**FBDEBUGGER user's manual** (version 2.01 for Fbdebugger version 3.01)

What is FBebugger
How to use FBDebugger



## What is Fbdebugger

- It's a debugger…... for FreeBasic (Windows and Linux)
- It allows to follow the execution of your program (named debuggee) in the source code. You can stop the execution on a selected line or depending a condition.
- The values of variables are displayed and can be edited. Same for the contain of memory. And other features.
- Debugging is made line by line not for each statement so separate instructions on several lines for details on execution.
- There are 2 versions of Fbdebugger one for 32bit and one for 64bit.but only one source code
- Trying to use 64bit Fbdebugger on 32bit exe throws an error message and vice versa.

## Used Files / Registry

Files : Fbdebugger.exe, fbdebugger.ini, Fbdebugger_help.chm/pdf
No change in registry

## How to use Fbdebugger

- create a directory and copy inside Fbdebugger.exe and fbdebugger.pdf
- If you prefer compile you own version :
Fbc -s gui fbdebugger.bas fbdebugger.rc (Windows). Don't forget to put the buttons folder and the icon file with these two files.

ALWAYS save your work before debugging

- compile your code with gas/gas64 and the option -g to add debug data.
- start Fbdebugger
- click on file button and select a exe compiled with
- debugging is starting
- use step button or an other execution button

- Fbdebugger accepts filename in commandline so you can start it in a bat file or from an editor..

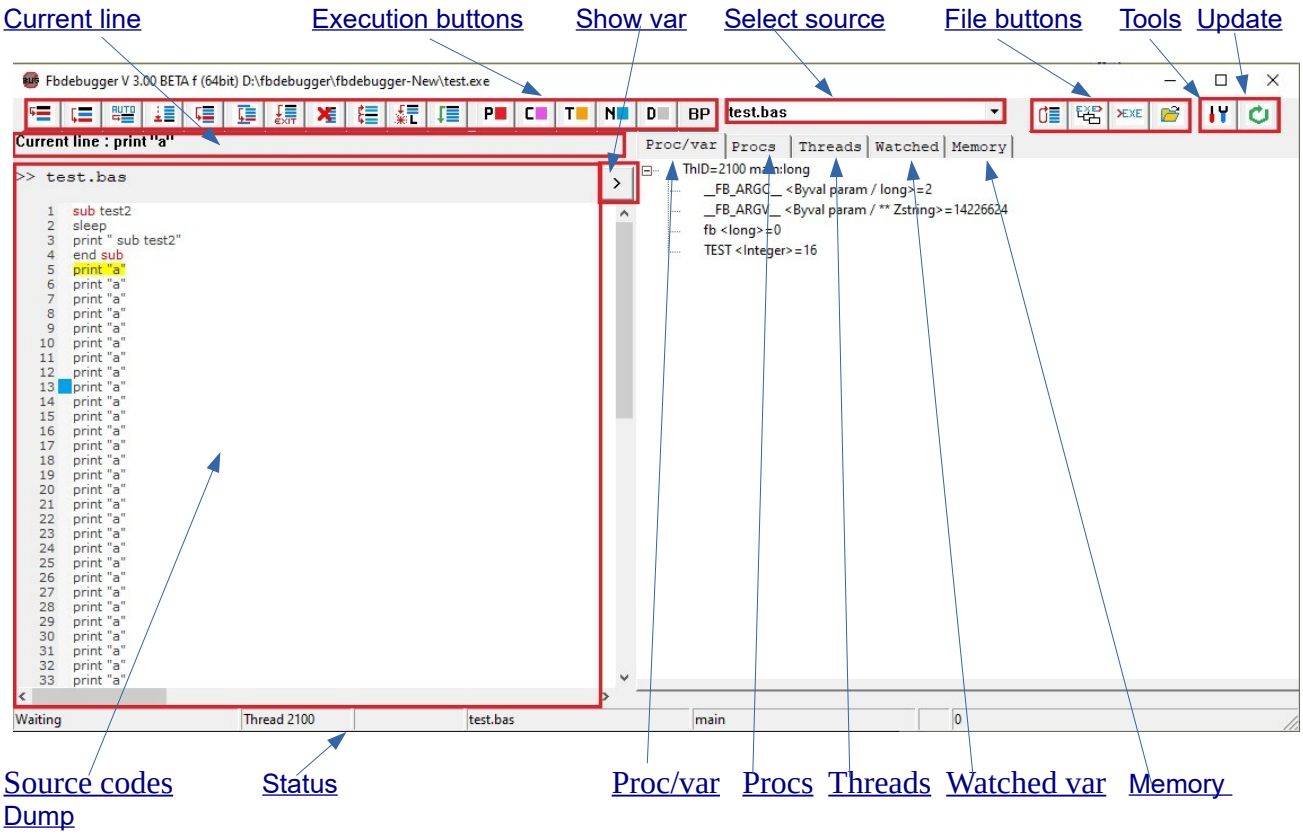
- It's also possible to attach a running program to the debugger Attach exe.
On Linux a system parameter PTRACE_SCOPE must defined with a value of zero.otherwise operation not permitted. To see the current value : $ cat /proc/sys/kernel/yama/ptrace_scope.
If not zero execute : sudo bash -c "echo 0 > /proc/sys/kernel/yama/ptrace_scope"

- Debug data stored inside the exe is procedure and variable names/addresses and pointers for each executable line. To see which lines of code are executable Executable lines

For now multi threaded programs are no well handled so avoid to debug them.

## Main window

- Window title : version of debugger and '64bit' or '32bit' followed by Full Name of debuggee
- Only the whole window is resizable.
- Position/size main window saved and restored when quiting/launching Fbdebugger




## Current line

- Shows the next line to be executed
- Click on to go to the line.

