BreakPoint for TI89 / T92 Plus / V200

V1.00

TABLE OF CONTENTS

BreakPoint for TI89 / T92 Plus / V200

PRESENTATION

BreakPoint is a utility program for debugging the BASIC programs. It has been wrote in BASIC and validated on Ti-89 Titanium.

BreakPoint imitates the breakpoint feature we can find in a debugger. It allows you to suspend the execution of the program being debugged, to inspect its variables and eventually to modify them; for this it just takes to insert the command ***brk\pt*** in the program being debugged in the places to be analyzed.
Usually, to debug a program one uses the instructions *DISP* and *PAUSE* to display the value of the variable to be checked; this method has the disadvantage to cause the crash of the program if the variable does not exist for example.

Advantages of BreakPoint

If the content of the variable cannot be converted in text, for instance variable of type *DATA*, *PRGM* or *GDB*, **BreakPoint** provides a minimum of information on the variable and the execution of the program can continue.

The insertion point of the command ***brk\pt*** is named what makes it possible to know in which part of the program takes place the stop.

The name of the shown variable is indicated at the same time as its contents.

During the run of **BreakPoint** it is possible to show other variables than those foresaw at the insertion of the command ***brk\pt***, without needing to stop and to modify the program to be debugged.

INSTALLATION

1- Requirements

BreakPoint runs only under OS in english version.
TI-89 with AMS 2.01 minimum
TI-89 Titanium
TI-92 Plus with AMS 2.03 minimum
Voyage™ 200

2- Size

15KB

3- Transferring file

Send the file *brkpt* to your calculator (.89p for TI-89, .9xp for TI-92 Plus or .v2p for V200).
The Program *pt*, coming from the file *brkpt*, must be put exclusively in the folder *BRK*.

USE

1- Example

Look at the use of **BreakPoint** through a simple example.
The following program displays an airplane and moves it from left to right. With each step of movement of the airplane the execution of **BreakPoint** makes it possible to inspect various variables of the program. After some steps, at the time of the run of **BreakPoint**, we will attribute the value 150 to the variable *horz* to shorten the run of the program.

```
plane()
```

BreakPoint for TI89 / T92 Plus / V200

```
Prgm
setFold(example)
DelVar horz,vert,lasth,lastv,lft,rght
brk\pt("","horz,vert,lasth,lastv,lft,rght") <-----  Choice of variables to be inspected
ClrIO
10→lft
150→rght
10→lasth
30→lastv
30+iPart(10*sin(.045*horz))→vert

For horz,lft,rght,2
 brk\pt("loop","") <---------------------------------------------  Inspection of variables
 Output lastv,lasth," "
 Output vert,horz,char(26)
 horz→lasth
 vert→lastv
EndFor

DispHome
EndPrgm
```

This example program is provided in the file *plane*. Load the file according to your calculator and run *example\plane()*.
During the run of the example program, in every passage by the command ***brk\pt("loop","")***, the program **BreakPoint** is launched:

The list of visualisables variables is displayed.
Select the 7th line

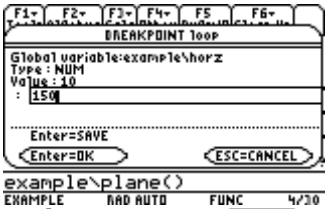
and press ENTER

The selected variable is displayed:

Press ESC to return to the list of variables.

Exit of **BreakPoint** by typing ESC to allow the program *plane* to execute several cycles.

To end the run of *plane*, modify the variable *horz*:
Select the variable *horz* and then type 150 and ENTER under the line *Value*.



and press again ENTER and then ESC.

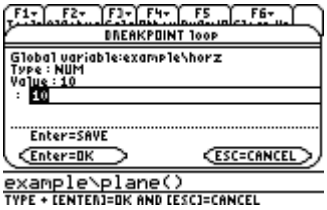
2- Insertion of the command brk\pt in a program

Place the command ***brk\pt(point,expressions)*** in the program to be debugged at the places to be inspected.
The arguments ***point*** and ***expressions*** are explained below.

a) How to name the insertion point of the command brk\pt

The 1st argument of the command ***brk\pt*** is an information that makes it possible to be located during the run of the program to be degugged.
This argument will be displayed in the title of dialog boxes and in the 1st line of the popups.

Example :
```
brk\pt("loop","horz,vert,lasth,lastv,lft,rght")
```



Notice : an empty string in the place of the name of the insertion point of the commande ***brk\pt*** orders the deletion of the permanent list of the global variables previously inspected.

b) <u>How to select global variables to be inspected</u>

The list of the global variables to be inspected is passed in 2nd argument of the command ***brk\pt***. Variables are separated by commas, the whole is put within quotation marks.
If the folder of the variable is not indicated, the variable is considered as belonging to the current folder at the time of the call of **BreakPoint**.

Example:
```
brk\pt("loop ","horz,vert,math\p")
```

If at the time of the call of **BreakPoint** the current folder is *EXAMPLE*, the variables *horz* and *vert* will be made explicit as *example\horz* and *example\vert*.

