1-number
ENTER x≷y LSTx R↓ R↑	+ − × ÷ y^x	$1/x$ 10^x e^x x^2 $\sqrt{x}$ LN LOG RND ABS INT FRAC x! SIN COS TAN SIN^{-1} COS^{-1} TAN^{-1} →R →H.MS →RAD →P →HR →DEG

Notice that the storage registers suffer no alterations when the calculator is in complex mode.

Polar ↔ Rectangular Conversions



When dealing with complex numbers, the conversion between rectangular and polar representations occurs only in the X-register, and it depends on current angular mode. The following diagram illustrates these operations.



Keep in mind that these are the only complex functions that consider an argument in polar representation. All other functions assume that the arguments are in rectangular representation. Also, current angular mode affects the way the angle (θ) in polar representation is either interpreted (→R) or computed (→P). If the current mode is degrees, θ must always be in decimal degrees prior to execute →R. To view the

Briefly Viewing the Imaginary Part of the X-register

The imaginary stack contents are not directly visible, but the imaginary part of the X-register can be shown briefly by pressing f (i). It remains visible for as long as f (i) is held. If you press f (i) while the HP15C is not in complex mode, Error 3 is shown in the display. Use f Re⇄Im only to interchange the imaginary and the real contents of the X-register.

Two “Tricky” complex operations

Some operations involving complex numbers need special attention.

Changing Signal

Regardless of the current mode, CHS changes only the signal of the real X-register. To change the signal of both parts of a complex number you can either:

multiply it by -1... (T-register contents are lost in this case)

1. press 1 CHS
2. press ×

... or change the signal of each one of the parts, one at a time.

1. press CHS.
2. press f Re⇄Im.
3. press CHS.
4. press f Re⇄Im once more.

Storing and Recalling

STO and RCL also act only upon the real X-register. So, either to store or to recall complex numbers, two numbered registers will be necessary. For example, to store a complex number already in the X-register into registers **R_4** (real part) and **R_5** (imaginary part) and to recall it back:

1. press STO 4. (stores real part)
2. press f Re≥Im.
3. press STO 5. (stores imaginary part)
4. press f Re≥Im once more. (returns to original complex number)

In order to recall it back to the X-register at any moment, you can choose one of the two known forms of entering a complex number:

1. press RCL 4 (recalls the real part).
2. Press RCL 5 (recalls the imaginary part)
3. Press f I. (creates complex number 4 + 5i)

Again, the contents of both T- and Z- registers are lost. The other option, to preserve Z-register contents, is:

1. press RCL 5. (recalls imaginary part)
2. press f Re≥Im.
3. press ←.
4. press RCL 4. (recalls real part)

Matrices

Basics

The HP15C allows operations with five definable matrices: A, B, C, D and E. Each matrix has a *descriptor*, a pointer to its contents that is used by the calculator's operating system to define upon which matrix functions apply. All five descriptors exist at any moment, even if no elements are associated to a particular matrix.

The HP15C does not support complex matrices, it performs calculations with real-number representations of complex matrices. Matrices functions use only the contents of the real stack, even if complex mode is active.

Defining a Matrix Dimension

Because the descriptors exist at any moment, you only need to define their *dimension* (#rows × #columns) by placing the number of rows (#r) in the Y-register and the number of columns (#c) in the X-register and pressing

f DIM {A to E}. Only the integer part of both X-register (#c) and Y-register (#r) will be used, whether they are negative or have fractional part. As an example, to define matrix $\mathbf{A_{2\times2}}$, simply press 2 ENTER 2 f DIM A. If you need to change a matrix dimension at any time, simply enter the new values for #r and #c in the stack and press f DIM {A to E}. New elements are zeroed and exceeding elements will be neglected. Existing elements are rearranged in a row-preference sequence if matrix grows.

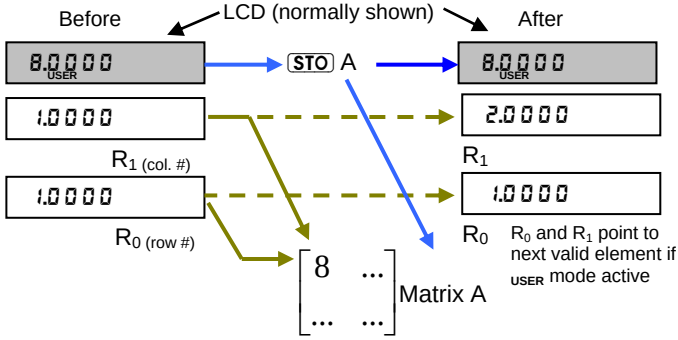
Checking the Descriptor

A matrix descriptor can be recalled and checked by pressing RCL MATRIX {A, B, C, D or E} at any moment. To recall the descriptor of matrix $\mathbf{A_{2\times2}}$ simply press RCL MATRIX A and the display shows:

The letter identifies the matrix, first number is #r and second number is #c.

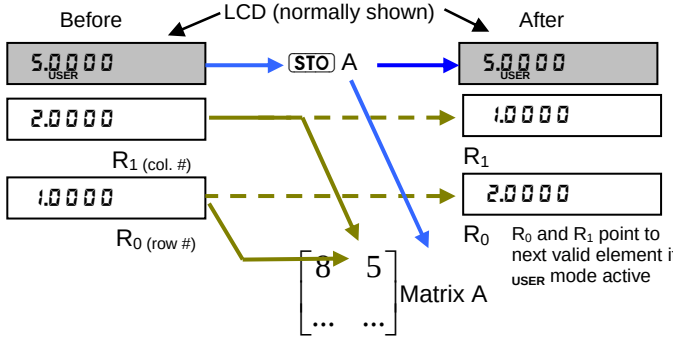
Filling a Matrix with Data

Each time a matrix is created, its elements are zeroed. To fill any matrix with custom data it is necessary to define which element of the matrix will store the new data. As a general rule, $\mathbf{R_0}$ and $\mathbf{R_1}$ are the row/column pointers, meaning that their contents point to the element to be accessed. The diagram bellow illustrates the storing of number 8.0000 in element $\mathbf{a_{1,1}}$ of matrix $\mathbf{A_{2\times2}}$: (USER active)



Notice that when USER is active, the contents of both $\mathbf{R_0}$ and $\mathbf{R_1}$ are updated to point to the next valid element of the current matrix (in this case, $\mathbf{a_{1,2}}$) each time its elements are accessed either through STO or through RCL. If USER mode is not active, their contents remain unchanged after either STO or RCL are performed.

