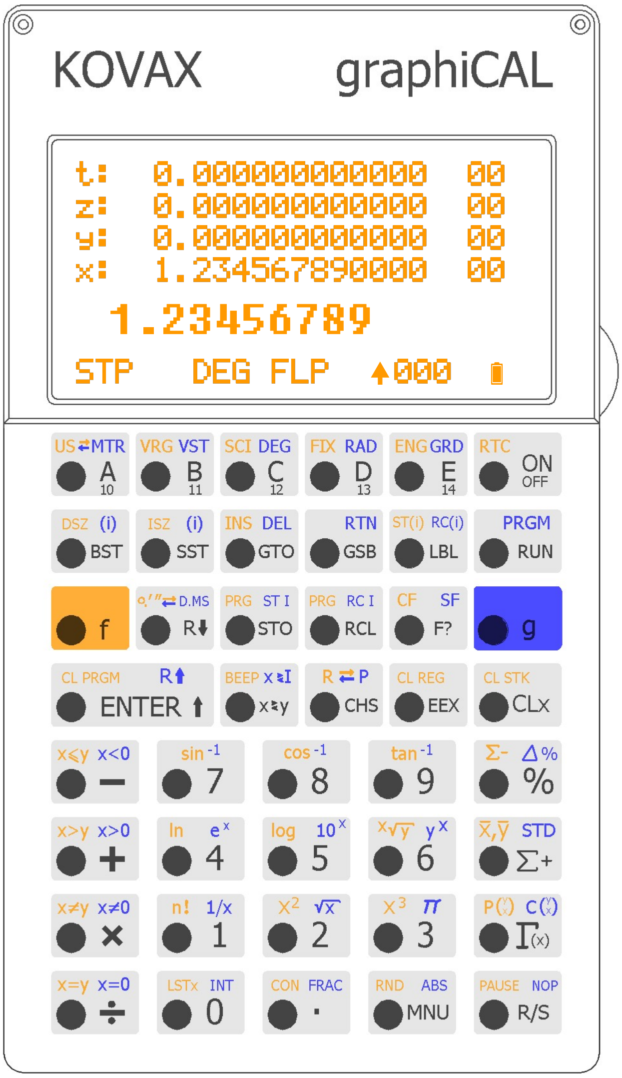
graphiCAL

Designed and made by Kovax

3D Case designed and made by Charly

Short Form Owner's Handbook and Programming Guide

This guide is based on HP-67 description of the same title.



Original size

Program memory

Prog_RAM

000	GTO 000
001	GTO 000
002	GTO 000
003	GTO 000
004	GTO 000
005	GTO 000
006	GTO 000
007	GTO 000
008	GTO 000
.	
.	
.	
.	
.	
.	
.	
.	
.	
.	
.	
.	
250	GTO 000
251	GTO 000
252	GTO 000
253	GTO 000
254	GTO 000
255	GTO 000

Default settings

Addressable Storage Registers

Directly accessible registers: R00–R14

Indirectly accessible registers: R15–R28

R00	
R01	
R02	
R03	
R04	Σx
R05	Σx^2
R06	Σy
R07	Σy^2
R08	Σxy
R09	n
R10 (RA)	
R11 (RB)	
R12 (RC)	
R13 (RD)	
R14 (RE)	
R15	
R16	
R17	
R18	
.	
.	
.	
R27	
R28	
LASTx	
I-REG	
FLAGS	

Automatic Memory Stack

T
Z
Y
X

X is always displayed
Exception: program editor

Display formats

f VRG — g VST

```
1.23456789
STP  DEG FLP  ▲000 ▮
```

```
FLG: 0 0 0 0 1 0 0 0
IND: 0.0000000000 00
LSx: 0.0000000000 00
R00: 0.0000000000 00
  1.23456789
STP  DEG FLP  ▲000 ▮
```

```
t: 0.000000000000 00
z: 0.000000000000 00
y: 0.000000000000 00
x: 1.234567890000 00
  1.23456789
STP  DEG FLP  ▲000 ▮
```

Default display — Display Registers (Navigation by thumb-wheel.) — Display Stack

```
STP f DEG FLP ▲000 ▮
RUN g RAD SCIn
      GRD ENGn
          FIXn
```

Annunciator line

STP Proram stopped RUN Program is running

f g Prefix keys

DEG RAD GRD Angle in Degree, Radian & Grads

FLP SCIn ENGn FIXn Display notations (n:0…9)

▲: Stack lift enable ▼: Stack lift disable (Useful info at debugging)

000 Program pointer (000…255)

▮ Battery voltage level

f or g US MTR

```
mil  < - > mm
inch < - > cm
feet < - > m
yard < - > m          <
nmile< - > km
smile< - > km
sm/h < - > km/h
inch² < - > cm²
```

US/Imperial < > metric conversion

Navigation by rotating the thumb-wheel.

