

# Bax58C Compiler Manual



**Bax58C** is a program that compiles **Basic** programs to generate programs for the **TI58C** emulator.



# SUMMARY

Presentation .....	4
The compilation .....	6
Using the program .....	8
Basic language .....	10
• <étiquette>	11
• CALL (statement)	12
• CHAIN (statement)	13
• CLEAR (statement)	14
• CLS (statement)	15
• DIM (statement)	16
• DO (statement)	17
• END (statement)	18
• FOR (statement)	19
• GOSUB (statement)	20
• GOTO (statement)	21
• IF (statement)	22
• INPUT (statement)	23
• LET (statement)	24
• ON (statement)	25
• PRINT (statement)	26
• PRINT TAB (statement)	27
• PRINT USING (statement)	28
• REM (statement)	29
• RETURN (statement)	30
• SET (statement)	31
• SLEEP (statement)	32
• SOUND (statement)	33
• STOP (statement)	34
• SWAP (statement)	35
• ABS(...) (function)	36
• ASC(...) (function)	36
• ATN(...) (function)	36
• CHR(...) (function)	36
• COS(...) (function)	36



• EXP(...)	(function)	36
• FRAC(...)	(function)	36
• INT(...)	(function)	36
• LOG(...)	(function)	36
• RND(...)	(function)	36
• SGN(...)	(function)	36
• SIN(...)	(function)	36
• SQR(...)	(function)	36
• TAN(...)	(function)	36
• INKEY	(variable)	37
• DATE(...)	(variable)	38
• TIME(...)	(variable)	39
• PI	(constant)	40
Languages .....		41
Special settings .....		42
Choosing editors .....		43
About .....		44
Program : Calculation of e .....		45
Program : Fibonacci sequence .....		46



The **TI58C** is a Texas Instruments programmable calculator from years 70 / 80.



It has a special language (SML: Specialized Machine Language)

- 480 program steps or 60 memories (1 step = 2 nibbles\*, 1 memory = 8 bytes), sharing steps/memories is partitionable. (Ex: 240 steps **and** 30 memories)
- RAM = 480 bytes, less than half K byte (about 4K bits!),
- ROM, named "Master Library", of 5000 steps comes standard. (many other modules are optional)
- display red LED 10 digits in 7 segments
- constant memory (for TI58C only, not for TI58 and TI59 !)

a **PC100** printer complete this programmable calculator.

To connect the TI58C to the printer, it is necessary to remove batteries for plug in the pocket calculator on the connector of the printer. Small sensible craftiness: batteries can be put in a trapdoor which assures the load. No ink, the printer "burns" a thermal paper which has the inconvenience to erasing in time... I keep nevertheless my "listings" of programs under forms of paper rolls.



ROM modules of 5000 steps containing about twenty programs or utilities are also marketed:

- ML Master Library
- MU Math Utilities
- LE Leisure Library
- SY Surveying
- EE Electrical Engineering
- ... and many others !

A TV interface has also been marketed by a French electronics company.

This interface has a receptacle identical to the base of connection of the printer and connects to a TV (Black & White) with a coaxial cable of antenna.

The display corresponds exactly to what usually goes out on the printer and a "trace mode" allows to follow step by step the execution of the program for debugging.



The language of the **TI-58/TI-58C/TI-59** calculators is considered a Specialized Machine Language (**LMS**) which uses direct algebraic notation (**AOS [Algebraic Operating System]**).

In 1983, two students from the "**Naval Postgraduate School**" in Monterey (California) wrote a thesis entitled "*Design and implementation of a Basic Cross-Compiler and virtual memory management for the TI-59 programmable calculator*".

This thesis was on the implementation of a compiler (written in Pascal) from **WBASIC** to the **AOS** language of the **TI-59**.

Since 3 programs have been developed :

- The **Claudio Larini's** compiler from Basic to **TI-59** : this compiler, **BAX59**, is very strongly inspired by the thesis of 1983 and takes up the idea of a compilation from Basic to the **AOS** language of the **TI**.  
(<http://www.claudiolarini.altervista.org/bax59.htm>)
- **Guillaume Tello's** Compiler : the starting language is inspired by the improved TI language : variable names, names of labels, flags, comments, LOOP, WHILE, UNTIL, FOR, IF, ELSE, SELECT, CASE...  
([http://gtello.pagesperso-orange.fr/ti58\\_f.htm](http://gtello.pagesperso-orange.fr/ti58_f.htm))
- **T\_Compiler** by **Philippe Tivolle** translates programs written in "T" which is a simple object oriented programming language, modeled on Java. (Language a little too specific!)  
(<http://ti59compiler.wixsite.com/ti59>)



# The compilation

The Basic language being better known and especially more easily accessible, a compilation of programs written in this language to give programs in the language of the **TI-58/TI-58C/TI-59** calculators can be considered as a logical complement to the **TI58C** emulator.

```

BaxEdit - number.bas
File Edit Options ?
100 REM ----- NUMBER -----
110 A:
120 PRINT
130 PRINT "FIND THE NUMBER"
140 LET x = RND()
150 LET x = INT(x * 1023) + 1
160 LET TRY = 0
170 AGAIN:
180 LET TRY = TRY + 1
190 INPUT "Number ?"; number
200 IF number > x THEN HIGH
210 IF number < x THEN LOW
220 PRINT
230 IF TRY = 1 GOTO SUPER
240 PRINT "Great"
250 PRINT "tries";TRY
260 STOP
270 SUPER:
280 PRINT "CHAMPION !"
290 PRINT "try ";TRY
300 STOP
310 LOW:
320 PRINT number
330 PRINT "too low !"
340 GOTO AGAIN
350 HIGH:
360 PRINT number
370 PRINT "too high !"
380 GOTO AGAIN
390 300 END
    
```

BASIC



```

TI Editor - NUMBER.T58
File Edit Utilities Options ?
// ----- NUMBER -----
LBL A
ADV
// FIND THE NUMBER
HIR 00 OP 00 2 1 2 4 3 1 1 6 0 0 OP 01
3 7 2 3 1 7 0 0 3 1 OP 02
4 1 3 0 1 4 1 7 3 5 OP 03 OP 05 HIR 10
( PGM 15 SBR DMS ) STO 03
( ( RCL 03 * 1 0 2 3 ) INT + 1 ) STO 03 0 STO 04
LBL COS
( RCL 04 + 1 ) STO 04
OP 00 3 1 4 1 3 0 1 4 1 7 OP 03
3 5 0 0 7 1 0 0 0 0 OP 04 OP 55
R/S
STO 05 RCL 05 X/T RCL 03
INV GE SIN
RCL 03 X/T RCL 05
INV GE TAN
ADV RCL 04 X/T 1
EQ DEG
// GREAT
HIR 00 OP 00 2 2 3 5 1 7 1 3 3 7 OP 01 OP 05 HIR 10
OP 00 3 7 3 5 2 4 1 7 3 6 OP 04 RCL 04 OP 06
R/S
LBL DEG
// CHAMPION !
HIR 00
OP 00 1 5 2 3 1 3 3 0 3 1 1
    
```