Consider now that number 5.0000 must be stored in $\mathbf{a_{1,2}}$ right after previous operation. Simply press 5 STO A. The diagram below illustrates what happens, then:

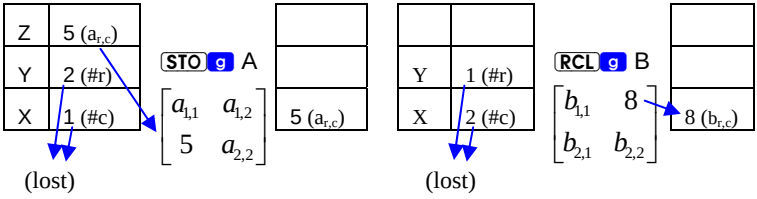


Again, as the USER mode is still active, the contents of both $\mathbf{R_0}$ and $\mathbf{R_1}$ are updated to point to the next valid element of the current matrix (in this case, $\mathbf{a_{2,1}}$) after STO or through RCL. If STO or RCL are performed while $\mathbf{R_0}$ and $\mathbf{R_1}$ are pointing to the last element of a matrix and USER mode is active, $\mathbf{R_0}$ and $\mathbf{R_1}$ will be updated to point back to the first element of the current matrix.

Keep in mind that the HP15C does not keep track of the matrix you are working with while storing or recalling data. The contents of $\mathbf{R_0}$ and $\mathbf{R_1}$ may point to an element that does not exist in a particular matrix, and if you try to access a nonexistent element with STO or RCL, the calculator displays Error 3.

Alternate Data Entry in Matrices

At any moment, you can use the stack registers Z, Y and X to store, or Y and X to recall a matrix element without changing **R_0** and **R_1** contents. For example, to store 5 as element $a_{2,1}$ in matrix A, fill the stack with the data shown in the diagram at the left side, and follow the keystrokes.



In the diagram at the right side, the element $b_{1,2}$ (consider $b_{1,2}$ = 8.0000) in matrix B is recalled. In both cases, LASTx contents are not affected

Checking for R_0 and R_1 Consistency

Whenever you want, it is possible to simply show the contents of **R_0** and **R_1** without disturbing either the matrix or the stack contents. Whether USER mode is active or not, after pressing [RCL], [STO], [RCL] [g] or [STO] [g], simply ***press and hold the matrix definition key*** ([A] to [E]) you want to check the pointers for. If the element does not exist, Error 3 is displayed and you may release the key. If the element exists, for as long as you hold the key the display shows: (consider previous example)

b 1,2

If the pointers point to the correct element and you want the operation to be performed, you may simply release the key you are pressing ([A] to [E]) right after seeing the indexes below. But if you hold the matrix definition key for more than about one second, the calculator aborts the operation and shows:

null

In this case, no operation is performed and data in all registers remain.

How to Apply Functions to Matrices

The most common way to perform calculations with matrices is:

1. place the descriptors of the matrices you are working with and any additional arguments in the corresponding stack registers, if applicable.
2. define where to store the result with f RESULT {A to E}, if applicable; some functions apply to the X-register descriptor.
3. press the keystroke sequence of the function you need to apply.
4. check the resulting elements in the result matrix defined in step 2.

Matrix A is the default result matrix and in a few cases, the function to be applied does not allow a matrix with original arguments to be the result matrix. Also, some matrix functions do not use the descriptors in the X- or Y- registers, instead they apply straight to the result matrix. Check the following tables and the *HP15C User Manual* for further information.

Keystrokes	Result in X-register	Effect on matrix specified in X-register	Effect on Result Matrix
CHS	No change.	Changes sign of all elements	None. ‡
1/x	Descriptor of result matrix.	None. ‡	Inverse of specified matrix. §
f MATRIX 4	Descriptor of transpose.	Replaced by transpose.	None. ‡
f MATRIX 7	Row norm of specified matrix*	None.	None.
f MATRIX 8	Frobenius or Euclidean norm of specified matrix.†	None.	None.
f MATRIX 9	Determinant of specified matrix.	None. ‡	LU decomposition of specified matrix. §

* The row norm is the largest sum of the absolute values of the elements in each row of the specified matrix.

† The Frobenius or Euclidean norm is the square root of the sum of the squares of all elements in the specified matrix.

‡ Unless the result matrix is the same matrix specified in the X-register.

§ If the specified matrix is a singular matrix (that is, one that doesn't have an inverse), then the HP-15C modifies the *LU* form by an amount that is usually small compared to round-off error. For [1/x], the calculated inverse is the inverse of a matrix close to the original, singular matrix. (Refer to the *HP-15C Advanced Functions Handbook* for further information.)

Useful Matrix Operations and Facts

[f] [MATRIX] 0 All matrices to zero dimension; free registers become available at common segment

[f] [MATRIX] 1 $1 \rightarrow \mathbf{R_0}$ $1 \rightarrow \mathbf{R_1}$ (does not change stack contents)

Operation	Elements of Result Matrix *	
	Matrix in Y-Register Scalar in X-Register	Scalar in Y-Register Matrix in X-Register
[+]	Adds scalar value to each matrix element	
[×]	Multiplies each matrix element by scalar value	
[−]	Subtracts scalar value from each matrix element.	Subtracts each matrix element from scalar value.
[÷]	Divides each matrix element by scalar value.	Calculates inverse of matrix and multiplies each element by scalar value.

* Result matrix may be the same matrix in the X- or Y-register.