## Status Line

- Execution status
- Current PID/TID
- Current thread ID -
- Current source file name
- Current proc (main or sub or function)
- Number of threads in different states RSBI : running, stopped, blocked, inactive (To be completed)

The status can have one of these values :

- no program : no debuggee has been loaded
- loading debug data : loading all the datas needed for debug, extracted from the exe. Normally very quickly
- loading sources : loading the source codes files

- waiting with reason :
    * no particular reason : before first execution or after step command
    * BP On line : stops after run to cursor command
    * BP xxxx : stops after reaching a breakpoint
    * halt by user : stops after the user halts the debuggee
    * access violation : stops after an access violation or an exception
    * new thread : stops after a new thread is started

- auto : the debuggee is running automatically step by step
- running : the debuggee is running to the cursor, the end of current procedure, the end of program
- released : the debuggee has been released, no debug
- terminated : the debugging is finished (end of program, debuggee killed or crashed)

**Execution Buttons**

To control the execution of the debuggee.

- Tooltips associated with all buttons. (Futur) Disabling [Settings].
- They are usable only when a debuggee is running.
- Most of them have a corresponding option in the contextual menu of the source codes window and a shortcut key.

**- Step** : executes only the current line. [F2]
- **Step over :** executes without displaying every move inside the procedures.[F3]
- **Auto stepping :** automatic execution line by line with a parameterizable pause time [Settings] [F4]
**- Halt** : to be used in any case to stop a running debuggee (run to cursor, freed debuggee, etc.).
The debuggee is stopped not killed, debugging can be continued. [F5]
- **Run to cursor** : executes till the executable line where is the cursor.
The debuggee is also stopped if a breakpoint is reached/true.[F6]
**- Run to end of procedure** : executes until reaching an exit of the current proc (end of sub, return, exit sub, etc). It allows to look at value of variables before exiting the proc. [F7]
**- Run to end of program** : executes until the program is terminated or a breakpoint is triggered. [F8]
**- Kill process :** Kills the debuggee process. To be used carefully. [F10]
**- Modify execution** : The purpose is to skip or rerun lines by changing the next executed line, only inside the same procedure (including the main) to avoid issue with the stack.
First put the cursor on the line that will be the new current one.[F9]
- **Run to crash** : executes line by line until a crash. Useful when the crash arises in a library. The debugger displays the last executed line.

**- Release:** frees debuggee, all the CPU breakpoints are removed but the debuggee can be stopped by the stop button

**Breakpoint buttons**
- One button for each type to set/remove and one to disable/enable existing one.
Button to open the management window. Breakpoints.

**File buttons**
- **Restart :** starts the last debuggee
**- Last 10 exe** : shows for selection the 10 last debugged exes. Select the one to be started
**- Attach running program** : lists all the process. Select one to attach it to the debugger.
There is no way to detach the debugee (even if the feature exists in the APIs) so either go to the end or kill it or free it.
**- Select file** : opens file explorer to select an exe and starts it.

If a debuggee is running when restarting or launching a new program, a message box offers the possibility to kill the current one.

[Top]

**Update proc/var On/Off**

When there are many variables in proc/var tab the update can be very long. This feature allows to control the update for that tab and also dump memory.

 Update is On (default). If update is off clicking on the button triggers the displays.

 Update is Off so when executing step by step the update is not done.

Watched variables stay updated so you can always see the changed values.
However when a proc is called an update is done but the variables are just labelled "Not filled"

**Source codes**

- Displays one code source either the one where is the next executed line or another one according the need (to show a BP by example). The combobox allows to selected any of them.
- The background of the current line is colored in yellow.
- Some keywords (sub, function, operator, constructor, destructor) could be highlighted in red
- A warning is displayed when a source file is newer than the executable file

- On the left the squares (red, blue, etc) indicate breakpoints.
- (future) when the mouse goes back to the source code panel the focus is set on this window.

Contextual menu :
- Set/disable/enable/remove Bps breakpoints
- Reset/change counter for counter BPs
- manage all the defined BPs
- Mark every executable line of the current file with a green.arrow.
- Click on (or near) a variable then on the button [>] to show it in the proc/var window
- (future) Step auto multi threads : steps alternatively automatically each thread. Select the threads concerned by checking the box in thread window.
- Find in text, circular search (when bottom reached go to top and inverse too).
Put cursor on a word or select a text right click and select "Find text" or ctrl+F
- Goto line: enter a number to reach the line in the current code source. (possible only If more than 10 lines)
- Line address : display the address in memory of the first asm instruction of the line.

**Procedures (subs / functions) and variables**

Displays the running procs and data of the associated variables/fields : name, datatype, value, pointed value if pointer. Use verbose mode to get more information about variables :size, offset and real address.

