FBDEBUGGER user's manual (version 1.21 for Fbdebugger version 2.98.1)

What is Fbdebugger

- It's a debugger for FreeBasic, for now only on Windows.
- It allows to follow the execution of your program (named debuggee) inside the source codes. The values of variables are also displayed and could be modified. Obviously there are some other advanced features.
- Debugging is made line by line not for each statement. To accelerate some operations keep them on one line or 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.

Summary :

How to use Fbdebugger

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

ALWAYS save your job before debugging

- start Fbdebugger *
- click on file button and select a exe **
- debugging is starting.
- use step button or an other execution buttons

* if Fbdebugger is already running a message box is displayed offering the choice to stop or to continue with one more instance.
** In all cases the compilation must be done with the -g parameter to add the debug data inside the exe. These data are procedures and variables names and pointers for each executable line.

Nota some lines, such dim shared foo as integer, are not executable because the compiler need not to generate assembly code. To see them no executable lines

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

-With Fbedit :
Select debug/quick run option in the options menu and enter (or use the file explorer button) the path/name of Fbdebugger. Then in the make menu select run with debug launch Fbdebugger with the debuggee.

Another way : select the tools menu option in the options menu and create an entry Fbdebugger.
Then in the tools menu a new option offers to launch Fbdebugger without parameters.

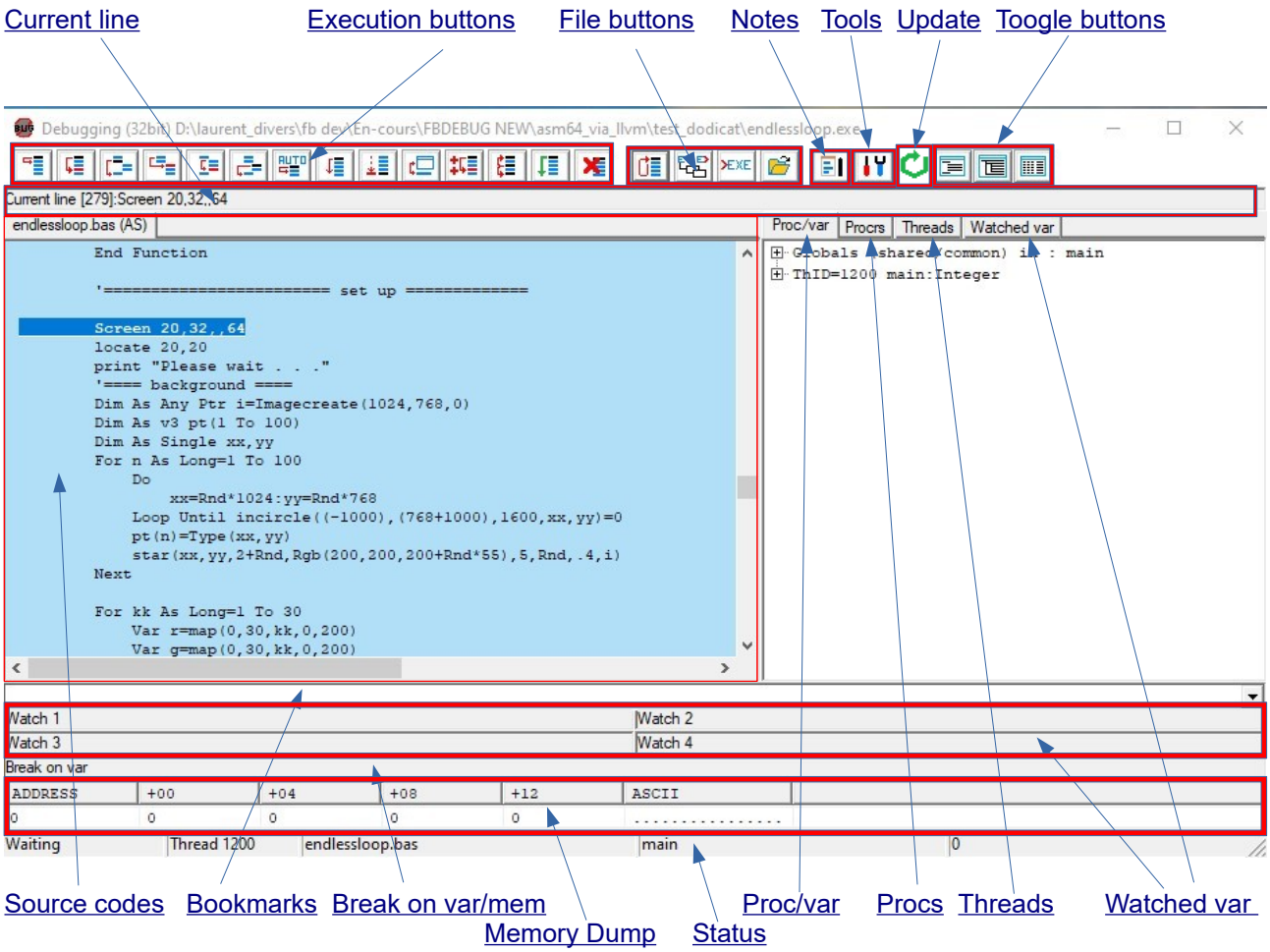
- You can drag and drop an exe directly on Fbdebugger inside the file explorer.
- You can also drag and drop an exe anywhere on the main window of Fbdebugger the exe is immediatly debugged.