TI58C

CODE

76	11	98	82	00	69	00	02	01	03	01	01	06	00	00	69	01	03		
07	02	03	01	07	00	00	03	01	69	02	04	01	03	00	01	04	01	07	03
05	69	03	69	05	82	10	53	36	15	71	88	54	42	03	43	03	65	01	00
02	03	95	59	85	01	95	42	03	00	42	04	76	39	69	24	69	00	03	01
04	01	03	00	01	04	01	07	69	03	03	05	00	00	07	01	00	00	00	00
69	04	69	55	91	42	05	43	05	32	43	03	22	77	38	43	03	32	43	05
22	77	30	98	43	04	32	01	67	60	69	00	02	02	03	05	01	07	01	03
03	07	69	03	69	55	69	00	03	07	03	05	02	04	01	07	03	06	69	04
43	04	69	06	91	76	60	69	00	01	05	02	03	01	03	03	00	03	03	69
03	02	04	03	02	03	01	00	00	07	03	69	04	69	55	69	00	03	07	03
05	04	05	00	00	69	04	43	04	69	06	91	76	30	43	05	99	82	00	69
00	03	07	03	02	03	02	69	01	02	07	03	02	04	03	00	00	69	02	07
03	00	00	00	00	00	00	00	00	69	03	69	05	82	10	61	39	76	38	43
05	99	82	00	69	00	03	07	03	02	03	02	69	01	02	03	02	04	02	02
02	03	69	02	07	03	00	00	00	00	00	69	03	69	05	82	10	61	39	





Each compilation of Basic program [.bas] produces a TI58C program [.t58] and a compilation trace file [.log] useful for understanding the process and make any necessary corrections to the Basic source program.

```
=====
BAX58C Compiler - 12 / 11 / 2020 09:49:07
=====
C:\ProgramData\TI58C\Basic\Bax58C\Nombree\number.bas
=====
```

```
100 REM ===== NUMBER =====
110 A:
120 PRINT
130 PRINT "  FIND THE NUMBER"
140 LET x = RND()
150 LET x = INT(x * 1023) + 1
160 LET TRY = 0
170 REM --- PLAY ---
180 AGAIN:
190 LET TRY = TRY + 1
200 INPUT "Number ?"; number
210 IF number > x THEN HIGH
220 IF number < x THEN LOW
230 PRINT
240 IF TRY = 1 GOTO SUPER
250 REM --- GREAT ---
260 PRINT "tries";TRY
270 PRINT "Great !"
280 STOP
290 SUPER:
300 REM --- CHAMPION ---
310 PRINT "try ";TRY
320 PRINT "CHAMPION !"
330 STOP
340 REM --- TOO LOW ! ---
350 LOW:
360 PRINT number
370 PRINT " too low !"
380 GOTO AGAIN
390 REM --- TOO HIGH ---
400 HIGH:
410 PRINT number
420 PRINT " too high !"
430 GOTO AGAIN
440 300 END
```

```
=====
number.t58
=====
Steps      : 312
Registers  : 6
=====
```

R E G I S T E R S	
Basic Name	Reg #
Reserved	00
Reserved	01
Reserved	02
x	=> 03
TRY	=> 04
number	=> 05

L A B E L S	
Basic Addr	TI58 LBL
A	=> A
AGAIN	=> COS
HIGH	=> SIN
LOW	=> TAN
SUPER	=> DEG

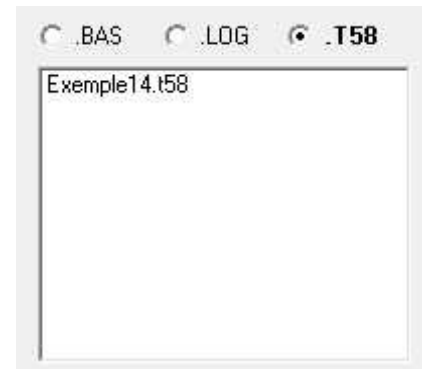
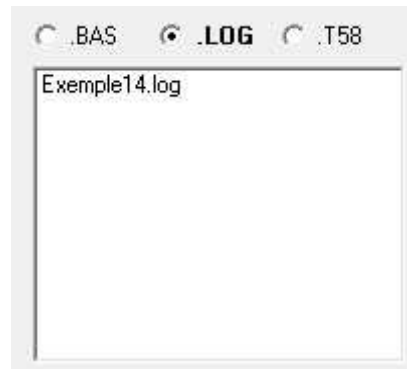
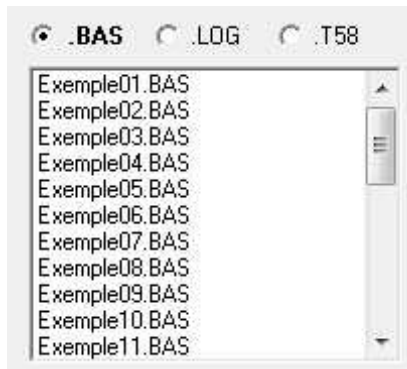
```
----- Basic Line 100
// ===== NUMBER =====
----- Basic Line 110
LBL A
----- Basic Line 120
ADU
----- Basic Line 130
//  FIND THE NUMBER
HIR 00 OP 00
2 1 2 4 OP 01
3 1 1 6 0 0 3 7 2 3 OP 02
1 7 0 0 3 1 4 1 3 0 OP 03
1 4 1 7 3 5 0 0 0 0 OP 04
OP 05 HIR 10
----- Basic Line 140
0 RND STO 03
----- Basic Line 150
RCL 03 * 1 0 2 3 = INT + 1 = STO 03
----- Basic Line 160
0 STO 04
----- Basic Line 170
// --- PLAY ---
----- Basic Line 180
LBL COS
----- Basic Line 190
OP 24
----- Basic Line 200
```

```
OP 00 3 1 4 1 3 0 1 4 1 7 OP 03
3 5 0 0 7 1 0 0 0 0 OP 04
OP 55 R/S
STO 05
----- Basic Line 210
RCL 05 X:T RCL 03 INV GE SIN
----- Basic Line 220
RCL 03 X:T RCL 05 INV GE TAN
----- Basic Line 230
ADU
----- Basic Line 240
RCL 04 X:T 1 EQ DEG
----- Basic Line 250
// --- GREAT ---
----- Basic Line 260
OP 00 3 7 3 5 2 4 1 7 3 6 OP 04 RCL 04 OP 06
----- Basic Line 270
OP 00
2 2 3 5 1 7 1 3 3 7 OP 03
7 3 0 0 0 0 0 0 OP 04
OP 55
----- Basic Line 280
R/S
----- Basic Line 290
LBL DEG
----- Basic Line 300
// --- CHAMPION ---
----- Basic Line 310
OP 00 3 7 3 5 4 5 0 0 OP 04 RCL 04 OP 06
----- Basic Line 320
OP 00
1 5 2 3 1 3 3 0 3 3 OP 03
2 4 3 2 3 1 0 0 7 3 OP 04
OP 55
----- Basic Line 330
R/S
----- Basic Line 340
// --- TOO LOW ! ---
----- Basic Line 350
LBL TAN
----- Basic Line 360
RCL 05 PRT
----- Basic Line 370
// too low !
HIR 00 OP 00
3 7 3 2 3 2 OP 01
2 7 3 2 4 3 0 0 OP 02
7 3 0 0 0 0 0 0 OP 03
OP 05 HIR 10
----- Basic Line 380
GTO COS
----- Basic Line 390
// --- TOO HIGH ---
----- Basic Line 400
LBL SIN
----- Basic Line 410
RCL 05 PRT
----- Basic Line 420
// too high !
HIR 00 OP 00
3 7 3 2 3 2 OP 01
2 3 2 4 2 2 3 OP 02
7 3 0 0 0 0 0 0 OP 03
OP 05 HIR 10
----- Basic Line 430
GTO COS
----- Basic Line 440
=====
```