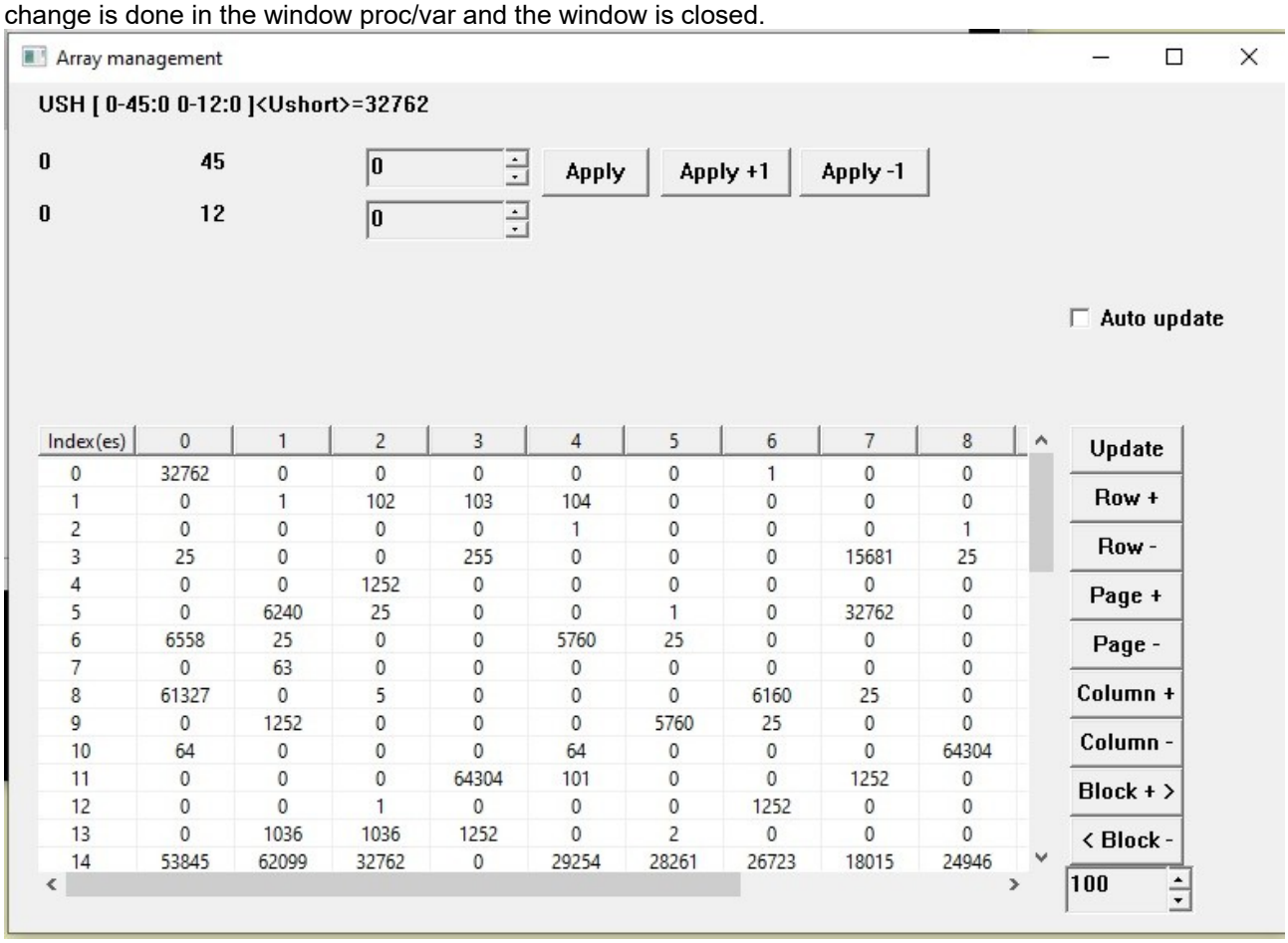
- Procs are sorted by thread.
- Shared and common variables displayed at the top (outside main proc).
- Globals from dll are in a separate part.
- In case of iterators (for loops) with datatype declaration and same name only one is kept as they all have the same adress, to avoid multiple unuseful lines in proc/var tree.
- Variables named lt_xxx are ignored
- Parent trees for inheritance are displayed with the variables added par the compiler (FB_BASE/FB_PVT)- -
- The update is is only done when Fbdebugger is waiting, after each step or on stop.
- Update can be limited by update ON/OFF
- Dynamic arrays could be not defined before execution of some lines so some commands are not usable.

Context menu

- Set watched : add the variable in the watched list w/wo trace
- Break on var value : select the function to be used and eventually enter the value needed for the comparison. Break on var also displayed in the menu.
- Conditional breakpoints

[Top]

- Index selection : only for arrays, you can easily watch the elements one by one
* display of values for 1 or 2 dims, if more only the 2 most right are shown (see select index).
* Select the index and change the value then click apply or use the +/- buttons to increment/decrement. The change is done in the window proc/var and the window is closed.