With matrix descriptors X and Y in both X- and Y-registers, and being their dimension compatible:

[+] - calculates Y + X and places result in result matrix

⊟ - calculates Y - X and places result in result matrix

⊠ - calculates Y × X and places result in result matrix

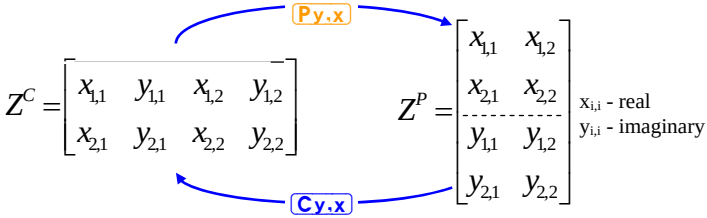
÷ - calculates **R = X^{-1} ×Y** and places R in result matrix (Y can be the result matrix but X); commonly used in linear systems solutions.

f MATRIX 5 - calculates Y^T × X and places result in result matrix

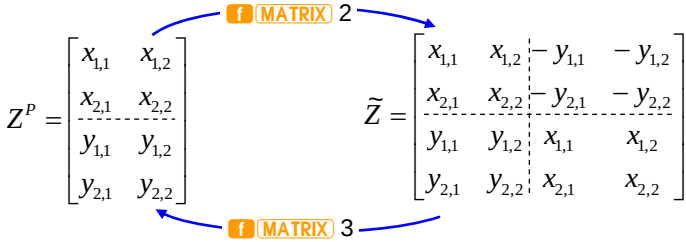
f MATRIX 6 - calculates residual R-Y×X; R is the result matrix, Y and X are descriptors in Y- and X-registers

Representing Complex Matrices with Real Matrices

The HP15C does not support complex numbers as matrix elements, so it represents complex matrices - Z^C - by using real matrices where complex numbers are split in real and imaginary parts, separately stored in each element of the real matrix. The complex matrix is not suitable to perform calculations, so we need two additional representations. One of them, designated as *partitioned* - Z^P -, has its elements reorganized in real and imaginary 'sections', separately. The diagram below illustrates the matrices and the functions to transform one representation into the other.



In order to obtain complex results from calculus with real matrix representations, the algorithms in the HP15C use a conjugated representation $\overline{Z}$ of the partitioned matrix, which is mirrored with conjugated complex elements. The conjugated form is suitable to generate complex results while performing real matrix calculations. The transformations from the partitioned to the conjugated forms and conversely are illustrated in the diagram below.



The user must identify which representation is being used at any moment because, in terms of internal structure, the HP15C stores all of them as real matrices. Only the user knows which matrices represent complex-number structures.

Performing Complex Calculations with Matrices

As a primary rule, the main procedure to perform complex calculations with real matrices is:

1. Enter complex data in a real matrix in complex or partitioned form; complex form Z^C is suitable to enter complex data.
2. If matrix is in complex form, use f Py,x to convert it into its partitioned form Z^P.
3. Use f MATRIX 2 to convert the partitioned form into its conjugate form $\tilde{Z}$ (mirrored with conjugate elements).
4. Perform the necessary complex operations;
5. Use f MATRIX 3 to convert the result matrix in $\tilde{Z}$ form to partitioned form Z^P.
6. If you want, convert the result matrix in partitioned form into complex form with g Cy,x; complex form is suitable for viewing complex data

Important Observation about Programming

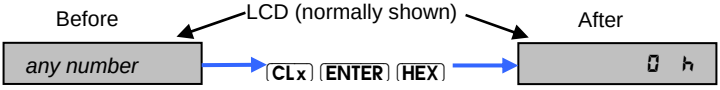
USER mode defines functionality and how STO{A to E} and RCL{A to E} are recorded as program lines. If USER is active, their keycodes will be added a 44, like this: 001 44 11 (STO A). When such lines are executed, pointers (**R0** and **R1** contents) will advance automatically as well.

The HP16C Special Resources

The distinguished features offered with the HP16C are:

- Integer arithmetic in four number bases (hexadecimal, decimal, octal, and binary), operating in 1's or 2's Complement or Unsigned mode.
- A variable word size, selected by the user, up to a maximum of 64 bits.
- Logical operators and bit manipulations.
- Floating-point decimal arithmetic *(covered in future editions).*

To match display presentations from now on, your HP16C should be in any of the four integer modes. Considering that it is in floating point mode, the following keystroke sequence sets hexadecimal mode:



All common features for the stack, memory and programming already presented to all Voyagers are applicable to the HP16C in integer mode as well.

Integer Mode and Number Base Modes

Integer Mode strictly relates to four number base modes: hexadecimal ([HEX]), decimal ([DEC]), octal ([OCT]) and binary ([BIN]). The HP16C handles fractional decimal numbers only in Floating-Point Decimal mode.

The four number base modes used by the HP-16C in Integer mode are mainly for purposes of display and digit entry. At the right of the eight-digit display, an h, d, o, or b indicates the present number base mode. Pressing any of the four number base keys establishes Integer mode, and Hexadecimal is the default mode. Regardless of the current number base mode, the internal representation of numbers is always binary.

Only valid digit keys are active and interpreted accordingly. In Hexadecimal mode, keys [A] to [F] respond as digit keys to enter numbers A, B, C, D, E and F and they appear in the display as A, b, C, d, E and F.

Temporary Display: [SHOW]

To temporarily view the displayed value in another base, press [f] [SHOW] {[HEX], [DEC], [OCT] or [BIN]}. The displayed number remains for at least one second or for as long as you hold down the number base key.

Word Size

The HP16C will work with words (data units) up to 64 bits long, and the default is 16 bits. To specify a word size (1_{10} to 64_{10}), key it in and press [f] [WSIZE]. It must be keyed in according to the current base number, only its absolute value is used and Error 2 results if you attempt to specify a word size larger than 64. If current word size limits the choices for a new word size, key 0 [f] [WSIZE] which acts as 64 [f] [WSIZE] so you can choose a new one. After [f] [WSIZE] is executed the stack drops.