## Using the program

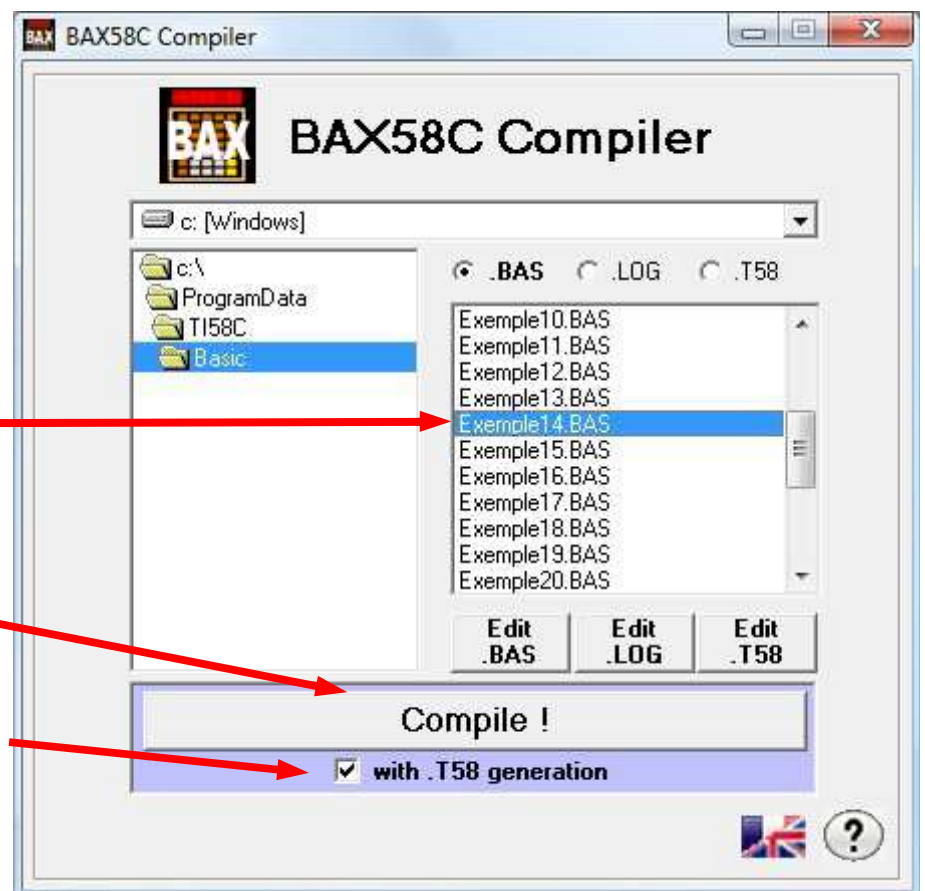
The Basic programs to be compiled have the .BAS extension and their compilation generates a "trace" file of the .LOG extension compilation as well as a program for the TI58C emulator with the .T58 extension.



• Select the program to compile

• Click on **Compile !**

by choosing beforehand whether the **TI58** program must be generated or not (simulation)



At the end of the compilation, a summary is displayed and the .LOG file can be viewed.





At any time, the **.BAS** (Basic program), **.LOG** (compilation "trace") and **.T58** (TI58c program) files can be opened for editing or consultation.

```

10 PRINT "BAX58C Example"
20 FOR I=1 TO 3
30 LET X = 12
40 GOSUB 80
50 PRINT I, X, Q
60 NEXT I
70 STOP
80 LET Q = INT(X/I)
90 RETURN
100 END
    
```

```

=====
BAX58C Compiler - 05 / 09 / 2020 20:55:30
=====
C:\ProgramData\TI58C\Basic\Exemples\Example.bas
=====
10 PRINT "BAX58C Example"
20 FOR I=1 TO 3
30 LET X = 12
40 GOSUB 80
50 PRINT I, X, Q
60 NEXT I
70 STOP
80 LET Q = INT(X/I)
90 RETURN
100 END
=====
          R E G I S T E R S
Basic Name      Reg #
-----
I               => 03
X               => 04
Q               => 05
=====
          L A B E L S
Basic Addr      TI58 LBL
-----
20              => COS
80              => SIN
=====
// BAX58C Example
HIR 00 OP 00
1 4 1 3 4 4 0 6 1 1 OP 01
1 5 0 0 1 7 4 4 1 3 OP 02
3 0 3 3 2 7 1 7 0 0 OP 03
OP 05 HIR 10
1 STO 03 LBL COS
1 2 STO 04
SBR SIN
RCL 03 PRT RCL 04 PRT RCL 05 PRT
1 SUM 03 RCL 03 X/T 3 GE COS
R/S
LBL SIN
( RCL 04 / RCL 03 ) INT STO 05
RTN
R/S
    
```

```

// BAX58C Example
HIR 00 OP 00
1 4 1 3 4 4 0 6 1 1 OP 01
1 5 0 0 1 7 4 4 1 3 OP 02
3 0 3 3 2 7 1 7 0 0 OP 03
OP 05 HIR 10
1 STO 03 LBL COS
1 2 STO 04
SBR SIN
RCL 03 PRT RCL 04 PRT RCL 05 PRT
1 SUM 03 RCL 03 X/T 3 GE COS
R/S
LBL SIN
( RCL 04 / RCL 03 ) INT STO 05
RTN
R/S
    
```



# Basic language

Since the origin of the Basic language, Darmouth College in October 1964, many versions have emerged. Every interactive computer system has had its version, from the most powerful mainframe to "pocket computers" ...