At every execution of **BreakPoint** the names of the global variables to be inspected are added to a permanent list and are kept until the next run. The variables inspected earlier are thus proposed during the execution of the commands ***brk\pt*** which follow.

For example if at every run of **BreakPoint** we wish to inspect only the global variables *horz*, *vert*, *lasth*, *lastv*, *lft* and *rght*, the list of variables will only be passed as argument of the 1st command ***brk\pt***:

```
...
brk\pt("Point #1","horz,vert,lasth,lastv,lft,rght")
...
brk\pt("Point #2","")  <---------------------------------------------- The variables horz, vert,
...                                                                    lasth,lastv, lft and rght
brk\pt("Point #3","")  <---------------------------------------------- will be proposed again
...                                                                    during the following
brk\pt("Point #4","")  <---------------------------------------------- executions of BreakPoint.
...
```

Variables passed as argument of the command ***brk\pt*** in the course of execution will be marked with a check in the popup window of variables selection.

c) <u>How to indicate expressions not containing local variables</u>

The expressions to be inspected, provided that they do not contain local variables, are passed in 2nd argument of the command ***brk\pt*** in this way:
The expressions are placed in a matrix in which each row contains the expression within quotation marks in the 1st column, followed by a 0 in the 2nd column.
Remark: an expression can be limited to a global variable.

Example:

```
brk\pt("wave1",[["3+sin(θ)",0]["θ",0]])
```

If the expression is a function, 2 solutions:

- `brk\pt("function",[["f",Ø]])`
  During its run **BreakPoint** will display the function and propose arbitrarily the arguments a,b,c...
- `brk\pt("function",[["f(x,y,z)",Ø]])`
  During its run **BreakPoint** will display the function by using the arguments x, y and z.

d) How to indicate local variables or expressions containing local variables

BreakPoint cannot access the local variables of the program to be debugged also it is necessary to pass the value of the variable or the expression in the form of a string in 2nd argument of the command ***brk\pt***.
The variables and expressions are placed in a matrix in which each row contains the name of the variable or the expression within quotes in the 1st column, followed by its value as a string in the 2nd column.

Example:
```
Local ω,t
…
brk\pt("wave2",[["3+sin(ω*t)",string(3+sin(ω*t))]["t",string(t)]])
```

Notice: the variables and expressions must simplify to strings, thus it is not valid with a variable of type DATA for example.

e) How to delete the list of the variables previously inspected

At every execution of **BreakPoint** the names of the global variables to be inspected are added to a permanent list and are kept until the next run.
To reset this list, put an empty string in 1st argument of the command ***brk\pt*** and eventually indicate the new list of global variables in 2nd argument.

Examples:
`brk\pt("","")` : Clears the permanent list of the names of global variables.
`brk\pt("","i,j,k")` : Resets the permanent list of the names of global variables with only i, j and k for inspection at the next run of **BreakPoint**.

Notice: This command does not induce the inspection of the variables during the run.

f) How to interrupt the program to be debugged immediately after the execution of ***BreakPoint***

`brk\pt(…,…):PAUSE`
The instruction PAUSE allows to edit the program being debugged at the current line, after a break by pressing the ON key.

3- BreakPoint flow of execution

a) Selection of the expression to be visualized

The expressions (an expression can be a variable) indicated in argument of the command ***brk\pt*** as well as all the global variables of the previous commands ***brk\pt*** are displayed. The expressions indicated in the current command ***brk\pt*** are marked with a check.
The name of the point of insertion of the commande ***brk\pt*** is indicated in the first line of the list.

Example:

commands ***brk\pt*** in the program being debugged:

```
brk\pt("","lasth,lastv,lft")
…
brk\pt("loop","horz,vert")
```

List of expressions to be visualized:

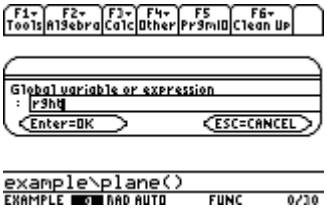
b) <u>Addition of an expression to be visualized</u>

Other expressions can be added in the list of selection by going on the last line "New expression".

Example:

Select the last line:

Type the expression to be added:



The expression is now in the list:

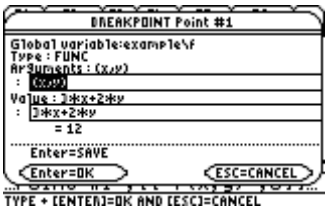
When a single expression is indicated in argument of the command ***brk\pt***, the selector is not displayed and **BreakPoint** passes directly to the visualizing of the expression.

Pressing the key [ESC] when the list of the expressions is displayed allows to exit **BreakPoint** and to resume the execution of the program being debugged.

c) <u>Use of the dialog box of edition in the case of a global variable</u>

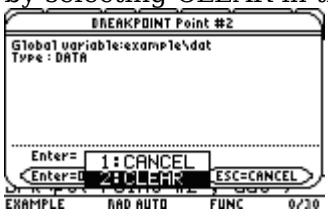
The name of the point of insertion of the command ***brk\pt*** is indicated in the title of the dialog box.