Selection by pushing the thumb-wheel.

g PRGM

```
100   1
101   ENT               <
102   1
103   +
104   LSTx
105   x<>y
106   +
107   GTO    104
```

Program editor display

(Here: counts the Fibonacci sequence)

Navigation by rotating the thumb-wheel.

The markings of the keys and their corresponding functions

(Where the connection is not obvious.)

Key marking	Function	The associated text in the editor
A B C D E	Label designators. When preceded by LBL	LBL A (B, C, D, E)
BST	**B**ack **ST**ep	Cannot be recorded in program
SST	**S**ingle **ST**ep	Cannot be recorded in program
GTO	**G**o **TO**	GTO
GSB	**G**o to **S**u**B**routine (Call a subroutine.)	GSB
LBL	**L**a**B**e**L**	LBL
RUN	**RUN** Exit from program editor	Cannot be recorded in program
R↓	**R**oll Down ↓	ROL↓
STO	**STO**re	STO
RCL	**R**e**C**a**L**l	RCL
F?	**T**est **F**lag	F?
CHS	**CH**ange **S**ign +/-	CHS
EEX	**E**nter **EX**ponent	EEX
CLx	**CL**ear **x**	CLx
%	Computes x**%** of y	%
Σ+	Summa + (Statistical function)	Σ+
Γ(x)	Gamma (x)	Γ(x)
MNU	**M**e**NU**	Cannot be recorded in program
R/S	**R**un/**S**top	R/S
US <> MTR	**US/Imperial < > Metric** Conversion	Cannot be recorded in program
VRG	**V**iew **R**e**G**isters	Cannot be recorded in program
SCI	Selects **SCI**entific notation display	SCI
FIX	Selects **FIX**ed point display	FIX
ENG	Selects **ENG**ineering notation display	ENG
RTC	**R**eal **T**ime **C**lock	Cannot be recorded in program
DSZ	**D**ecrement and **S**kip if **Z**ero	DSZ
ISZ	**I**ncrement and **S**kip if **Z**ero	ISZ

Key marking	Function	The associated text in the editor
INS	INSerts Line (In program editing mode)	NOP
ST(i)	STore according to (i)	ST i
° ' ''<>D.MS	Degrees. minutes seconds < > Decimal Degrees.MinutesSeconds	>DMS <DMS
PRG STO	STOre PRoGram	Cannot be recorded in program
PRG RCL	ReCaLl PRoGram	Cannot be recorded in program
CF	Clear Flag	CF
CL PRGM	CLear PRoGraM	Cannot be recorded in program
BEEP	Audible signal	BEEP
R<>P	Rectangular < > Polar coordinate conversion	R->P R<-P
CL REG	CLear REGisters (0…9, A…E, I, Flags)	CLRR
CL STK	CLear STacK	CSTK
x<y	Tests to see if the value in the X-register is less than y.	x<y
x>y	Tests to see if the value in the X-register is greater than Y.	x>y
x≠y	Tests to see if the value in the X-register is unequal to the value in the Y.	x≠y
x=y	Tests to see if the value in the X-register is equal to the value in the Y.	x=y
Σ-	Correcting cumulation entries	Σ-
$\bar{x},\bar{y}$	Computes mean (average) of x and y values accumulated by Σ+	mean
P($^{y}{}_{x}$)	Permutation	PERM
LSTx	LaST x	LSTx
CON	CONstants	Cannot be recorded in program
RND	RaNDom Number	RNDM
PAUSE	Stops program execution and transfers control to keyboard for 1 second, then resumes program execution.	PAUS
VST	View STack	Cannot be recorded in program
DEG	Sets decimal DEGrees mode for trigonometric functions.	DEG
	Sets RADians mode for trigonometric functions.	RAD
GRD	Sets GRaDs mode for trigonometric functions.	GRD
g DSZ (i)	Decrements and Skip if Zero according to (i)	DSZi
g ISZ (i)	Increments and Skip if Zero according to (i)	ISZi