When it comes to PCs, whether under Windows, Linux or other OS, the variants are plethora of. (QuickBasic, QBasic, Bywater Basic, True Basic, PowerBasic, Turbo Basic, wxBasic, Liberty Basic, SmallBasic, FreeBasic...)

The most widespread, Qbasic, QB45 and QB64 versions, inspired the Basic that can be used with the Bax58C compiler. Not all features of Qbasic were implemented and variations had to be introduced to adapt to the target language.

Each line in the Basic program begins with a line number followed by at least one space and then a label or a statement.

A statement may contain :

- text *<text>*,
- variable names *<variable>*,
- numeric or alphanumeric values *<value>*,
- assignment expressions *<expression>*,
- conditional expressions *<condition>*,
- branch addresses: line numbers or labels *<line>* ou *<label>*.



## **<label>:**

label definition.

### **Syntax :**

<label>:

<label> can be any alphanumeric name or a **TI58C** label (A, B, C, D, E, A', B', C', D', E')

### **Example :**

110 A:

.../...

240 Calc:

.../...

380 EndOfPrg:

.../...

### **TI58C equivalent :**

LBL A

.../...

LBL COS

.../...

LBL DEG

.../...

### **Comments:**

All the "Basic" labels are translated into TI labels (LBL COS ...) except the labels corresponding to TI function keys which take the same name (A ... E, A '... E')



# CALL

external subroutine call.

## Syntax :

CALL <label> : { [ <value># ] <value> | <variable> }

## Variant :

LET <variable> =CALL <label> : { [ <value># ] <value> | <variable> }

ou

LET <variable> =CALL( <label> : { [ <value># ] <value> | <variable> } )

<value> # number or code of the library module (01, 02, 03, ... or ML, ST, RE, ...)

<value> program number (suffix of the program name).

<variable> name of the variable containing the program number (suffix of the program name).

## Example :

```
150 CALL A:01
220 CALL B:Var
260 CALL WRI:MU#01
310 CALL A:PH#05
```

## TI58C equivalent :

```
LPG 01 A
LP* 03 B
LIB 10 PGM 01 SBR WRI
LIB 14 PGM 05 A
```



# CHAIN

program chaining and loading.

## Syntax :

```
CHAIN { <value> | <variable> }
```

<value> program number (program name suffix).

<variable> name of the variable containing the program number (suffix of the program name).

## Example :

```
120 CHAIN 01  
140 CHAIN Var
```

## TI58C equivalent :

```
LDP 01  
LD* 03
```



# CLEAR

clear memory.

## Syntax :

CLEAR

## Example :

230 CLEAR

## TI58C equivalent :

CMS





# CLS

clear screen.

**Syntax :**

CLS

**Example :**

230 CLS

**TI58C equivalent :**

CLR



# DIM

definition of variables.

## Syntax :

DIM <variable> [ , <variable> [ , <variable> ] ...]

<variable> can be

- a numeric variable: alphanumeric name
- an alphanumeric variable: alphanumeric name ending in \$
- a flag : alphanumeric name ending with % (10 max)
- an array: alphanumeric name followed by (<value>)

## Example :

```
110 REM === numeric variables
120 DIM Amount, Rate
130 REM === alphanumeric variables
140 DIM NAME$, Message$
150 REM === flags
160 DIM Flag1%, Flag2%, Flag3%
170 REM === arrays
180 DIM MyArray(10)
```

## TI58C equivalent :

none, the DIM instruction allows you to reserve, in precompilation, the registers to be used in the **TI58C** program.

## Comments:

Alphanumeric variable: if declared by DIM (Ex: DIM Var \$) a variable takes 4 registers (20 alphanumeric characters) if not declared a variable takes only one register (5 alphanumeric characters)



# DO...LOOP

conditional loop.

## Syntax :

```
[ DO ] { WHILE | UNTIL } <expression>  
LOOP
```

## Example :

```
25 LET CPT = 0  
30 DO UNTIL CPT = 10  
40 LET CPT = CPT + 1  
50 PRINT CPT  
60 LOOP
```

```
25 LET VAL = 0  
30 WHILE VAL < 5  
40 LET VAL = VAL + 1  
50 PRINT VAL  
60 LOOP
```

## TI58C equivalent :

```
0 STO 03  
LBL COS  
( RCL 03 + 1 ) STO 03  
RCL 03 PRT  
1 0 X:T RCL 03 INV EQ COS
```

```
0 STO 09  
LBL DEG  
( RCL 09 + 1 ) STO 09  
RCL 09 PRT  
5 X:T RCL 09 INV GE DEG
```



# END

end of program.

**Syntax :**

END

**Example :**

230 END

**TI58C equivalent :**

R/S



# FOR...NEXT

loop according to a counter.

## Syntax :

```
FOR <variable> = { <variable> | <value> } TO { <variable> | <value> } [ STEP { <variable> | <value> } ]  
NEXT <variable>
```

## Example :

```
20 FOR I=3 TO 1 STEP -1  
25   FOR Z=1 TO 4  
40   NEXT Z  
45 NEXT I
```

## TI58C equivalent :

```
3 STO 03 LBL COS  
1 STO 04 LBL SIN  
1 SUM 04 RCL 04 X/T 4 GE SIN  
DSZ 03 COS
```



# GOSUB

subroutine call.

## Syntax :

```
GOSUB { <line> | <label> }
```

<line> referring to an existing program line number.

<label> referring to a declared label

## Example :

```
110 GOSUB 240
```

```
.../...
```

```
240 GOSUB EndOfPrg
```

```
.../...
```

```
380 EndOfPrg:
```

```
.../...
```

## TI58C equivalent :

```
SBR COS
```





# GOTO

unconditional connection.

## Syntax :

GOTO { <line> | <label> }

<line> referring to an existing program line number.

<label> referring to a declared label

## Example :

```
110 GOTO 240
```

```
.../...
```

```
240 GOTO EndOfPrg
```

```
.../...
```

```
380 EndOfPrg:
```

```
.../...
```

## TI58C equivalent :

```
GTO COS
```



# IF

conditional branching.