Setting a smaller word size may truncate a word in the stack registers, leaving only the least significant bits. Restoring previous word size or setting a larger one will not retrieve stack registers contents nor the sign bit of a negative number. The effect on storage registers is different.

Window Display and Scrolling

Any of the stack registers can hold up to 64 binary digits, depending on the word size. The display can only show X-register contents, as it happens in all other Voyagers, and its contents are shown only eight digits at a time regardless current integer mode. To allow the viewing of all digits of any 64-bit number in the X-register, they are split into eight-digit representations and shown in what we call *windows*.

Window 0 shows the eight rightmost, least significant digits of the X-register. Leading zeros are not normally displayed, and a period on the left and/or right side of the h, d, o, or b indicates hidden, existing digits.

When keying in a new integer, the most significant digits move off the left end of the display and into window 1.

Pressing f WINDOW {0 to 7} displays each window at a time. Error 1 results if you try to specify a window number greater than 7. Any of the eight existing windows can be shown at anytime you choose it, regardless current integer mode or existing digits (blank window if empty). At each new entry into the X-register the display returns to window 0 (default).

g < and g > move the number representation ***one digit at a time*** into the display for viewing purposes (X-register contents are not changed). A period can appear both on the left or right sides of the base indicator if the current window is not at one end of the number. g < cannot move window 0 contents to the left, but g > can push window 7 contents to the right and leave blanks in the left of the display. Scrolling is "reset" to window 0 when a bit manipulation or mathematical function is executed.

Consider the 64-digit number FF00FF00FF00FF00$_{16}$ in the X-register, hexadecimal mode, and follow the keystroke sequence below.

Keystroke	Display	Comments
40 f WSIZE	0 h	Sets WSIZE = 64$_{10}$
FF00FF00FF00FF00	FF00FF00 .h	Enters X-register data
g >	0FF00FF0 .h.	Notice the periods in h
g >	00FF00FF .h.	
g >	F00FF00F .h.	
f WINDOW 1	FF00FF00 h.	Notice the period in h
f WINDOW 2	h.	Display blanks;
g <	F .h.	Notice the periods in h

Complement Modes and Unsigned Mode

In the binary representation of a signed number, the leftmost or most significant bit with respect to word size serves as the sign bit: 0 for plus and 1

for minus. The HP-16C provides three conventions for representing numbers: 1 's Complement mode, 2 's Complement mode (default), and Unsigned mode. The numbers stored in the calculator memory and stack registers are not changed when a representation is changed. In 1´s or 2´s complement mode negative numbers are displayed with a minus sign only in decimal mode.

1's Complement Mode

Pressing f SET COMPL 1'S will set 1's Complement mode, which is formed by complementing all bits of the negative number in the X-register. One's Complement accommodates an equal number of positive and negative numbers, but has two representations for zero: 0 and -0.

2's Complement Mode

Pressing f SET COMPL 2'S will set 2's Complement mode, which is formed by complementing all bits of the negative number in the X-register and adding 1. In 2's Complement there is just one representation for zero, but there is always one more negative number than positive number represented.

Unsigned Mode

Pressing f SET COMPL UNSGN will set Unsigned mode, which uses no sign bit and leaves the most significant bit to add magnitude. Pressing CHS in unsigned mode causes the number in the X-register be the 2's complemented and Flag 5 to be set as a reminder that the true result is out of the range of the unsigned mode.

The following table summarizes how the complement modes affect the decimal interpretation of all possible 4-bit patterns (word size 4).

Decimal Interpretation of 4-Bit Binary

Binary	1 's Complement Mode	2's Complement Mode	Unsigned Mode
0111	7	7	7
0110	6	6	6
0101	5	5	5
0100	4	4	4
0011	3	3	3
0010	2	2	2
0001	1	1	1
0000	0	0	0
1111	-0	-1	15
1110	-1	-2	14
1101	-2	-3	13
1100	-3	-4	12
1011	-4	-5	11
1010	-5	-6	10
1001	-6	-7	9
1000	-7	-8	8

Flags

The HP-16C has three user flags (0, 1, and 2) and three system flags (3, 4, and 5). These six flags behave the same as the HP11C and HP15C flags for setting, clearing and testing. Flags 4 and 5 can be automatically cleared or set as a result of some operations and functions.

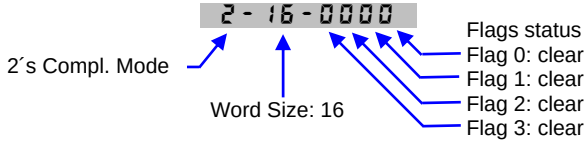
- Flag 3 controls the display of leading zeros; if set they are shown. Leading zeros are always suppressed in Decimal and Floating-Point modes. In any integer mode, they are shown according to word size.
- Flag 4 is set or cleared (and the C annunciator) according to a carry or borrow occurrence.
- Flag 5 is set or cleared (and the G annunciator) according to any out-of-range occurence.

Machine Status: STATUS

Pressing f STATUS will temporarily show a special group of digits and separators meaning:

- current complement mode
- current word size
- status of the four lower-numbered flags[49]

The following illustration represents default display when f STATUS is pressed:



The status display remains for at least 1 second or for as long as you hold the STATUS key down. Word Size is always shown in decimal mode.

Arithmetic and Bit Manipulation Functions

Integer arithmetic operations and bit manipulation functions can only be performed in Integer mode.

Carry and Out-of-Range Conditions

The execution of certain arithmetic and bit manipulation operations can result in a carry and/or an out-of-range condition, which depend on the particular function being executed.

Flag 4: Carry Flag

Flag 4 (carry flag) is set if the carry bit is 1 and cleared if the carry bit is 0. All results from shifting, rotating, and arithmetic operations listed below affect flag 4 and the C annunciator in Integer mode.

[49] flags 4 and 5 have their own C and G annunciators.