Key marking	Function	The associated text in the editor
DEL	**DEL**etes a line (In program editing mode)	Cannot be recorded in program
RTN	**R**e**T**ur**N** (from subroutine)	RTN
RC(i)	**R**e**C**all according to **(i)**	RC i
PRGM	Enters **PR**o**G**ra**M** editing mode	Cannot be recorded in program
ST I	**ST**ores number in **I**-register	ST I
RC I	**R**e**C**alls number from **I**-register	RC I
SF	**S**et **F**lag	SF
R↑	**R**oll-up↑	ROL↑
x<>I	Exchanges contents of displayed X-register with those of I-register	x<>I
x<0	Tests to see if the value in the X-register is less than zero.	x<0
x>0	Tests to see if the value in the X-register is greater than zero.	x>0
x≠0	Tests to see if the value in the X-register is unequal to zero.	x≠0
x=0	Tests to see if the value in the X-register is equal to zero.	x=y
Δ%	Computes percent of change from number in Y-register to number in displayed X-register.	Δ%
STD	Computes sample **ST**andard **D**eviations of x and y values accumulated by Σ+	SDEV
C($^y{}_x$)	Combination	COMB
INT	**INT**eger	INT
FRAC	**FRAC**tion	FRAC
ABS	**ABS**olute Value	ABS
NOP	**NO** o**P**eration	NOP

Function keys pressed from the keyboard execute individual functions as they are pressed. Input numbers and answers are displayed. All function *keys* listed below operate either from the keyboard or as recorded instructions in a program.

Prefix keys

f Pressed before function key, selects **gold** function printed above key

g Pressed before function key, selects **blue** function printed above key

Display control

There are three keys, f SCI FIX ENG that allow you to control the manner in which numbers appear in the display in the graphiCAL. Each key above, followed by a number key changes the number of displayed digits without changing the format. FIX displays numbers in fixed decimal point format while SCI permits you to see numbers in scientific notation format. ENG displays numbers in engineering notation, with exponents of 10 shown in multiples of three (e.g., 10^3, 10^{-6}, 10^{15}). No matter which format or how many displayed digits you choose, these display control keys alter only the *manner* in which a number is displayed in the graphiCAL . The actual number itself is not altered by any of the display control keys. No matter what type of display you select, the graphiCAL always calculates internally with numbers consisting of full 13-digit mantissas multiplied by 10 raised to a two-digit exponent.

Note: f SCI or FIX or ENG followed by CLx key clears the notation format: floating point format is displayed: FLP

Digit entry

1 through 9 Digit keys

CHS CHanges Sign of number or exponent of 10 in displayed X-register

EEX Enter EXponent. After pressing, next numbers keyed in are exponents of 10

. Decimal point

ENTER ⬆ Enters a copy of number in displayed X-register into Y-register. Used to separate numbers.

Storing and Recalling Numbers

STO STOre. Followed by address key, stores displayed number in storage register (R_0 through R_9, R_A through R_E,) specified. Also used to perform storage register arithmetic.

RCL ReCaLl. Followed by address key, recalls number from storage register (R_0 through R_9, R_A through R_E,) specified into the displayed X-register. Also used to perform storage register arithmetic.

f PRG STO Followed by a two-number address, (00 through 63) stores the current program in the external program memory.

f PRG RCL Followed by a two-number address, (00 through 63) recalls a program from the external program memory and replaces it in the PROG_RAM.

Special case: Program #99 is a Test Program. Can be recalled, only. (See details later.)

g ST I STO STore-I. Stores number in I-register.

g RC I RCL ReCall-I. Recalls number from I-register.

g x<>I Exchanges contents of displayed X-register with those of I-register.

Using the I-Register for Indirect Control

You have seen how the value in the I-register can be altered using the **ST I**, **RC I** and **x<>I** operations. But the value contained in the I-register can also be used to *control* other operations. The (i) *(indirect)* function combined with certain other functions allows you to control those functions using the current number in the I-register. (i) uses the number stored in the I-register as an *address.* The indirect operations that can be controlled by the I-register are:

ST(i) when the number in the I-register is 0 through 28, stores the value that is in the display in the storage register addressed by the current number in the I-register.

RC(i) when the number in the I-register is 0 through 28, recalls the contents of the storage register addressed by the current number in the I-register.

Performing Register Arithmetic

Arithmetic operations (+ , -, *, ÷) can be performed between a data storage register and the X-register (display). To modify the contents of the storage register, press STO followed by the applicable operator key (+, - , * , ÷), then the number key specifying the storage register. For example, store 3 in register R_1 then increment it by 2.

3 STO 1

2 STO + 1 $R_1 = 5$

Conversely, to alter the X-Register (displayed value) without affecting the contents of the data storage register or the other stack registers, press RCL, the applicable operator, then the number key specifying the storage register. For example, add the current value stored in R_1 (5.00) to a new entry (2).

2 RCL + 1 7,00 <- displayed X-Register

Function Keys

Number alteration

g **ABS** Gives **ABS**olute value of number in displayed X-register

g **INT** Leaves only **INT**eger portion of number in X-register by truncating fractional portion.

g **FRAC** Leaves only **FRAC**tional portion of number in X-register by truncating integer portion.