## Syntax :

IF <condition> THEN [ GOTO ] { <line> | <label> }

<condition> could be :

- [ NOT ] <flag> = { 0 | 1 }
- [ NOT ] { <variable> | <expression> | <value> } { > | < | >= | <= | <> | = } { <variable> | <expression> | <value> }

## Example :

```
20 IF A > 10 THEN GOTO 200
30 IF A < 20 THEN 300
40 IF A >= 10 GOTO EndOfProg
50 IF A <= 20 THEN GOTO 300
60 IF A <> 20 THEN 300
70 IF A = 10 GOTO 200
80 IF Flg% THEN GOTO 200
100 IF NOT DRAP% GOTO 300
```

## TI58C equivalent :

```
RCL 03 X:T 1 0 INV GE COS
2 0 X:T RCL 03 INV GE SIN
1 0 X:T RCL 03 GE X2
RCL 03 X:T 2 0 GE SIN
2 0 X:T RCL 03 INV EQ SIN
RCL 03 X:T 1 0 EQ COS
IFF 0 COS
INV IFF 1 SIN
```



# INPUT

data entry.

## Syntax :

INPUT [ { <variable> | <value> } , ] [ <label>: ] <variable>

<label> must be a TI58C label (A, B, C, D, E, A', B', C', D', E')

## Example :

```
30 INPUT VAL
40 INPUT NAME$; N$
130 INPUT "Order ?";ORD
180 INPUT "Choice ?";A:Var
```

## TI58C equivalent :

```
R/S STO 03
OP 00 RCL 04 OP 03 OP 55 R/S STO 05
OP 00 3 2 3 5 1 6 1 7 3 5 OP 03
0 0 7 1 0 0 0 0 0 0 OP 04
OP 55 R/S STO 06
OP 00 1 5 2 3 3 2 2 4 1 5 OP 03
1 7 0 0 7 1 0 0 0 0 OP 04
OP 55 R/S LBL A
STO 07
```



# LET

data assignment.

## Syntax :

LET <destination> = { <expression> | <variable> | <value> } | { TRUE | FALSE }

<destination> could be :

- a numeric variable,
- an alphanumeric variable,
- an alphanumeric variable,
- a flag,
- a cell of an array.

<expression> can consist of variables, values, functions, operators and parentheses:

- functions : EXP(), LOG(), SIN(), COS(), TAN(), ATN(), ABS(), SQR(), INT(), FRAC(), SGN(), RND(), ASC(), CHR()
- operators : + - \* / ^ &

## Example :

```
45 LET A(1) = B(3) + C(5)
50 LET Val0 = EXP(F) + SIN(G)
60 LET Val1 = 0.0000001
70 LET X% = TRUE
80 LET F = 3
100 LET F = F * X
120 LET G = G + ( 1 / F )
130 LET D = G - E
170 DIM A$
180 LET A$ = CHR(13) & CHR(14) & CHR(15) & CHR(16) & "ZZZ" & CHR(17) & CHR(33) & "YYY"
```

## TI58C equivalent :

```
RCL 15 + RCL 22 = STO 03
RCL 24 INV LN X + RCL 25 SIN = STO 23
0 . 0 0 0 0 0 0 1 STO 26
STF 00
3 STO 24
RCL 24 * RCL 27 = STO 24
RCL 25 + ( 1 / RCL 24 ) = STO 25
RCL 25 - RCL 29 = STO 28

1 3 1 4 1 5 1 6 4 6 STO 03
4 6 4 6 1 7 3 3 4 5 STO 04
4 5 4 5 0 0 0 0 0 0 STO 05
0 STO 06
```



# ON...GOTO

conditional branching.

## Syntax :

ON <variable> GOTO { <line> | <label> } [ { <line> | <label> } [, { <line> | <label> } ] ... ]

## Example :

```
30 ON VALUE GOTO 100, 200, 300
90 ON COUNTER GOTO Lab1, Lab2, Lab3, Lab4
```

## TI58C equivalent :

```
RCL 03 1 EQ COS 2 EQ SIN 3 EQ TAN
RCL 07 1 EQ DEG 2 EQ GRD 3 EQ RAD 4 EQ X2
```



# PRINT

printing.

## Syntax :

```
PRINT [ { <variable> | <value> } [, { <variable> | <value> } ] ]
```

## Example :

```
200 PRINT VAL  
210 PRINT "N ";RANG  
220 PRINT "F(N)";LONG  
240 PRINT  
260 PRINT "-----"
```

## TI58C equivalent :

```
RCL 04 PRT  
OP 00 3 1 0 0 OP 04 RCL 03 OP 06  
OP 00 2 1 5 5 3 1 5 6 OP 04 RCL 05 OP 06  
ADV  
// -----  
HIR 00 OP 00  
2 0 2 0 2 0 2 0 2 0 OP 01  
OP 02  
OP 03  
OP 04  
OP 05 HIR 10
```





# PRINT TAB

alphanumeric printing with spacing.

## Syntax :

```
PRINT TAB( {<variable> | <value>} ; { <variable> | <value> | "*" | "." } ]
```

## Example :

```
200 PRINT TAB(8);"*"  
220 PRINT TAB(42);"."  
230 PRINT TAB(8);"XYZ"
```

## TI58C Equivalent :

```
8 OP 07  
4 2 OP 77  
HIR 00 OP 00  
4 4 4 5 OP 02  
4 6 0 0 0 0 0 0 0 0 OP 03  
OP 05 HIR 10
```



# PRINT USING

printing with mask.

## Syntax :

```
PRINT USING {<mask> | <address>} [, { <variable> | <value> } ]
```

## Example :

```
200 PRINT USING ".#####^^",N
```

```
210 PRINT USING 260, PI
```

```
260 : USING "#.####"
```

## TI58C Equivalent :

```
ENG FIX 5
```

```
RCL 04 PRT
```

```
FIX 4
```

```
PI PRT
```



# REM

introduction of a comment in the program.

## Syntax :

```
REM [ <text> ]
```

## Example :

```
110 REM ***** This is a comment *****  
120 REM
```

## TI58C equivalent :

```
// ***** This is a comment *****  
NOP
```

## Comments:

In the case of several successive comments, only the last one will be kept by the TI58C emulator.

REM instructions not followed by comment text will be translated into NOP.



# RETURN

subroutine return.

## Syntax :

RETURN

## Example :

```
110 GOSUB 240
.../...
240 REM ===== SUBROUTINE
.../...
280 RETURN
```

## TI58C equivalent :

RTN



# SET

settings.