Various informations on the variable are displayed:



- In the first line it is indicated if it is a global variable or a system variable and the name of the variable.
- In the second line it is indicated the variable type.
- If the variable is a function its arguments are displayed. They can be modified in the following line; they must be put in brakets. In the previous example, the variable is the function f(x,y).
  If the arguments of the function were not indicated during the insertion of the command ***brk\pt*** in the program to be debugged, **BreakPoint** names them arbitrarily a, b, c, d...
  By eliminating the arguments and the brackets, the variable ceases to be a function.
  In the case of an empty variable, the entry line of the arguments gives the possibility of defining a new function.
- If it is visualisable, the value of the variable is then displayed (evaluation of first level). It can be modified in the following line.

On the following line the complete evaluation is displayed if it differs from the evaluation of first level. If the complete evaluation is not realizable an error is displayed in the place. Character strings are entered and displayed exactly as they appear in the entry line of the HOME screen.
To erase the variable, clear the entry line of the value.
If the value of the variable is not displayed, the clearing of the variable is however possible by selecting CLEAR in the selector at the bottom of the window then by typing [ENTER]:



The attempt of modification or deletion of a locked or archived variable results in an error message.

d) Use of the dialog box of visualization of an expression

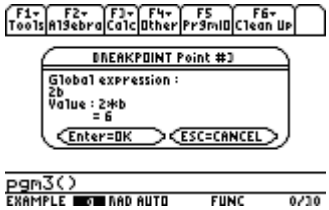
Examples:

1- Global expression

In the program to be debugged:

```
3→b
…
brk\pt("Point #3",[["2b",Ø]])
```

Flow of execution of the **breakpoint**:

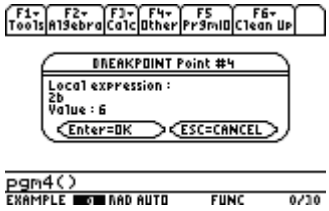


2- Local expression

In the program to be debugged:

```
Local b
…
3→b
…
brk\pt("Point #4",[["2b",string(2b)]])
```

Flow of execution of the **breakpoint**:



- In the first line it is indicated if it is a local or global expression (a local expression can be a local variable).
- In second line the selected expression is indicated.
- If it is about a global expression:
  On the third line the evaluation of the expression is displayed (evaluation of first level). If the evaluation is not realizable an error is displayed in the place.
  On the following line the complete evaluation is displayed if it differs from the evaluation of first level. If the complete evaluation is not realizable an error is displayed in the place. Character strings are entered and displayed exactly as they appear in the entry line of the HOME screen.
- If it is about a local expression:
  On the third line is displayed the evaluation passed in 2nd argument of the command ***brk\pt***.

BreakPoint for TI89 / T92 Plus / V200

LOCAL AND GLOBAL VARIABLES

The TI OS defines two types of variables; local and global. Local variables are variables declared in programs or functions with the *Local* statement, all other variables are global variables.
A local variable is a temporary variable that exists only while a program or a function is running. When the program stops, local variables are deleted automatically.
If a program has local variables, another program cannot access them.
During debugging, replace local variables by global variables so that **Breakpoint** can access theirs values.
The calculator cannot use a local variable to perform symbolic calculations.

LIMITATIONS

BreakPoint can not be used in functions: transform temporarily the functions into programs.
Breakpoint cannot read or modify the contents of the local variables; it is necessary to pass the contents of the variable in argument of the command ***brk\pt*** or to declare the variables as global.
Breakpoint works only with the english OS.

KNOWN BUGS

A variable contained in the inspected expression, or an inspected variable itself, can become inaccessible in reading if it contains a non-algebraic variable (DATA type for example). It is then necessary to stop the program and to generate an error: for example type □. then ENTER in the HOME screen. The variable becomes then again accessible.

VARIABLES USED BY THE PROGRAM

BreakPoint uses the variable BRK\ælstvars to store the list of variables inspected during its successive executions. **BreakPoint** uses temporarily the folder BRKBLÅNK to evaluate expressions, this folder is automatically created and deleted by **BreakPoint**.
The folders BRKBLÅNK and ìíïïðñòó must not exist at the launch of **BreakPoint**.

CUSTOM MENU

The custom menu **BreakPoint** makes easy the insertion of the command ***brk\pt*** in the programs to be debugged. Load the file ***menu*** according to your calculator and run ***brk\menu()***.

CREDITS

Thanks to :

- The TiEmu team for their emulator.
- David Fernando Suescun Ramirez for its Daisuke-Edit.

This tutorial has been wrote with OpenOffice.org suite using the fonts Bitstream DejaVu Serif, TI-89 symbols and Ti89pc.

CONTACT THE AUTHOR

Send me an email at francois.vuillet@laposte.net
If you find a bug, if you have suggestion, DON'T HESITATE ! I will try to answer to ALL your mails. I hope this program will help you.

François VUILLET