Reciprocals

1/x Calculates the reciprocal of a number in the displayed X-register.

Factorials

f **n!** Calculates the factorial of a positive integer in the displayed X-register.

(0 < n < 69)

Square Roots

g **√x** Calculates the square root of a number in the displayed X-register.

Squaring

f **x^2** Calculates the square of a number in the displayed X-register.

Qube raising

f **x^3** Calculates the cube of a number in the displayed X-register.

Using π

g **π** The value **π** accurate to 12 places (3.141592653589) is provided as a fixed constant in the graphiCAL.

Percentages

% The % key is a two-number function which allows you to compute percentages .
The formula is used: (x*y)/100 = %

Percent of Change (Δ%)

g **Δ%** The **Δ%** *(percent of change)* key is a two-number function that gives the percent increase or decrease from Y to X. The formula is used: ((x-y)*100) / y = Δ%

Trigonometric Functions

Your graphiCAL provides you with six trigonometric functions, which operate in decimal degrees, radians, or grads. You can easily convert angles from decimal degrees to radians or vice versa, and you can convert between decimal degrees, and *degrees, minutes, seconds* . You can also add angles specified in *degrees, minutes, seconds* directly, without converting them to decimal.

The six trigonometric functions provided by the calculator are:

f sin sine
g sin^{-1} arc sine
f cos cosine
g cos^{-1} arc cosine
f tan tangent
g tan^{-1} arc tangent

To be continued!

Polar/Rectangular Coordinate Conversion

Two functions are provided for polar/rectangular coordinate conversions. Angle is assumed in decimal degrees, radians, or grads, depending upon the trigonometric mode first selected by g DEG RAD GRD

Rectangular to Polar conversion

f R<>P

t	>	t
z	>	z
y-coordinate	>	angle
x-coordinate	>	magnitude

Polar to Rectangular Conversion

g R<>P

t	>	t
z	>	z
angle	>	y-coordinate
magnitude	>	x-coordinate

Logarithmic and Antilog (Exponential) Functions

The graphiCAL computes both natural and common logarithms as well as their inverse functions (antilogarithms):

[f] [ln] is $\log_e$. (natural log). It takes the **ln** of the value in the X-register to base *e (2.718 ...).*

[g] [e^x] is antilog_e. (natural antilog). It raises ***e*** (2.718 .. .) to the power of the value in X-register.

[f] [log] is $\log_{10}$. (common log). It takes the **log** of the value in the X-register to base 10.

[g] [10^x] is antilog_{10}. (common antilog). It raises **10** to the power of the value in X-register.

Random Number

[f] [RND] The generated "random number" in X-register
$0 < x < 0{,}999\ 999\ 999\ 999$

John von Neumann first suggested the used approach in about 1946. His idea was to take the square of the previous random number and to extract the middle digits. The "middle square" sequence ***isn't*** random, but it ***appears*** to be.

Operations & the Stack

Unary or Monadic Operations:

N!, x^2, x^3, log*x*, ln*x*, 10^x, e^x, √*x*, π, sin, cos, tan, $\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$, INT, FRAC, ABS, Γ(*x*)

```
t  ----------->  T

z  ----------->  Z

y  ----------->  Y

x  --> operation -->  X

x  ----------->  Last x
```

Binary or Dyadic Operations: y+x, y-x, y*x, y/x, $\sqrt[x]{y}$, y^x

Exceptions: R<>P, P(y,x), C(y,x)

```
t  ----------\---------> T
z  -------\   \--------> Z
y -----\   \-----------> Y
x ------operation ----> X

x  ----------->  Last x
```

Special functions

Statistical functions

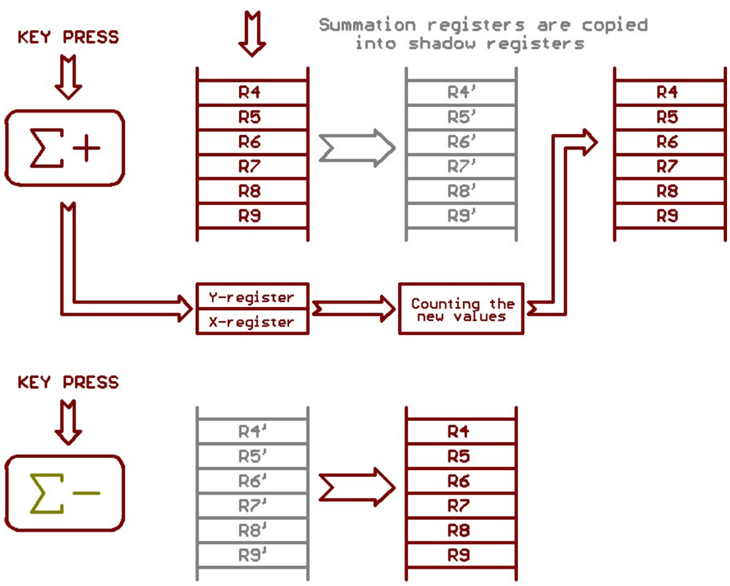
Pressing the Σ+ key automatically gives you several different sums, and products of the values in the X- and Y -registers at once. In order to make these values accessible for sophisticated statistics problems, they are automatically placed by the calculator into storage registers R04 through R09 . *The only time that information is automatically accumulated in the storage registers is when* Σ+ *is used.* Before you begin any calculations using the Σ+ key, you should first clear the storage registers by pressing f CL REG .