## Syntax :

SET <parameter> [ , <parameter> [ , <parameter> ] ]

<parameter> could be :

- ENG, NOT ENG, ENG=TRUE, ENG=FALSE, ENG TRUE, ENG FALSE...
- DEG, RAD, GRD
- FIX={ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | TRUE | FALSE }...

## Example :

```
230 SET ENG FALSE
240 SET FIX=4, DEG
260 SET NOT ENG = FALSE
```

## TI58C Equivalent :

```
INV ENG
FIX 4 DEG
ENG
```



# SLEEP

pause.

## Syntax :

SLEEP [ <value> ]

## Example :

230 SLEEP  
250 SLEEP 3

## TI58C equivalent :

PAU  
PAU PAU PAU



# SOUND

play a musical note.

## Syntax :

SOUND <value1>, {<value2> | <variable2>}

<value1> note of the range (C1, C1 #, D, ..., F3 #, G3, G3 #) or frequency in Hertz (according to the following table)

1	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
	la	la#	si	do	do#	ré	ré#	mi	fa	fa#	sol	sol#
	110	116.54	123.47	130.81	138.59	146.83	155.56	164.81	174.61	185	196	207.65
2	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
	la	la#	si	do	do#	ré	ré#	mi	fa	fa#	sol	sol#
	220	233.08	246.94	261.63	277.18	293.66	311.13	329.63	349.23	369.99	392	415.3
3	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
	la	la#	si	do	do#	ré	ré#	mi	fa	fa#	sol	sol#
	440	466.16	493.88	523.25	554.37	587.33	622.25	659.26	698.46	739.99	783.99	830.61

{<value2> | <variable2>} delay (from 1 to 10) after the note or name of the variable containing the value of the delay after the note.

## Example :

130 SOUND 440, 2.5

140 SOUND "C2#", Tempo

## TI58C equivalent :

2 . 5 X/T 1 3 0 4 SND

RCL 03 X/T 1 5 0 3 6 7 SND



# STOP

program stop.

**Syntax :**

STOP

**Example :**

130 STOP

**TI58C equivalent :**

R/S





# SWAP

data exchange.

## Syntax :

SWAP <variable> , <variable>

## Example :

```
40 SWAP VAL1,VAL2
50 SWAP A(IND),B(IND)
```

## TI58C equivalent :

```
RCL 16 EXC 15 STO 16
```

```
// ===== A(n) =====
LBL COS ( CE + 2 ) STO 00 RTN
// ===== B(n) =====
LBL SIN ( CE + 7 ) STO 00 RTN
```

```
( RCL 17 SBR COS RCL 00 STO 01 )
( RCL 17 SBR SIN RCL 00 STO 02 )
RC* 02 EX* 01 ST* 02
```



# Functions

Functions can be used in assignment expressions (LET)

Basic		TI58C
<b>ABS(...)</b>	absolute value of a number.	IXI
<b>ASC(...)</b>	returns the code PC100	x x
<b>ATN(...)</b>	arctangent of a number	INV TAN
<b>CHR(...)</b>	returns the code PC100	x x
<b>COS(...)</b>	cosine of an angle	COS
<b>EXP(...)</b>	e (base of natural logarithms) raised to a power	INV LNX
<b>FRAC(...)</b>	decimal part of a number.	INV INT
<b>INT(...)</b>	integer part of a number.	INT
<b>LOG(...)</b>	natural logarithm of a number	LNX
<b>RND(...)</b>	return a random number. ([ <mini> [ , <maxi> ] ])	<mini> X/T <maxi> RND
<b>SGN(...)</b>	change the sign of a number	+/-
<b>SIN(...)</b>	sine of an angle.	SIN
<b>SQR(...)</b>	square root of a number	SQR
<b>TAN(...)</b>	tangent of an angle.	TAN



## INKEY (*variable*)

waiting for a character typing.

### Syntax :

<variable> = { INKEY | INKEY\$ }

### Example :

30 mychoice = INKEY

### TI58C equivalent :

KEY STO 03



# DATE (*variable*)

today's date.

## Syntax :

```
{ DATE | DATE$ } ( [ "<value>" ] )
```

<value> can be :

yyyyy			
yyyymm	yyyy.mm	yyyy/mm	yyyy-mm
yyyymmdd	yyyymm.dd	yyyymm/dd	yyyymm-dd
yyy.mmdd	yyy/mmdd	yyy-mmdd	
mmddyyy	mmdd.yyyy	mmdd/yyyy	mmdd-yyyy
ddmmyyy	ddmm.yyyy	ddmm/yyyy	ddmm-yyyy

## Example :

```
230 LET A = DATE()  
250 PRINT DATE$("yyyy-mm")
```

## TI58C equivalent :

```
1 3 NOW STO 05  
1 2 NOW PRT
```



## TIME (variable)

current time.

### Syntax :

```
{ TIME | TIME$ } ( [ "<value>" ] )
```

<value> can be :

hh	mm	ss	
hhmm	hh.mm	hh:mm	hh-mm
hhmmss	hhmm.ss	hhmm:ss	hhmm-ss
hh.mmss	hh:mmss	hh-mmss	

### Example :

```
230 LET A = TIME()  
250 PRINT TIME$("hh.mmss")
```

### TI58C equivalent :

```
2 5 NOW STO 05  
2 7 NOW PRT
```



## PI (*constant*)

PI.

### Syntax :

PI

Value : 3.141592653589

### Example :

```
230 LET A = PI
250 PRINT PI
```

### TI58C equivalent :

```
PI STO 05
PI PRT
```



# Languages

In the **Bax58c** program it is possible to change the language.

As standard 2 files are provided: Bax58cFR.lan (French) and Bax58cEN.lan (English) but you can create your own Bax58cXX.lan language file by translating one of the existing files.