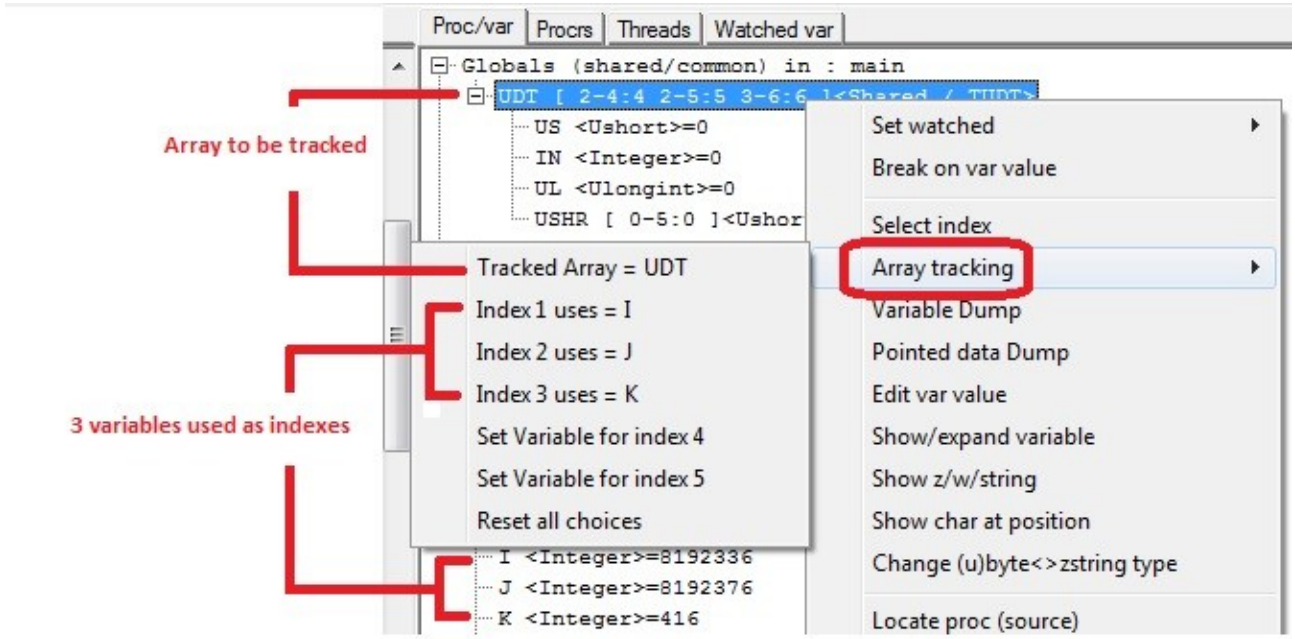



| Index(es) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 32762 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 102 | 103 | 104 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 25 | 0 | 0 | 255 | 0 | 0 | 0 | 15681 | 25 |
| 4 | 0 | 0 | 1252 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 6240 | 25 | 0 | 0 | 1 | 0 | 32762 | 0 |
| 6 | 6558 | 25 | 0 | 0 | 5760 | 25 | 0 | 0 | 0 |
| 7 | 0 | 63 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 61327 | 0 | 5 | 0 | 0 | 0 | 6160 | 25 | 0 |
| 9 | 0 | 1252 | 0 | 0 | 0 | 5760 | 25 | 0 | 0 |
| 10 | 64 | 0 | 0 | 0 | 64 | 0 | 0 | 0 | 64304 |
| 11 | 0 | 0 | 0 | 64304 | 101 | 0 | 0 | 1252 | 0 |
| 12 | 0 | 0 | 1 | 0 | 0 | 0 | 1252 | 0 | 0 |
| 13 | 0 | 1036 | 1036 | 1252 | 0 | 2 | 0 | 0 | 0 |
| 14 | 53845 | 62099 | 32762 | 0 | 29254 | 28261 | 26723 | 18015 | 24946 |

* auto update : all the cells are automatically updated (default = no update)
* update : to update all the cells. Useful only if autoupdate is off.
* possibility of moving by row, column, page, block
* the size of each column can be changed (spin gadget)
* click on a cell to edit its value
* select indexes by CTRL+ clicking on any cell (after window is closed)
- (Future) tracking array associates an array and variables used as index (see below)
- Variable dump : shows memory using the address of the variable as start.
- Pointed data Dump : shows memory of deferenced data
- Edit : to change the variable value. For pointers the value of pointed data is displayed and can also be changed.
- Expands or shows a variable. To show return value of a proc : use directly on the proc name (proc/var tab)
- Show z/w/string : displays every type of string up to 32000 characters, for more use dump.
- (Future) display of the character at a given position in a (z/w)String with possibility to change and to move forward and backward 40 chars (proc/var tab) and select first/last char.
- Locate proc in source : show the beginning of the proc in the sources window.
- Locate calling line : show the line that has called the proc.
- List to log or copy to clipboard : log all the proc/var or only those from one proc (where is the cursor)
- Collapse/expand : collapse or expand the tree.

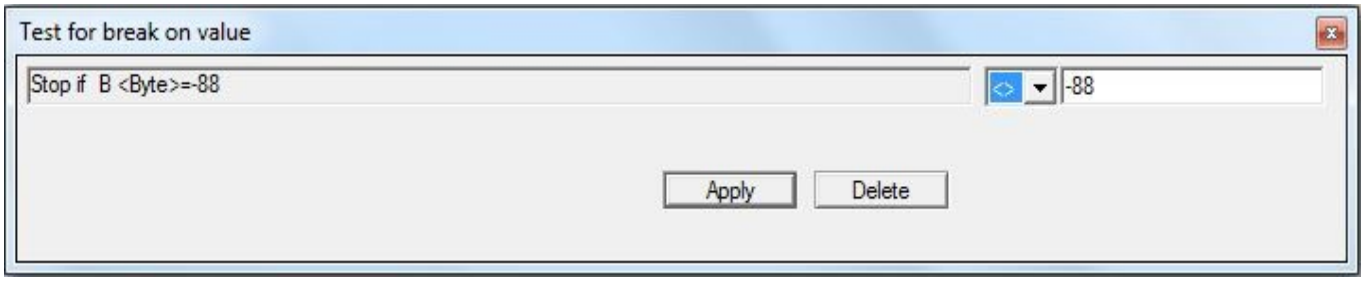
[Top]

- (future) tracking array, usefull to easily follow the values of an array with the help of variables used in loops. Select the array then the variables used as indexes. Then when one index changes the corresponding value in the array is displayed.




**Break on var / on mem**

Like a standard breakpoint, a break on var / on mem stops the debuggee but when a condition is verified. Strings, except wstring, coud be used but the test is done only on the 25 first chars. The test is not linked to a line so at each line executed the condition is tested.




- Possible tests on value : change, equality, difference, etc.
- To set : contextual menu on proc/var breakpoint on var or in memory box.
- To delete : click on delete button.
- a message is displayed when a break on var occurs for better visibility and allowing to keep or remove it.