Registers used in statistical operations (summation registers)

R04	Σx
R05	Σx^2
R06	Σy
R07	Σy^2
R08	Σxy
R09	n

After pressing the Σ+ key the contents of the summation registers are first saved in so-called shadow-registers. After that, it calculates the new values and loads them into summation registers.

If one of the values is changed, or if you discover after you have pressed the Σ+ key that one of the values is in error, you can correct the summations by using the Σ- *(summation minus)* key. The previous state can be restored by pressing the Σ- key.



Shadow registers are copied back into summation registers.

After you have accumulated these products and sums using the **Σ+** key, they remain in storage registers, where they are used to compute mean and standard deviation using the **x,y** and **STD** functions.

Note: To use *only* the Σx and Σy that you have accumulated in the storage registers, you can press **RCL** followed by **Σ+** . This brings Σx into the displayed X-register and Σy into the Y-register, overwriting the contents of those two stack registers. The stack does not lift. (This feature is particularly useful when performing vector arithmetic.)

To be continued!

Program editing

Let's load a program, what is "burned-in" in program memory and can't be altered. This is a test program. There are five different sub-programs in it:

1. Moon-lander. Start address .000

    Initial settings: Start address .050

    500 STO 0 -> height: 500 feet

    50 CHS STO 1 -> velocity: -50 feet/sec

    120 STO 2 -> fuel: 120

2. Fibonacci sequence. Start address .100 (The shortest program.)
3. Calculator forensics by Mike Sebastian. Start address: .120
4. Benchmark #1 (Math pseudo code) https://www.hpmuseum.org/speed.htm

    Start address: .140

5. Benchmark #2 (Trig pseudo code) https://www.hpmuseum.org/speed.htm

    Start address: .180

Load the program:

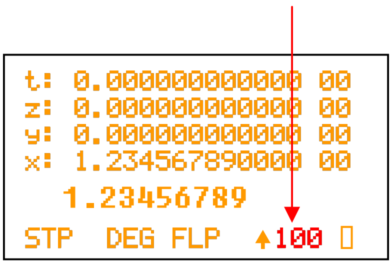
[f] [PRGM / RCL] [9] [9]

Promt: `Test prgm loaded`

Let's see the Fibonacci sequence program. Fibonacci sequence in which each number is the sum of two preceding ones. The sequence commonly starts from 0 and 1, although some authors start the sequence from 1 and 1, or sometimes (as did Fibonacci) from 1 and 2.

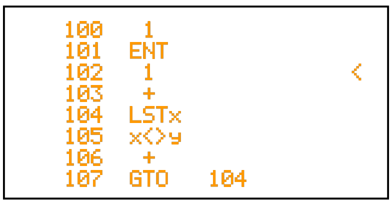
Enter the start address of the the selected program:

[GTO] [.] [1] [0] [0]



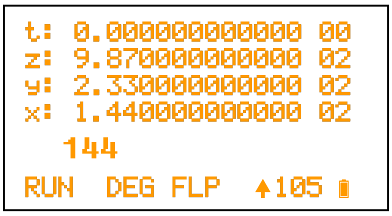
To enter into program editing mode press:

[g] [PRGM] Scroll the thumb-wheel CW to see the "whole" program. (100…107)



Switch back to run mode to run the program:
(Set the starting address again to be safe: GTO . 100)

The program is started with the [R/S] key.



The display shows the 13th and 12th elements of the series.

The first few Fibonacci numbers: 1 1 2 3 5 8 13 21 34 55 89 **144** 233 377 610 987 1597 ..
The ratio of the last element of the series to the one preceding it, approaches the Golden-ratio:
1,618 033 988 …..

To be continued!