SL	RL	RLn	+ (carry)
SR	RLC	RLCn	− (borrow)
ASR	RR	RRn	÷ (reminder ≠ 0)
	RRC	RRCn	DBL÷ (reminder ≠ 0)
			√x (reminder ≠ 0)

Flag 5: Out-of-Range Flag

Flag 5 and (G annunciator) is set if the correct result of an operation cannot be represented in the current status. The functions below affect flag 5 status in Integer mode:

+	−	×	÷	ABS	CHS
DBL×	DBL÷				

+, −, ×, ÷ and FLOAT also affect flag 5 in Floating-Point Decimal mode and in other circumstances.

Arithmetic Functions in Integer Mode

+, −, × and ÷ can also be performed using integers in any base. ÷ always truncates fraction part of the result, if any.

Addition and Subtraction in 1's Complement Mode.

In 1's Complement mode the result of an addition is affected by the occurrence of a carry (1 is added to the result) and the result of a subtraction is affected by the occurrence of a borrow (1 is subtracted from the result). Both cases set flag 4

The Carry Flag During Addition and Subtraction

For all complement modes, the carry flag will be set whenever a binary addition results in a carry "out of the most significant bit or a binary sub-

traction results in a borrow into the most significant bit. Otherwise, it is cleared.

The Out-of-Range Flag

Arithmetic results that cannot be shown in the current word size and complement mode set Flag 5. For [÷] this occurs only in 2's Complement mode when the largest possible negative number is divided by -1.

Remainder After Division [RMD] and Square Root [√x]

[f] [RMD] calculates the remainder of a division, or |y| MOD |x|. The sign of the result matches the sign of y (dividend). The [√x] function calculates the square root of the number in the X-register. The fractional part of the square root is truncated. Flag 4 is set if the remainder ([RMD]) or fraction part ([√x]) of the result is not zero, otherwise is cleared.

Negative Numbers and Complementing

Changing Signs. The [CHS] function (change sign) will change the sign of the number in the X-register, forming the complement (1's or 2's). In Unsigned mode, [CHS] forms a 2's complement and sets flag 5, too[50].

Absolute Value. Pressing [g] [ABS] converts the number in the X-register to its absolute value, forming the 1's or 2's complement of a negative number.

If the X-register holds the largest possible negative number in 2's Complement mode, either [CHS] or [g] [ABS] causes flag 5 to be set.

Logical Operations

NOT

[f] [NOT] - 1-number function that complements (inverts) the values of all bits in the binary number in the X-register.

[50] A reminder that a negative number is outside the range of Unsigned mode

AND

f AND - 2-number function (*the logical product*) that compares each corresponding bit in two words and returns 1 only if both operand bits are 1.

OR

f OR - 2-number function (*the logical sum*) that compares each corresponding bit in two words and returns 0 only if both operand bits are 0.

EXCLUSIVE OR

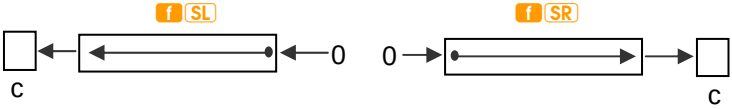
f XOR - 2-number function (*the logical difference*) that compares the corresponding bits in two words and yields a 1 only if two corresponding bits are different.

Shifting and Rotating Bits

Shifting and rotating operations cause the bits of a word to be moved left or right. Except with LJ (left-justify), Flag 4 reflects the status of the last out-of-the-word bit, defined by current word size.

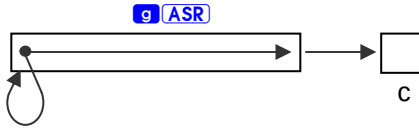
Shifting Bits

Logical Shifts. Pressing f SL (shift left) or f SR (shift right) moves all the bits of the word in the X-register one bit to the left or right. The new bits generated at the opposite end of the word are always zeros.



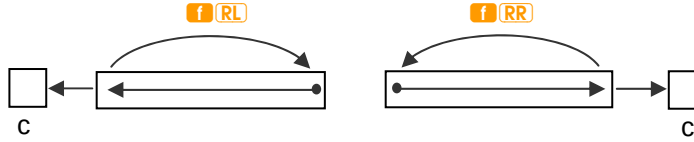
Left-Justify. g LJ left-justifies a word within its word size. Places the left-justified word in the Y-register and the number of bit-shifts necessary to left- Justify it (count) in the X-register. g LJ does not affect Flag 4.

Arithmetic Shift Right - if the calculator is in 1's or 2's complement mode, g ASR moves (shifts) bit pattern in the X-register one bit to the right *and regenerates its sign bit*. In Unsigned Mode, g ASR operates like SR. Carry bit follows any bits that are shifted out of the word in X-register

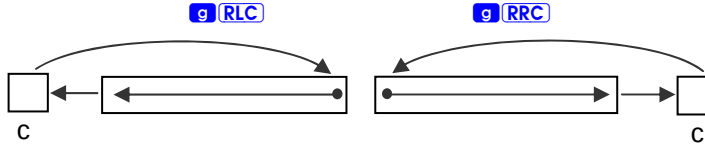


Rotating Bits

Rotation ("circularly shift") - f RL (rotate left) and f RR (rotate right) rotates X-register contents one bit to the left or right. The Flag 4 follows state of rotated bit.



Rotation Through the Carry Bit - g RLC (rotate left through carry) and g RRC (rotate right through carry) differ from f RL and f RR functions respectively because Flag 4 does not hold a copy of the rotating bit, it actually holds 'the rotating bit'. Flag 4 is virtually 'added' to the bit pattern while rotating its contents.



Rotating More Than One Bit at a Time. The four functions above have their 'n-time' versions: f RLn, f RRn, g RLCn and g RRCn work the same as their 'one-time' counterparts except that the bit pattern is taken from the Y-register and the number of times they rotate is taken from the X-register. The stack drops, placing the result in the X-register.