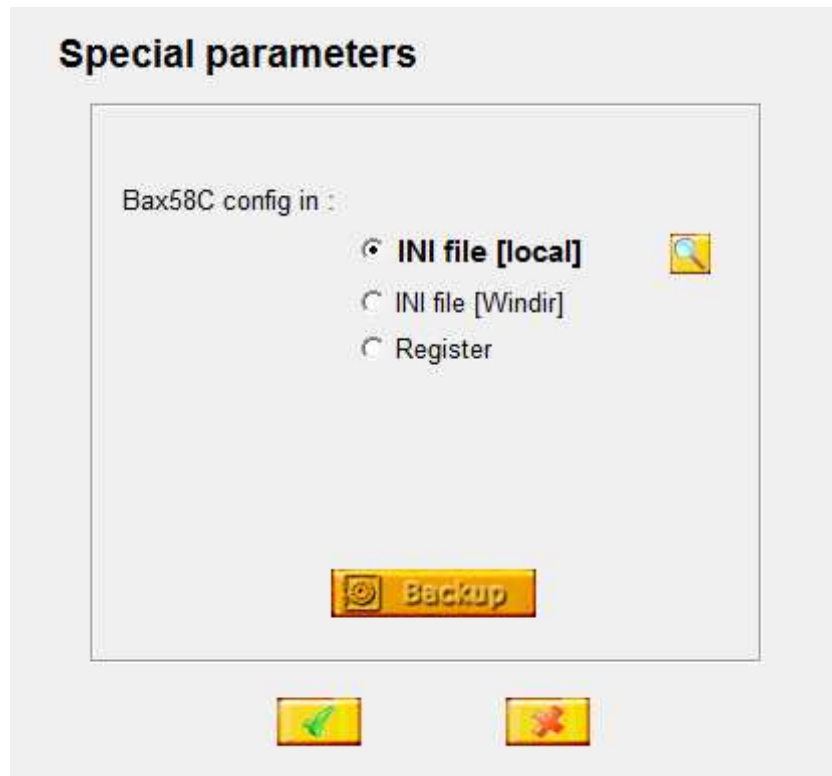


## Special settings SHIFT+ALT+I

The **BAX58C** Compiler configuration parameters can be stored

- either in a local **Bax58C.ini** file (application directory),
- either in a **Bax58C.ini** file in the Windows directory (systemroot),
- either in the Windows registry.

- ⇒ Choosing **local INI file** deletes the **Bax58C.ini** file from the Windows folder and creates the local file **Bax58C.ini**,
- ⇒ Choosing **Windows INI file** deletes the **Bax58C.ini** file from the application folder and creates the Windows file **Bax58C.ini**,
- ⇒ Choosing **Register** deletes the file **Bax58C.ini** (local or Windows) and creates the settings in the registry.



Click on the "Backup" button to save the parameters in

- a **Bax58Cyyyyymmddhhmmss.ini** file for general parameters

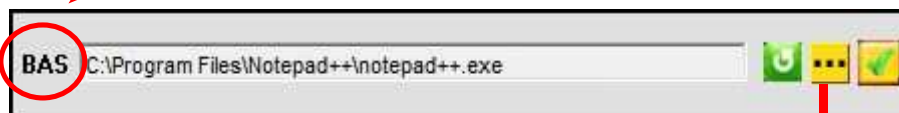
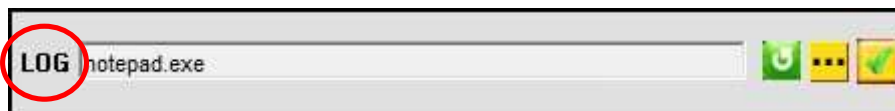
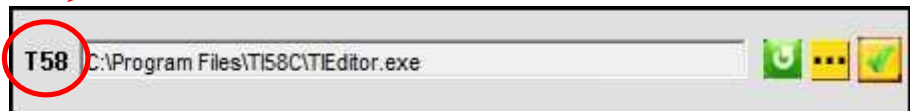




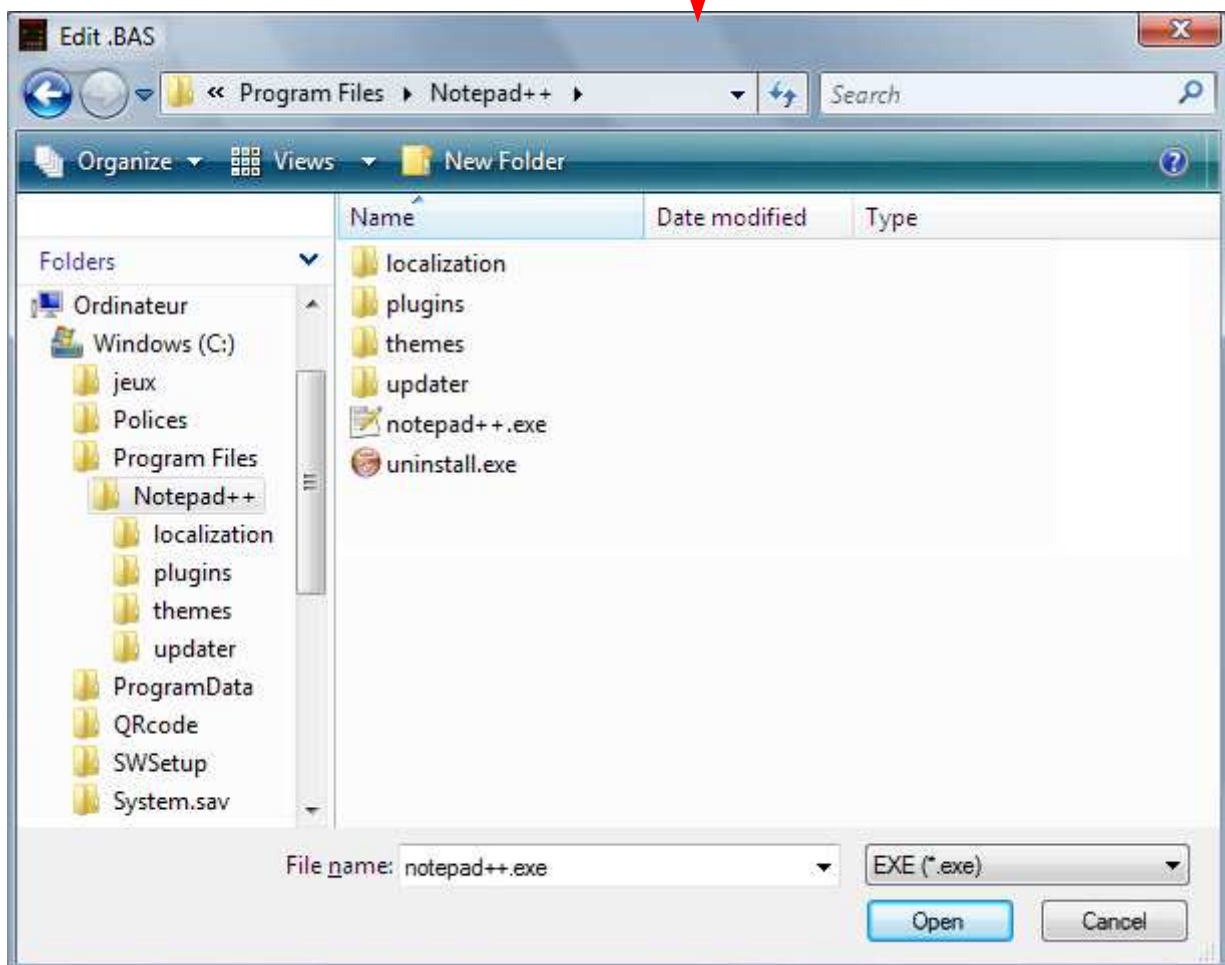
## Choosing editors

Files handled by the **Bax58C** compiler can be edited with external programs such as **Notepad**.