- You can do the same with a .bas. It is firstly compiled then debugged. In this case some informations about FBC.exe must be filled [Settings]

[Summary]

Main window

- Version debugger ‘64bit' or '32bit' displayed in the window title followed by Path and Name of debuggee
- Only the whole window is resizable. But the toggle buttons expand some of the main windows.
Position/size main window saved and restored when launching Fbdebugger

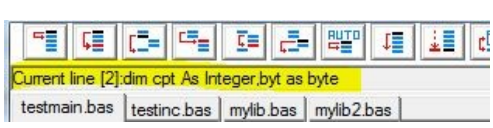


Toogle buttons

Each of these buttons enlarges the corresponding window :

 for code source /  for variables-proc-threads /  for memory dump.

Current line



- Shows the next line to be executed, with its line number
- Click on to reach the line.

[Summary]

Status Line

- Execution status
- Current thread ID
- Current source file name
- Current proc (main or sub or function)

- Elapsed time is displayed in status area when using run or fast run.

The execution could 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
    - cursor : stops after run to cursor command
    - tempo break : stops after temporary breakpoint
    - break : stops after permanent breakpoint
    - break var : stops after variable breakpoint
    - break mem : stops after memory 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, command run
- fast running : the debuggee is mode fast running, command fast run
- released : the debuggee has been released, no debug
- terminated : the debugging is finished (end of program, debuggee killed or crashed)

[Summary]

Execution Buttons

To control the execution of the debuggee.

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

Step : executes only the current line.
Hotkey = S

Run to cursor : executes till the line where is the cursor. First put the cursor on an executable line.
The debuggee is running until it reachs the line under the cursor or a breakpoint is true (reached line, change of value for var or mem).
In the same time the "path" of procedures is stored. In fact only the display is not updated. For doing all those operations, each executed line triggers a CPU breakpoint.
Hotkey = C

Step over proc : executes without displaying every move inside the procedures.
Altough all the standard operations are done so its not speedy.
Hotkey = O

Step top : executes until reaching the first executable line of first called proc.

Hotkey = T

Step bottom : 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.

Hotkey = B

Step out proc : executes lines till going out of the current procedure. Not possible if procs are executed before main (constructor for shared)
Hotkey = E

Auto stepping : automatic execution line by line with a parameterizable freeze time [Settings]
Hotkey = A

Run: Same as run to cursor without the debuggee halting on the cursor line
Hotkey = R

Stop : 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.
Hotkey = H

Mini screen : a sort of remote command that could be moved by dragging (stay clicked on red area and move the mouse). Usefull to see other windows behind the debugger.
Click on again to restore the normal display.

[Summary]

 **Fast run :** very fast execution till the line where is the cursor but no possibility of break on var or on mem as standard run.
First put the cursor on an executable line.
The fast mode just lets CPU breakpoints on the cursor line and every breakpoint line. So the execution is like without debugging.
When the debuggee is "stopped" Fbdebugger, exploring the memory, creates the tree of procedures and variables.
In case of infinite loop use the stop button. The execution time is available by show fast run timer in tools.
Hotkey = F

 **Modify execution** : The purpose is to skip or rerun lines by changing the next executed line, inside a procedure (including the main).
First put the cursor on the line that will be the new current one.
Hotkey = M

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

 **Kill process :** Kills the debuggee process. To be used carefully.
Hotkey = K

File buttons
If a debuggee is running when launching a new one, a message box offers the possibility to kill the current one.