In PROGRAM mode only five operations are **active**. These operations are used to record programs, and cannot themselves be recorded in program

[GTO] Followed by [.] [n] [n] [n] positions calculator to step **n n n** of program memory. No instructions are executed.

[BST] Back step. Moves calculator one step back in program memory. No instructions are executed.

[SST] Single step Moves calculator forward one step of program memory. No instructions are executed.

Moving through the program table can also be done by scrolling the thumb-wheel. CCW: moves the table down. CW: moves the table up.

[g] [DEL] Deletes current instruction key from program memory. All subsequent instructions moved up one step. The addresses of the **GTO**, **GTO LBL** and **GSB** instructions are re-counted.

[f] [INS] Inserts a NOP instruction into program memory. All subsequent instructions moved down one step. The last instruction (255.) is lost. The addresses of the **GTO**, **GTO LBL** and **GSB** instructions are re-counted.

There are few **inactive** operations what cannot be recorded:
US <> MTR US (Imperial)-metric conversions, VRG View registers, VST View stack, TIME, CON Constants, RND Random number, MNU Menu

Unconditional Branching

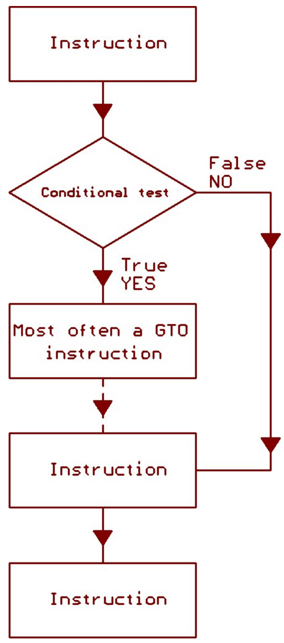
You have seen how the non-loadable operation [GTO] [.] [n] [n] [n] can be used from the keyboard to transfer execution to any step number of program memory. You can also use the *go to* instruction as part of a program, but in order for [GTO] to be *recorded* as an instruction, it must be followed by an address: [GTO] [n] [n] [n] or label designator: [GTO] [LBL] [0] … [E] .
(nnn < 255)

Conditionals and Conditional Branches

Often there are times when you want a program to make a decision.
The *conditional* operations on your **graphiCAL** keyboard are useful as program instructions to allow your calculator to make decisions. The eight conditionals that are available on your **graphiCAL** are:

f x<y x>y x≠y x=y g x<0 x>0 x≠0 x=0

Each conditional essentially asks a question when it is encountered as an instruction in a program. If the answer is YES, program execution continues sequentially downward with the next instruction in program memory. If the answer is NO, the calculator branches *around* the next instruction. For example:



Flag Test Operations

The calculator has eight flags *(called flag 0 .... flag* 7) available for your use. A flag is an invisible piece of information with just two possible conditions: **on** or **off**. The flag operations are given in Figure above. You can **clear** or **set** a flag on or off by using the **Clear** or **Set Flag** operations. These operations can be executed from the keyboard or from a program. The reason for setting a flag is so that a program can later make a decision based on the condition of the flag *(using the test flag operations).*

Command-Cleared flag

There are two types of flags. Flags F0 and F1 are *command-cleared flags* -that is, once they have been set by an g SF n operation, they remain set until they are commanded to change by f CF n operations. Command-cleared flags are generally used to remember program status (e .g., are outputs desired?). (n = 0,1)

Test-Cleared Flags

Flags F2 .... F7 are *test-cleared flags.* They are cleared by a test operation . For example, if you had set flag F2 with an g 2 operation and then it was tested later in a program with an F? 2 instruction, flag F2 would be cleared by the test-execution would continue with the next step of program memory (the " DO if TRUE" rule) ,but the flag would then be cleared and would remain cleared until it was set again . The test-cleared flags are used to save the f CF operation after a test. (However, test-cleared flags *can* be cleared by the f CF operation, if desired.) Besides being a test-cleared flag, flag F3 alone is *set by digit* entry-that is, as soon as you key in a number from the keyboard, flag F3 is set.

Incrementing and Decrementing the I-Register

You have seen how a number can be stored in the I-register and then changed, either by storing another number there, or by using the g x<>I operation . You will find either of these methods useful, whether you are utilizing them as instructions in a program or using them manually from the keyboard. Another way of altering the contents of the I-register, and one that is most useful during a program, is by means of the f ISZ *(increment I, skip if zero)* and f DSZ *(decrement I, skip if zero)* instructions . These instructions either add the number 1 to (increment) or subtract the number 1 from (decrement) the I-register each time they are executed. In a running program, if the number in the I-register has become zero, program execution *skips* the next step after the ISZ or DSZ instruction and continues execution (just like a false conditional instruction).