**Breakpoints**

When a breakpoint l10 max) is defined on a line the running debuggee stops on that line before executing it.

- There are 4 types :
    - permanent (red) stays until removed by user.
    - conditional (purple) the breakpoint is conditioned by a logical comparison between a variable and a constant or another variable.
    - temporary (orange), deleted after have been reached
    - counter (blue), using a counter decremented each time the line is executed and stopping when value is zero eg if counter=10 stops before the 11th execution.

-

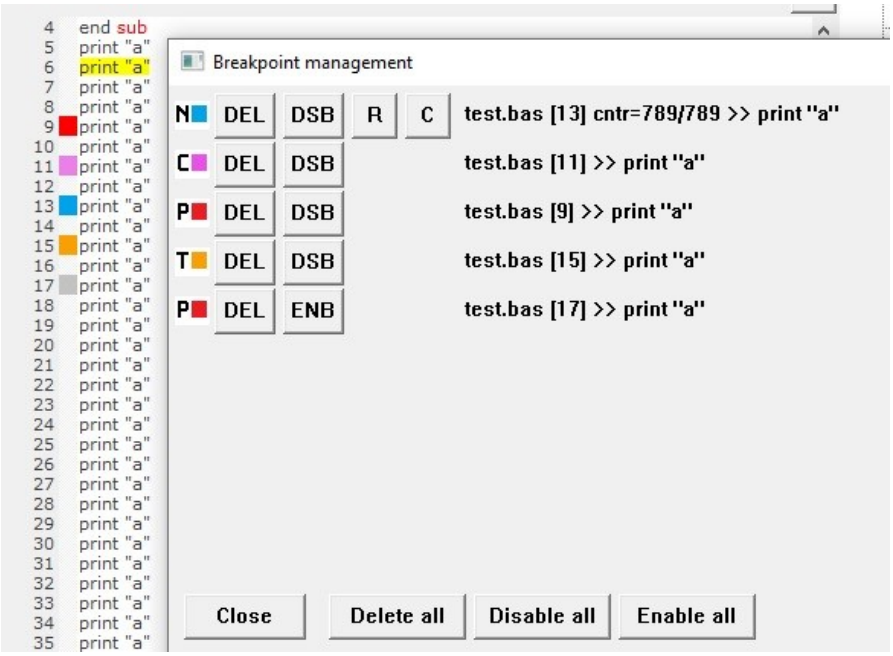
Put the cursor on a line,right click and select option or use CTRL + <P/C/T/N> or the button

- The line must be executable.
- They can be disabled, not active, (grey color) and enabled. CTRL + D
 - (future) Disabling a line without breakpoint disables permanently the line.

- The breakpoints are kept when restarting the debuggee. You can possibly delete one or more.

- Breakpoints are saved in the ini file.
Caution : breakpoints in a dll are displayed only if the dll is loaded. An "invisible" breakpoint could be removed when creating a new one if the limit of 10 is reached.

- a dialog box allows to manage all the breakpoints in one view. CTRL + B or BP button
Open automatically on start of debuggee.