 **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 the debugger to this running program. Usefull in case of infinite loop.

 **Select file (exe or bas)** : opens file explorer to select an exe or a bas file.
In case of bas file, the file is compiled and the exe launched after confirmation. If any errors they are displayed.

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 new proc is started an update is done but the variables are labelled "Not filled"

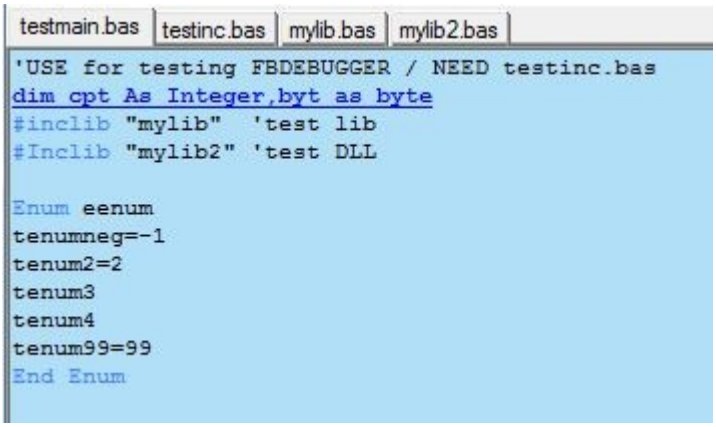
[Summary]

Source codes

- Displays one code source file by tab.

- (AS)"," (CS)"," (CD)" displayed according the compiler : A=GAS or C=GCC)

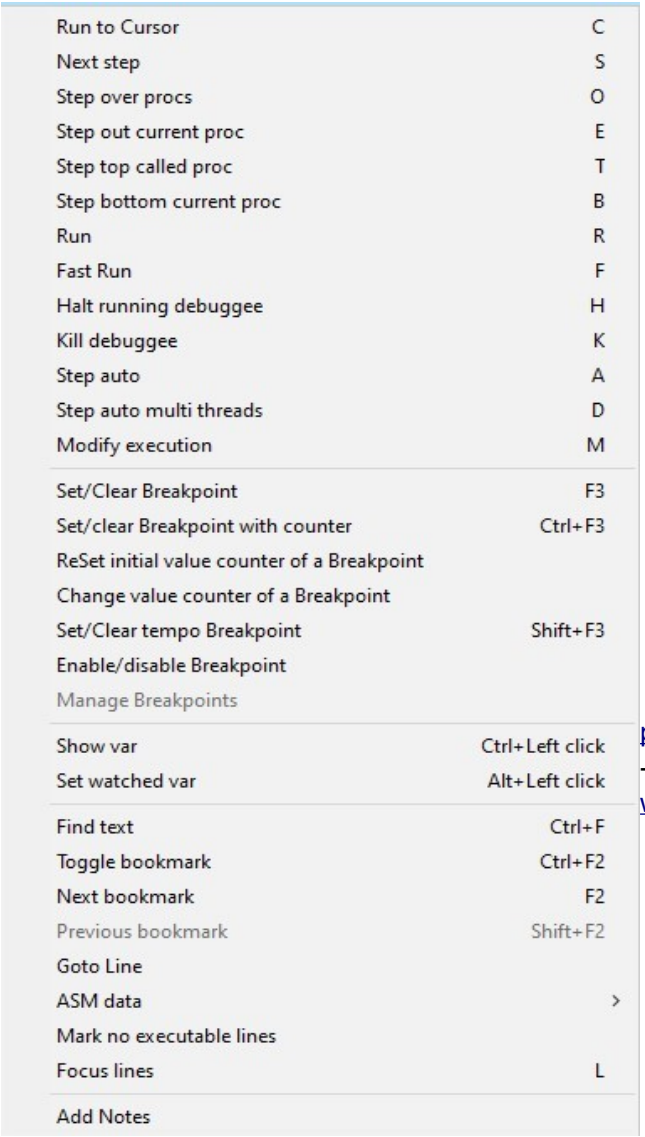
and used debugging data : S=stabs or D=dwarf



- The current line is colored in blue or in purple if its also a breakpoint (default colors). Change in Settings.
- The keywords could be highlighted by a different color. Change in Settings.
- In red a permanent breakpoint and in orange temporary one.
- Beginning of procs can be marked in red (keywords sub and function).
- Operator, Property, Constructor, Destructor, Public and Private are displayed in red like Sub And function
- A warning is displayed when a source file is newer than the executable file
- Each file size up to 5 000 000 bytes.
- when the mouse goes back to the source code panel the focus is set on this window.