Indirect Incrementing and Decrementing of Storage Registers

Above, you learned how to increment or decrement the I-register by using the instructions ISZ and DSZ . By using the number in the I-register as an *address,* the instructions g ISZ (i) and g DSZ (i) increment or decrement the contents of the *storage register* addressed by the number in I.

In this case, the value of the number entered in the I-register cannot be greater than 28. Otherwise, the program will stop with an error message: `Not valid reg no`

Subroutines

Often, a program contains a certain series of instructions that are executed several times throughout the program. When the same set of instructions occurs more than once in a program, it can be executed as a subroutine . A subroutine is selected by the GSB *(go to subroutine)* operation, followed by a label address: 0 .... 9 , A .... E .

A GSB instruction transfers execution to the routine specified by the label address, just like a GTO instruction . However, after a GSB instruction has been executed, when the running program then executes a RTN *(return),* execution is transferred back to the next instruction after the GSB . Execution then continues sequentially downward through program memory.

To be continued!

Auxiliary functions

Selecting a constant

f CON

```
   Load constants
A:Astronomical const.
B:Mathematical const.
C:Chemical constants
D:Physical constants
E: Exit
```

A

```
Astronomic. Unit     m
Parsec               m
Light year           m
Speed of light     m/s   <
Solar mass          kg
Solar radius         m
Solar luminos.       W
Earth mass          kg
```

Navigation by rotating the thumb-wheel, selection by pushing the wheel.

Real Time Clock

f RTC

```
    2023-04-14
     08:10:23
      Friday
Push a key to resume
```

Menu

MNU

```
Set date-------->
Set time-------->
Set contrast---->
Run Monitor----->   <
Get ID.--------->
File transfer--->
Bootloader------>
Exit------------>
```

Navigation by rotating the thumb-wheel,
selection by pushing the wheel.

```
     graphiCAL
Serial number: 001
HW version:    2.0
SW version:    2.2

Exit: push a key
```

Get ID.: Unique features of the unit

The Set Date, Set Time, & Set Contrast menu items speak themselves.

Every PIC circuit, I made, has a so called Monitor program. It is used in debugging period.

The detailed description of the Monitor program is: ***graphiCAL Monitor.pdf***

Get ID: Displays Serial number, SW & HW versions.

File transfer: The current program can be transferred to or downloaded from a PC.
The detailed description of the file transfer in Hungarian: ***graphiCAL_XMIT_HU.pdf***

The detailed description of the file transfer in English: ***graphiCAL_XMIT_EN.pdf***

The detailed description of the Bootloader program: ***graphiCAL Bootloader_EN.pdf***

Addendum

Astronomical Constants

Name		Value	Dimension
Astronomical Unit	AU	1,4959.7870.6600 * 10^{11}	m
Parsec	pc	3,0856.7760.0000 * 10^{16}	m
Light Year	LY	9,4607.3047.2000 * 10^{15}	m
Speed of Light	c	2,9979.2000.0000 * 10^{8}	m/s
Mass of Sun	MS	1,9891.0000.0000 * 10^{30}	kg
Radius of Sun	RS	6.9550.8000.0000 * 10^{8}	m
Solar Luminosity		3,8390.0000.0000 * 10^{26}	W
Mass of Earth	ME	5,9742.0000.0000 * 10^{24}	kg
Radius of Earth	RE	6,3781.3600.0000 * 10^{6}	m
Earth Acceleration of Gravity	g	9,8066.5000.0000	m/s2
Escape velocity		1,1200.0000.0000 * 10^{4}	m/s
Earth-Moon Distance		3,8439.9000.0000 * 10^{8}	m
Mass of Moon	MM	7,3500.0000.0000 * 10^{22}	kg
Radius of Moon	RM	1,7400.0000.0000 * 10^{6}	m
Moon Acceleration of Gravity		1,6000.0000.0000	m/s2
Moon Escape Velocity		2,4000.0000.0000 * 10^{3}	m/s

Math Constants

Name		Value
Golden ratio	Φ	1,6180.3398.8749
Inverse golden ratio	φ	6,1803.3988.7498 * 10^{-1}
Silver ratio	δ_S	2,4142.1356.2373
Plastic number (or silver constant)	ρ	1.3247.1795.7244
Euler number	e	2,7182.8182.8459
Euler-Mascheroni constant	γ	5,7721.5664.9015 * 10^{-1}
Archimedes' constant	π	3,1415.9265.3589
2π		6,2831.8530.7179
Buffon's constant 2/ π		0,6366.1977.2367
π^2		9,8696.0440.1089
√ = Geometric Mean	GM	1.7724.5385.0905
Conic constant, e^2		7.3890.5609.8930
√e		1.6487.2127.0700
e^e		1,5154.2622.4147 * 10^{1}
e^π Gelfond's constant		2,3140.6926.3277 * 10^{1}
π^e		2,2459.1577.1836 * 10^{1}