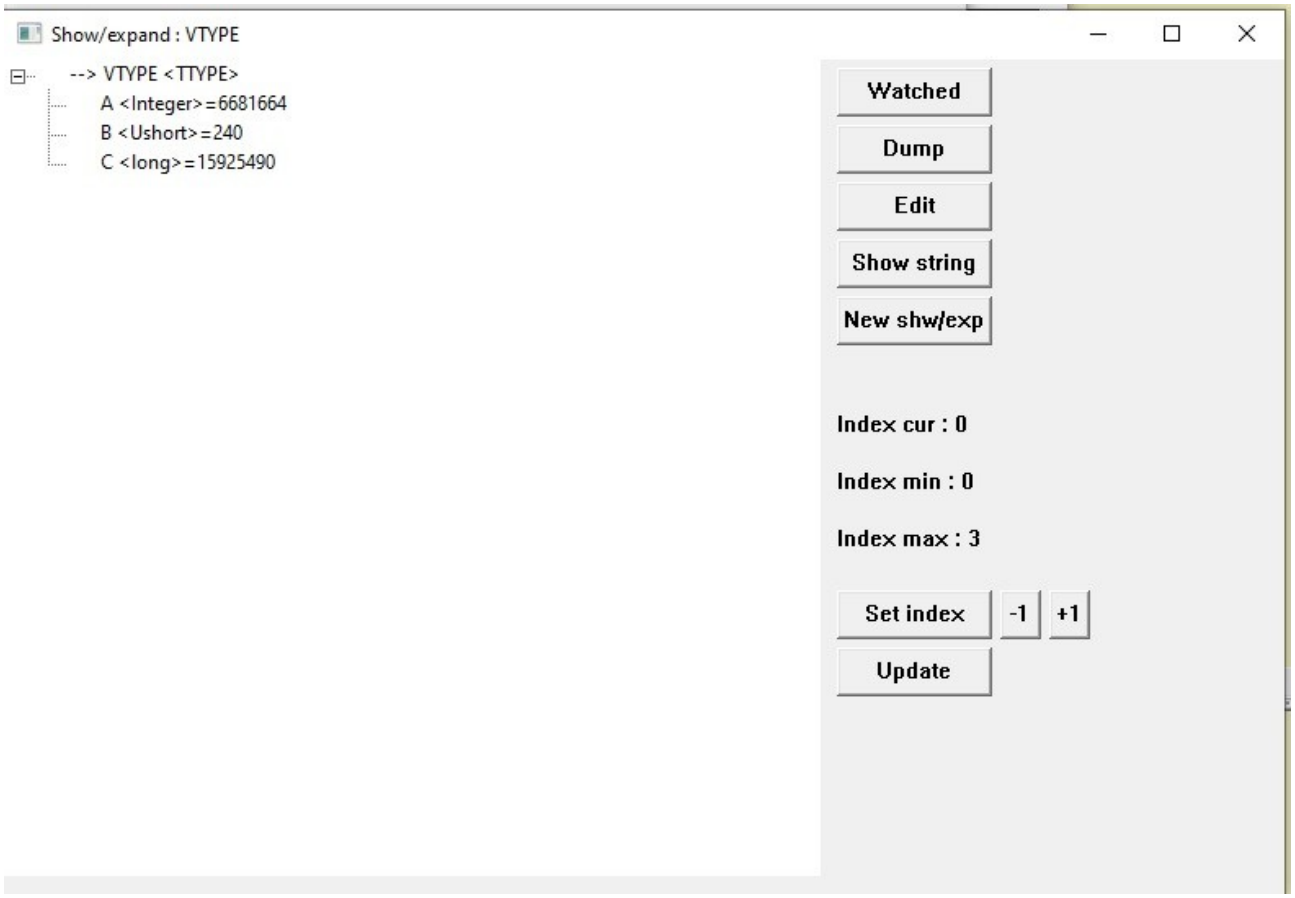

- DEL : to delete one BP
- ENB / DSB.: to enable / disable according current state
- R / C : only counter BP reset the initial value or change the value which are also displayed

- Click on a row : selection and display of the corresponding line in code source
- Close : No change and exit
- Delete all : delete all the existing breakpoints
- Disable/enable all : disable/enable all

[Top]

**Show/expand**

An easy way to see values for udt or pointed area. Option in proc/var and watched context menus.




- Arrays inside structure are handled.
- There are the same possibilities than with the variables (add watched, dump, edit..)
- New expand : replace with the selected element
- Index cur : shows current index for array. In case of multi dim it's the last dim.
- Index min/max : shows low and upper bounds.
- Set index : to change the index value in the range (min/max).
- +1/-1 : to increase/decrease the index in the range.
  These actions update automatically the display. See below.

- 'Update' : update the display (not automatically done as in proc/var tab)
Be careful the data could have been deleted

- The shw/exp window is closed when the procedure owner of the variables is closed, except for statics.

- When a dynamic array is re-dimmed use 'Update ' button. A message box is displayed for warning (full update or only indexes) , the min/max indexes are updated and the current is set in the new range.
Don't use +-1/set index before 'Update' as the address is not yet updated…..
Not done if the array is a field in a structure, in this case close and redo shw/exp.

[Top]

**Procedures**

Displays all the procs (running or not) with file source and address (verbose mode). For operators, destructors, constructors Fbdebugger recreates the full names.
- Check one for disabling it : the proc is excluded from debugging. BE CAREFULL.

Context menu
- Locate proc : select a proc name to find its location in the code source files and in proc/var tab or select the proc and then just press shift key.
- Procs followed : select an option to follow or not all the procs.
- Toogle sort : toogle between the procs sorted by their name or by the name of the module where they are coming from.
- automatic constructor/destructor added by the compiler are only visible in the procs tab.
- Proc call chaining to follow in one page : call chain
  Closed if running, (options in proc/var Tab and thread tab
- procs are sorted by thread in proc/var windows

- Naked procs are allowed.

**Threads**

Displays all the threads (10 max) with ID and in verbose mode the handle (settings) then shifted every proc attached to the thread (call stack).
- The current thread is indicated by : (next execution).
- In case of automatic change of thread (mutex or threadcreate) a message is displayed giving the choice of which thread will be the current.(see below)
- Use the checkbox to indicate if the thread must be executed in case of multi threads.
- Multi threads and step auto : step auto alernates between threads (those checkboxed)

Context menu.
- Select next thread to be executed : the selected thread becomes the current thread.
- Show next executed line : in the window sources code, the line corresponding to the thread is highlighted and displayed (tab changed if needed).
- Show line creating thread : in the window sources code, the line which has launched the thread is highlighted and displayed (tab changed if needed).
- Show first proc of thread : in the window sources code, the line which is at the begin of the thread is highlighted and displayed (tab changed if needed).
- Show proc : in proc/var window select the current proc of the selected thread.
- Show proc : in sources code window select the current proc of the selected thread.
- Addresses : Displays the begin, end and stack adresses for the current proc of the selected thread.
- Kill : terminate the selected thread. Be carefull.
- Expand one thread : displays all the procs of the thread.
- Collapse : collapse all the threads, to display only the lines with thread id.
- List all threads : displays all the threads of the process included the threads not debugged.
- Show register values 32/64bit for the current thread

- For multi threaded programs WIP. Threads can be running, stopped= waiting something (eg a mutex), blocked by a user, inactive = not debugged (threads not managed by the user)

**Calling/called procedure chain**

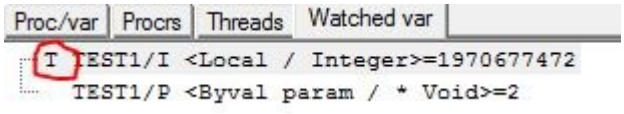
There are 2 ways to see the chain of all the calling/called procs.
1/ In thread tab
2/ Calling chain (see context menu in proc/var tab and thread tab) : all the procs for a thread are listed.
Click anywhere on one line to navigate in the chain of procs : the calling line is highlighted in source code and the proc is selected in proc/var tab.
The dialog box is canceled by any execution (step, run, etc) .