Setting, Clearing, and Testing Bits

g SB (set bit) and g CB (clear bit) functions set (1) or clear (0) any of the existing bits in a word, while g B? tests any of them, in a manner

analogous to flag-testing even as a program step. To set, clear, or test a specific bit in a word:

- The word containing the specific bit must be in the Y-register.
- The magnitude of the number in the X-register locates the bit

Bits are numbered in the same way they are signified. After any of these bit-wise test functions, stack drops.

Masking

f MASKL (mask left) and f MASKR (mask right) create left- or right-justified strings of 1 bits with magnitude (≤ wordsize) specified in the X-register (# of 1´s). The mask replaces X-register contents (stack unchanged). Although the mask is justified, it can be placed in the middle of the field of a number by using other functions (shift, rotate, multiply by 2^n)

Bit Summation

g #B (number of bits) sums the bits in the pattern stored in the X-register and returns that value back to the X-register, saving the bit pattern in LASTx register. No stack lift occurs. In word sizes 1 and 2, the result must be interpreted in Unsigned mode.

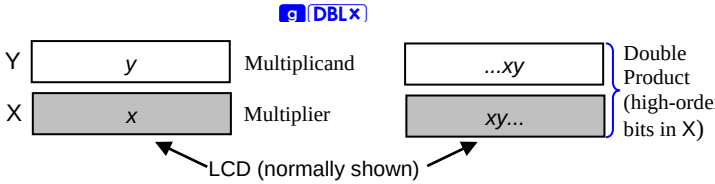
"Double" Functions

Double functions in the HP16C allow the exact calculation of a product double the given word size and the exact calculation of a quotient and remainder from a dividend of double word size.

Because the result is often a number split in parts that fit in current word size, to obtain meaningful double numbers as results in Hexadecimal and Octal modes the word boundary (which is based on the numbers of bits) must not "split" a digit. Therefore, you should specify a compatible word size: a multiple of four in Hex mode and a multiple of three in Octal mode.

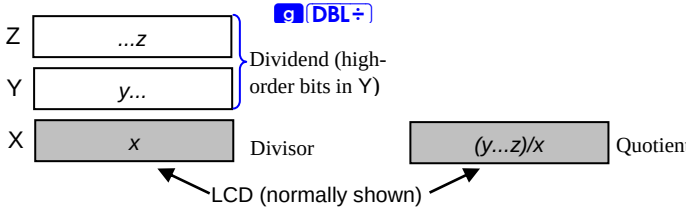
Double Multiply

g DBL× multiplies two single-word quantities in the X- and Y-registers to yield a double-word, right-justified result in the X- and Y-registers. The stack contents during this operation are shown below.



Double Divide

g DBL÷ computes the quotient of a dividend of double word size split in the Y- and Z-registers, divided by a single-word divisor in the X-register. The stack drops twice, placing the single-word result in the X-register.



Error 0 occurs if the answer cannot be represented in a single word size. Flag 4 is set if the remainder is not equal to zero.

Double Remainder

g DBLR operates like g DBL÷ except that the remainder is returned instead of the quotient. It is determined as for the f RMD (single reminder) function, with the sign of the result matching the sign of the dividend. Error 0 occurs if the quotient exceeds 64 bits.

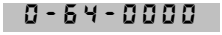
Example: Applying Double Divide[51]

Compute the quotient of $\frac{5714AF2_{16}}{7E14684_{16}}$ to 16 hexadecimal places.

Although the result is a fraction, this problem can be solved in Integer mode by first finding the integer quotient of

$$\frac{5714AF2\overbrace{000\cdots00}^{16\text{ zeroes}}_{16}}{7E14684_{16}}$$

and then placing a decimal point to the left of the result (thereby dividing the result by 2^{64}). Use [DBL÷] to accommodate a numerator this large.

Status:  (unsigned mode, word size=64)

Keystroke	Display	Comments
[HEX] 40 [f] [WSIZE]	0 h	Sets WSIZE = 64_{10}
0 [ENTER]	0 h	
5714AF2 [ENTER]	5714AF2 h	double-sized dividend
7E14684 [g] [DBL÷]	7E985d8C C h	Carry bit set
[f] [WINDOW] 1	b0d06F6A C h.	

Result is $B0D06F6A7E985D8C_{16}$, so the actual answer is:

$0.B0D06F6A7E985D8C_{16}$

[51] Copy of the original example from the **HP16C Owner's Handbook**, p. 54

APPENDIXES

Error Messages

Error messages are identified by the `Error #` display, where # is the number that identifies the error type. The following table describes the errors and their application to each model. The reference for each error message can be found in the back-label of any Voyager (except the HP12C).

Message	Description	Obs.
`Error 0`	Improper Math Operation	x/0, LN(0), √(-), etc.
`Error 1`	Storage Register Overflow Improper Mtx. Operation Improper Reference Digit	HP10C/11C/12C HP15C HP16C
`Error 2`	Improper Stat. Operation Improper Bit #	HP10C/11C/12C/15C HP16C
`Error 3`	Statistics Reg Unavailable Improper Register # IRR (needs estimate)	HP10C HP11C/15C/16C HP12C
`Error 4`	Improper Memory Call	register, label / prgm line
`Error 5`	Improper Register # Subroutine Level too deep Compound Interest	HP10C HP11C/15C/16C HP12C
`Error 6`	Improper Flag # Storage Registers Error Invalid Register Contents	HP11C/15C HP12C HP16C
`Error 7`	IRR (no roots for IRR) Recursive SOLVE or ∫f	HP12C HP15C
`Error 8`	Calendar No Root from SOLVE	HP12C HP15C
`Error 9`	Service Needed	ON/× ON/÷
`Error 10`	Insufficient Memory	HP15C only
`Error 11`	Invalid Matrix Argument	HP15C only
`Pr Error`	Memory Contents Lost	power loss, ON/−

Annunciators

In the beginning of this book you were shown how to perform some tests in your calculator by following simple keystroke sequences. The first test leads to the following display: (all models)