Chemical Constants

Name		Value	Dimension
Atomic mass unit	amu	1,6605.3873 * 10^{-27}	kg
Avogadro's number	N_A	6.022 141 7930 * 10^{23}	1/mol
Bohr radius	a_0	5,2917.7211 * 10^{-11}	m
Boltzman constant	k_B	1,3807.649 * 10^{-23}	$m^2kgs^{-2}K^{-1}$
Faraday constant	F	9,6485.3399.2400 * 10^4	C/mol
Gas constant	R	8,3144.7215.0000	J/mol*K
Molar Planck constant	N_Ahc	3,9903.1268.2157 * 10^{-10}	Js/mol
Planck constant	h	2,9979.2458 * 10^8	m/s
Specific heat capacity of liquid water	*c*	4.18	kJ kg^{-1}°C^{-1}
Stefan-Boltzman constant	σ	5.6703.7441.9000 * 10^{-8}	$Wm^{-2}K^{-4}$
To be continued			
To be continued			
To be continued			
To be continued			
To be continued			
To be continued			

Physical Constants

Name		Value	Dimension
Speed of Light in Vacuum	c	2,9979.2458.0000 * 10^8	m/s
Elementary Charge	e	1,6021.7653.1400 * 10^{-19}	C
Electron Mass	me	9,1093.8261.6000 * 10^{-31}	kg
Neutron Mass	mn	1,6749.2728.2900 * 10^{-31}	kg
Proton mass	pm	1,6726.2171.2900 * 10^{-31}	kg
Vacuum electric permittivity	ε_0	8.854187812813 * 10^{-12}	F/m
Vacuum magnetic permeability	μ_0	1,2566.3706.2121 * 10^{-6}	H/m
Boltzman constant	k_B	1,3807.649 * 10^{-23}	$m^2kgs^{-2}K^{-1}$
Energy	1eV	1,6021.7663.4000 * 10^{-19}	J
Planck constant		6,6260.6931.1000 * 10^{-34}	Js
Newtonian constant of gravitation	G	6,6742.1000.0000 * 10^{-11}	$m^3kg^{-1}s^{-2}$
Rydberg constant	R_∞	1,0973.7315.6854 * 10^7	m^{-1}
Bohr radius	a_0	5,2917.7210.8180 * 10^{-11}	m
Standard acceleration of gravity	g	9,8066.5000.0000	m/s^2
Heat capacity of liquid water	c	4,1800.0000.0000	kg^{-1}*C^{-1}
Platinum thermal coefficient		3,8505.5000.0000 * 10^{-3}	

Conversions (US/Imperial < > Metric)

US unit	Metric unit	Conversion value
mil	mm	2,5400.0000.0000 * 10^{-2}
inch	cm	2,5400.0000.0000
feet	m	0,3048.0000.0000 * 10^{-1}
yrd	m	9,1440.0000.0000 * 10^{-1}
nmile	km	1,6093.4400.0000
smile	km	1,8520.0000.0000
m/h	km/h	1,6093.4400.0000
$inch^2$	cm^2	6,4516.0000.0000
$foot^2$	cm^2	9,2903.0400.0000 * 10^2
$inch^3$	cm^3	1,6387.0640.0000 * 10^1
gallon	L (l)	3,7854.1178.4000
ounce	g	2,8349.5231.0000 * 10^1
pounds	kg	4,5359.2370.0000 * 10^{-1}
psi	bar	6,8947.5729.0000 * 10^{-2}
lbf	N	4,4482.2162.0000
BTU	J	1,0550.5585.0000 * 10^3

Technical specifications:

Size: (L*W*H) [mm]	145*83*34
Weight: [g]	200
Accu: EEMB LP103454 [mAh]	2000
RTC battery	CR1216, CR1220 3V, LiIon coin
Display: 128*64 pixel, passive OLED	RAYSTAR REX012864Q ... orange, sky-blue or white
Power consumption: [mA]	< 80 (full display, program is running)
Power consumption: [mA]	< 25 (single line, stand-by)
Power consumption: [µA]	< 10 (turned-off)
Processor:	PIC18F67K22 RISC
fosc : [MHz]	64 MHz (16 MIPS)
RTC:	RV-3049-C2
Ext. storage:	48L512M (EERAM)
Sound:	35 mm Piezo disc
Connection/charging/file transmit	USB-C 3.0

graphiCAL.pdf

Velence, 2023. 09. 13.