For each type (**.BAS**, **.LOG**, **.T58**) the editor used can be configured by clicking, with the right mouse button, on the "Edit" button of each type



Choose the program then confirm



## About

displays the version number.



# Program : Calculation of e

Program eEN.bas / eEN.t58

[ E ] initialization and first calculation [ A ] next calculation

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots + \frac{1}{n!} + \dots$$

```

=====
BAX58C Compiler - 30 / 09 / 2020 10:16:01
=====
C:\ProgramData\TI58C\Basic\eeEN.bas
=====
100 REM ===== CALCULATION OF E =====
110 E:
120 GOSUB DASHES
130 PRINT " CALCULATION OF E "
140 GOSUB DASHES
150 LET A = 1
160 AGAIN:
170 GOSUB FACTN
180 PRINT A, E
190 GOSUB DASHES
200 STOP
210 A:
220 LET A = A + 1
230 GOTO AGAIN
240 REM ===== CALCULATION =====
250 FACTN:
260 LET E = 1
270 FOR I=1 TO A
280 LET FACT = 1
290 FOR J=1 TO I
300 LET FACT = FACT * J
310 NEXT J
320 LET E = E + 1 / FACT
330 NEXT I
340 RETURN
350 REM ===== LINE OF EQUALS =====
360 DASHES:
370 PRINT "=====
380 RETURN
390 END
=====

                eEN.t58
=====
                Steps      : 169
                Registers   : 8
=====

    R E G I S T E R S
    Basic Name      Reg #
    -----
    Reserved        00
    Reserved        01
    Reserved        02
    A                => 03
    E                => 04
    I                => 05
    FACT            => 06
    J                => 07
=====

    L A B E L S
    Basic Addr      TI58 LBL
    -----
    E                => E
=====

```

```

                A          => A
                DASHES    => COS
                AGAIN     => SIN
                FACTN     => TAN
                270       => DEG
                290       => GRD
=====

// ===== CALCULATION OF E =====
LBL E
SBR COS
// CALCULATION OF E
HIR 00 OP 00
1 5 1 3 2 7 OP 01
1 5 4 1 2 7 1 3 3 7 OP 02
2 4 3 2 3 1 0 0 3 2 OP 03
2 1 0 0 1 7 0 0 0 0 OP 04
OP 05 HIR 10

SBR COS
1 STO 03
LBL SIN
SBR TAN
RCL 03 PRT RCL 04 PRT
SBR COS
R/S
LBL A
OP 23
GTO SIN
// ===== CALCULATION =====
LBL TAN
1 STO 04
1 STO 05 LBL DEG
1 STO 06
1 STO 07 LBL GRD
( RCL 06 * RCL 07 ) STO 06
1 SUM 07 RCL 07 X/T RCL 05 GE GRD
( RCL 04 + 1 / RCL 06 ) STO 04
1 SUM 05 RCL 05 X/T RCL 03 GE DEG
RTN
// ===== LINE OF EQUALS =====
LBL COS
// =====
HIR 00 OP 00
6 4 6 4 6 4 6 4 6 4 OP 01
OP 02
OP 03
OP 04
OP 05 HIR 10

RTN
R/S
=====

```



# Program : Fibonacci sequence

Program fiboEN.bas / fiboEN.t58

[ A ] initialization, enter rank [ R/S ] → calculation

```

=====
BAX58C Compiler - 30 / 09 / 2020 10:17:31
=====
C:\ProgramData\TI58C\Basic\Fibonacci\fiboEN.bas
=====
110 REM ===== FIBONACCI =====
120 A:
130 INPUT "Rank ?";RANK
140 IF RANK > 49 GOTO A
150 LET WIDTH = 0
160 LET LENGTH = 1
170 FOR I=1 TO RANK-1
180 LET RESERU = LENGTH + WIDTH
190 LET WIDTH = LENGTH
200 LET LENGTH = RESERU
210 NEXT I
220 GOSUB DASHES
230 PRINT "N ";RANK
240 PRINT "F(N)";LENGTH
250 GOSUB DASHES
260 PRINT
270 STOP
280 B:
290 LET RANK = RANK + 1
300 GOTO 140
310 DASHES:
320 PRINT "-----"
330 RETURN
340 END
=====

                fiboEN.t58
=====
                Steps      :   155
                Registers   :     8
=====

                R E G I S T E R S
                Basic Name   Reg #
                -----
                Reserved     00
                Reserved     01
                Reserved     02
                RANK          => 03
                WIDTH        => 04
                LENGTH       => 05
                I             => 06
                RESERU       => 07
=====

```

```

                L A B E L S
                Basic Addr   TI58 LBL
                -----
                A            =>   A
                B            =>   B
                140         =>  COS
                170         =>  SIN
                DASHES      =>  TAN
                -----

// ===== FIBONACCI =====
LBL A
OP 00 3 5 1 3 3 1 2 6 0 0 OP 03
7 1 0 0 0 0 0 0 0 0 OP 04
OP 55 R/S
STO 03
LBL COS
RCL 03 X:T 4 9 INU GE A
0 STO 04
1 STO 05
1 STO 06 LBL SIN
( RCL 05 + RCL 04 ) STO 07
RCL 05 STO 04
RCL 07 STO 05
1 SUM 06 RCL 06 X/T ( RCL 03 - 1 ) GE SIN
SBR TAN
OP 00 3 1 0 0 OP 04 RCL 03 OP 06
OP 00 2 1 5 5 3 1 5 6 OP 04 RCL 05 OP 06
SBR TAN
ADU
R/S
LBL B
OP 23
GTO COS
LBL TAN
// -----
HIR 00 OP 00
2 0 2 0 2 0 2 0 2 0 OP 01
OP 02
OP 03
OP 04
OP 05 HIR 10

RTN
R/S
=====

```



**Warning to the readers**

The informations contained in this manual are given as an indicative guide and have no exhaustive character even certain.

As an example not restrictive, this manual can propose you one or several addresses of Web sites which will be not more current or which the contents will have changed when you will access it.

So, this information should not engage the responsibility of the author of this manuel.

The author cannot be considered responsible for any omission, error or gap which would have been present into this manual as well as consequences, whoever they are, who would result from information and indications supplied as well as with their use.

Products named in this manual are protected, and trademarks are registered by their holders of respective rights.

This manual is neither published, nor produced by the owners of the calculators which are concerned and the marks are used only as name of products.