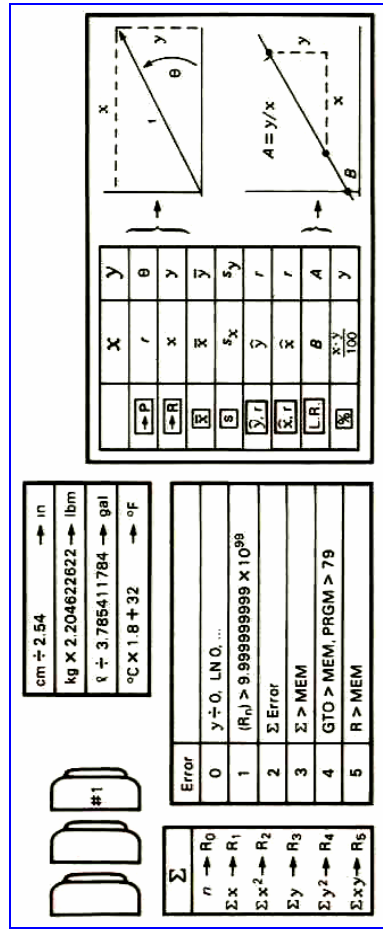
which means that the calculator is working fine. The set of characters that appear in the button of the display are named annunciators, and they signalize (announce) particular operating conditions at the moment you are using the calculator. Their meaning is summarized in the table below.

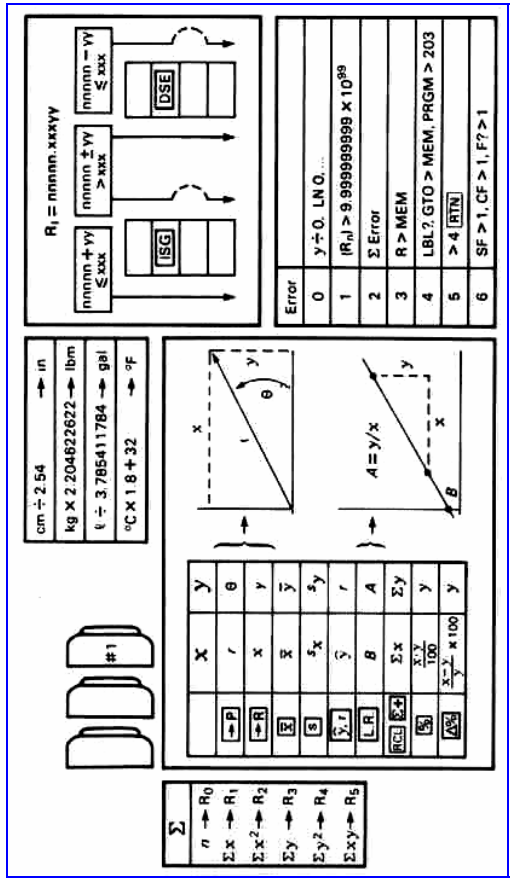
Annunciator	Meaning
*	normally off or flashing; when this anunciator flashes, batteries should be changed for new, fresh ones.
f	f key pressed, prefixed action active
g	g key pressed, prefixed action active (all but HP10C)
USER	USER keyboard active (HP11C/HP15C only)
BEGIN	Payment in advance; (HP12C only)
G	Out-of-Range condition (HP16C only)
GRAD	grads angle mode set (HP10C/HP11C/HP15C)
RAD	radians angle mode set (HP10C/HP11C/HP15C)
D.MY	dd.mmyyyy date mode selected (HP12C only)
C	HP12C: consider compound interest in odd period HP15C: complex mode active HP16C: carry/borrow condition occur
PRGM	program mode active

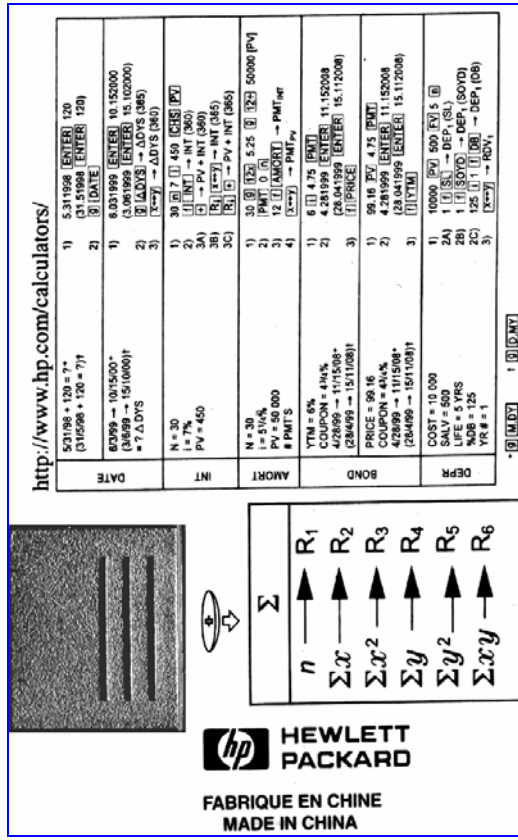
If there is no specific mention to a particular model, the annunciator has the same meaning in all models.