[Top]

**Watched variables/memory**

Watched variables are an easy way to keep an eye on some specific variables or part of memory.
Up to 20 watched values (variables/memory), updated also when auto mode

- add a variable in the watched list via an option in proc/var.
- For memory use the manage dump dialog.
- Variables or memory could be traced in the log when their values change. A 'T' to the left of name indicates that the var is traced. In the watched context menu and also in the context menu of proc/var.




- Different formats (hex, binary,..) for watched vars, select twice or more to get a new format
- Local var watched are kept between proc calls but are displayed as non-existent and can just be deleted.
- If you rerun immediately the debuggee without closing Fbdebugger, all the watched are kept. In case of dll only standard types and pointers are reused.
- When you close Fbdebugger the arrays and memory watched are removed. Then later on run of debuggee proc, var name, pointer, array are used to verify if the watched vars still exist.
- The watched are saved in the ini file for a next session.
- Some issues could be happened with vars in recursives procs.

Contextual menu

- Show in var window: selects the variable in the proc/var
- There are the same possibilities that with the variables (dump, edit..)
- Toggle tracing : set/reset the tracing mode in the log
- Cancel all tracing : the tracing is suppressed for every watched
- Delete : delete the selected one.
- Delete all : delete all with confirmation.

[Top]

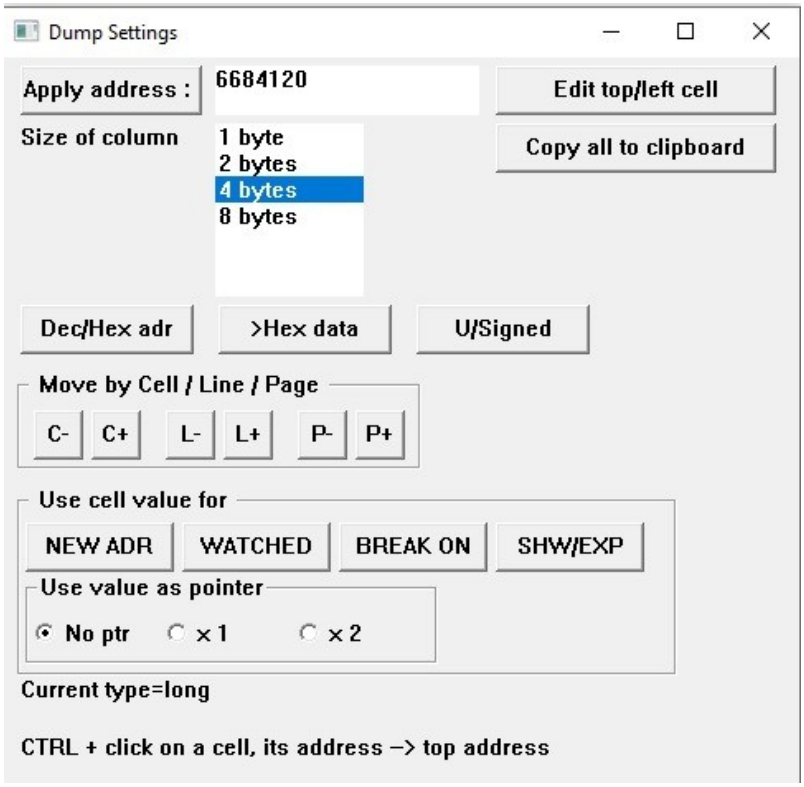
**Memory Dump**

- Displays a part of memory  by line of 16 bytes with auto update (step or auto) unless update off
- Beginning on address of variable or free address
- Formated (byte,ubyte,integer,etc/DEC,HEXA) according to datatype. Possible changes in dump settings.
- On the right side the equivalent in ascii of the 16 bytes, below value 31 replaced by a dot.

- Click on a column header to change a value (only first row)
- to change the value of any cell in dump memory : left click on it
- to copy dump into clipboard : shift+left click on one cell / on header for whole page / address area or ascii area for one line or multilines

 the dialog box to valid selection is automatically opened if currently closed
- to change the beginning address of dump : ctrl + left click on one cell (data or address area), datatype must be integer/uinteger
- to use the value in a cell as a pointer or directly as beginning of udt data in shw/exp : alt+left click on a cell then select no pointer or pointer and the datatype. The dialog box to valid selection is automatically opened if currently closed.

- Right click to display a box offering the possibility to change format (datetype,dec/hexa), to move into the memory (address,forward/backward), to add a watched memory and create a breakpoint on memory change.