Back Labels

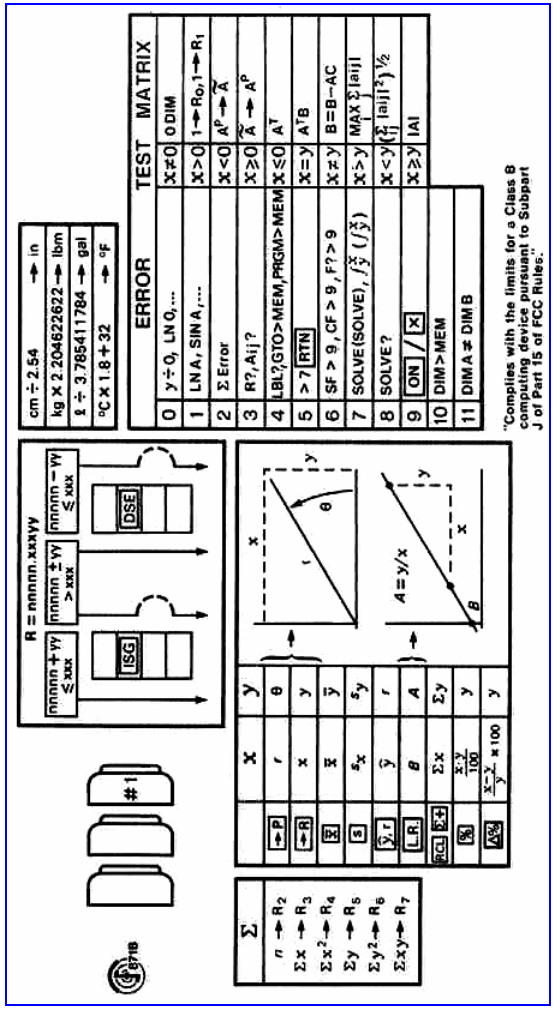
HP10C

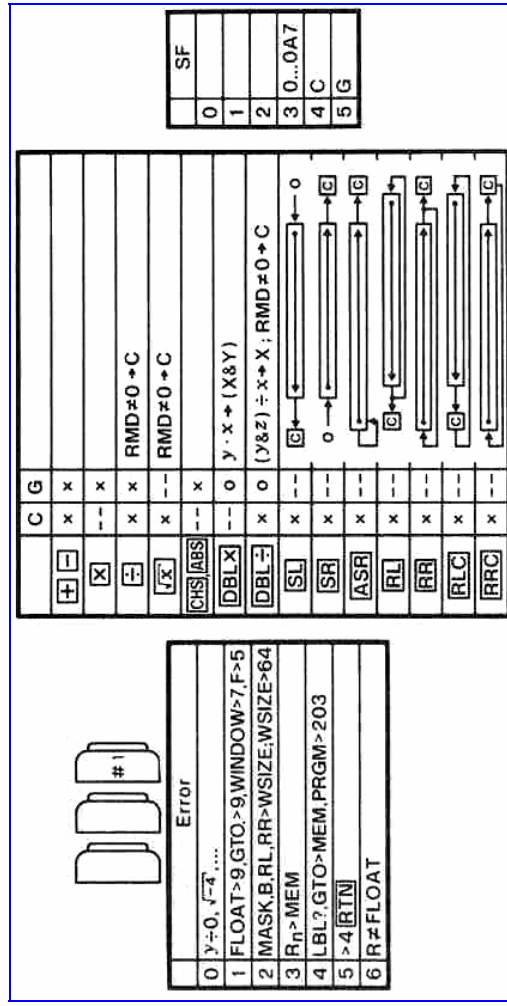



Σ
n → R0
Σx → R1
Σx² → R2
Σy → R3
Σy² → R4
Σxy → R5
#1
cm ÷ 2.54 → in
kg × 2.204622622 → lbm
ℓ ÷ 3.785411784 → gal
°C × 1.8 + 32 → °F
x
y
→P
→R
Σ+
Σ-
LR
r
θ
x̄
ȳ
sx
sy
x̂, ŷ
A
Σx
Σy
x·y/100
(x−x̄)/x̄ × 100
A=y/x
Ri = nnnnn.xxxyy
nnnnn + yy ≤ xxx
ISG
nnnnn ± yy > xxx
DSE
nnnnn − yy ≤ xxx
Error
0 y ÷ 0, LN 0, ...
1 |Ri| > 9.999999999 × 10^99
2 Σ Error
3 R > MEM
4 LBL?, GTO > MEM, PRGM > 203
5 > 4 RTN
6 SF > 1, CF > 1, F? > 1



Note that this is from the newer calculators, with one single button-type battery; the battery-door was kept in the drawing for reference.


cm ÷ 2.54 → in
kg × 2.204622622 → lbm
ℓ ÷ 3.785411784 → gal
°C × 1.8 + 32 → °F
ERROR TEST MATRIX
0 y÷0, LN 0,... X≠0 0 DIM
1 LN A, SIN A,... X>0 1→R0, 1→R1
2 Σ Error X<0 AP→Ã
3 R?, Aij? X≥0 Ã→AP
4 LBL?,GTO>MEM,PRGM>MEM X≤0 AT
5 >? RTN X=y A*B
6 SF > 9, CF > 9, F? > 9 X≠y B=B−AC
7 SOLVE(SOLVE), f(x)/f(y) X>y MAX Σj|aij|
8 SOLVE? X<y (Σi|aij|²)½
9 ON / X X≥y |A|
10 DIM>MEM
11 DIM A ≠ DIM B
"Complies with the limits for a Class B computing device pursuant to Subpart J of Part 15 of FCC Rules."
R = nnnnn.xxxyy
nnnnn + yy ≤ xxx
nnnnn ± yy > xxx
nnnnn − yy ≤ xxx
ISG
DSE
#1
Σ
n → R2
Σx → R3
Σx² → R4
Σy → R5
Σy² → R6
Σxy → R7
x
y
→P r θ
→R x y
x̄ ȳ
sx sy
ŷ,r r
L.R. θ A
RCL Σ+ Σx Σy
% x·y/100 y
Δ% (x−y)/y ×100 y
A = y/x
r
θ
x
y
A
β


Error
0 y÷0, √-4, ...
1 FLOAT>9,GTO.>9,WINDOW>7,F>5
2 MASK,B,RL,RR>WSIZE,WSIZE>64
3 Rn>MEM
4 LBL?,GTO>MEM,PRGM>203
5 >4 RTN
6 R≠FLOAT
C G
+ −
× ×
÷ ×
√x ×
CHS ABS ×
DBL× o
DBL÷ o
SL ×
SR ×
ASR ×
RL ×
RR ×
RLC ×
RRC ×
RMD≠0 → C
RMD≠0 → C
y · x → (X&Y)
(y&z) ÷ x → X ; RMD≠0 → C
SF
0
1
2
3 0...0A7
4 C
5 G

1st Edition