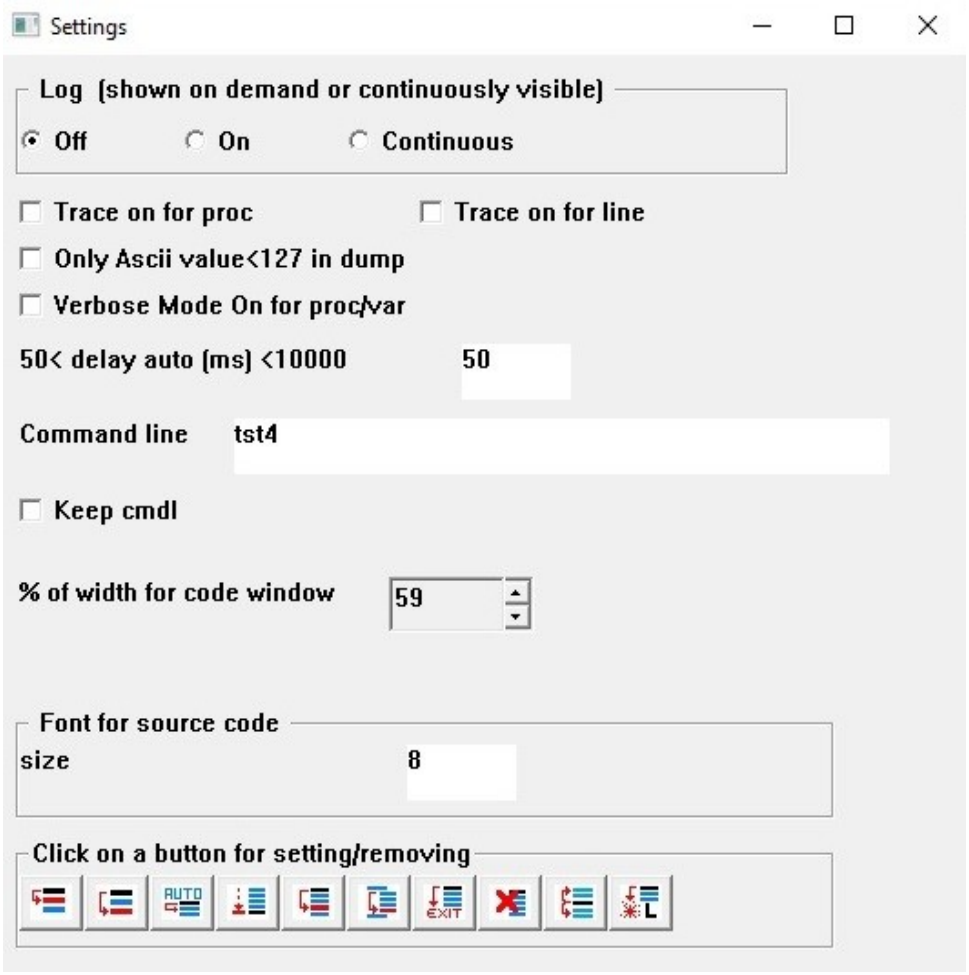

[Top]

## Tools

Settings : to set parameters for Fddebugger.
About : fbdebugger version and author
Feel free to send me (debug AT aliceadsl DOT fr) bugs, remarks, suggestions and improvement requests
And an option to launch a browser to the dedicated page on the freebasic's forum
Compile info: to displays the version of the compiler FBC and (future) the debuggee compilation date.
Help: displays help, F1
Show log : shows the log as a code source
Hide log : supress (close) the log in the source codes tab.
Delete log : deletes the log
List Enum : list of named enums with all texts and values
Process list : just a list of all the running process.
DLL list : just a list of all the **debugged** dll.
Translate win message / Linux error : Enter the code to get the corresponding text.
Bin/hex/dec : enter a decimal or hexa (&hxxx) value to get the value in dec/hexa/bin.

[Top]

## Settings



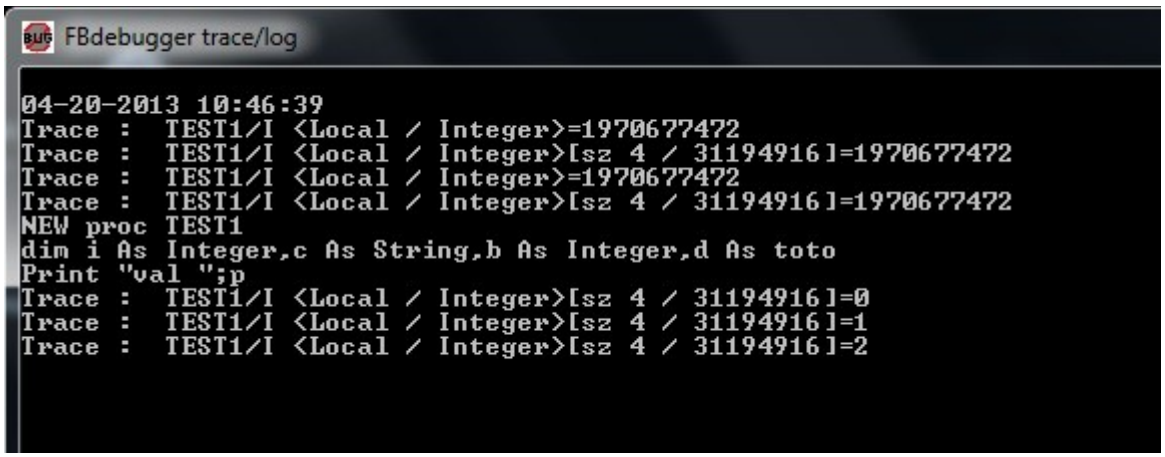

- Some parameters are directly applied others take in account when exiting the window.
- Most of settings are saved in the ini file

- Activate different use of log default is none..
- Trace of executed procs and lines stored in log
- Only asciii<127 in dump memory
- Toogle verbose mode on/off, to get address and advanced information

- Select the delay between every line execution. Min (50) or to max (10000)

- Enter parameters used as commanline for the debuggee. One time or always
- Set the width of main windows (percentage of code window / total width)
- (Future) Change the font type/size/color/ (8,10,12)
- Disable or enable some execution buttons

**Log**
Some informations about the execution (system, procs and lines, procs and vars, watched var) could be displayed on a separate window.[settings].




- Displays log in tools option to open the window.
- New data is appended so use delete log to reset the view.

**Access violation**
Access violation (memory error) and some similar exceptions are trapped and Fbdebugger waits on the faulty line.
You can change variable values and/or skip (not execute) the concerned instructions or terminate the program.

The error line is displayed with some information (Read or Write error,..).
Click on yes to continue. Then skip the current line to execute the last one by putting the cursor on print "end" and press M or click the Modify exec button.

[Top]