[Summary]

Right click to display the context menu



Run to cursor
Next step
Step over
Step out
Step top
Step bottom
Run
Fast run
Halt
Kill debuggee
Step auto / Step auto multi threads (see note below)

Modify execution

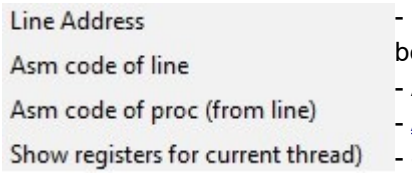
Set/reset/manage breakpoints

Ctrl+Left click on (or near) a variable to select it in the proc/var window
- Alt+Left click on (or near) a variable to add it in the watched list

- 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/near a text right click and select "Find text" or ctrl+F
- Goto line : a box to enter a line number to reach it in code sources windows, current tab. Shows also the current line number.

- Asm information



- Line address : displays a box with the address in memory of the beginning of the line.
- Asm code of line with the correponding basic code line
- Asm code of procedure from current line
- Show register values 32/64bit for the current thread

[Summary]

- Mark no executable lines : every line not executable of the current tab is colored in green. To restore the initial color reset/set highlighted keywords.

testthread.bas

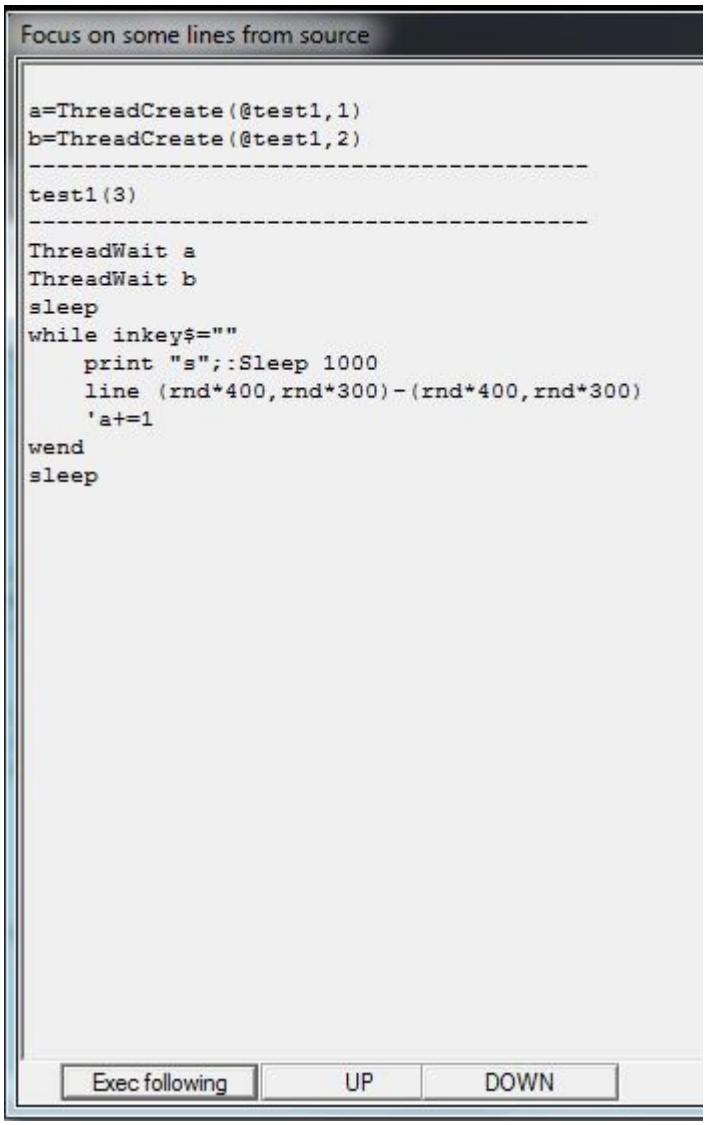
```
' For testing thread
declare sub test1(p As Any ptr)
Declare Sub test2(p As Any ptr)
Type toto
  As Integer a
  As Integer b
End Type
Sub test2 (p As Any Ptr)
  Print "inside test2"
  Print "yes and then... valeur origine=";p
  'Sleep 10000
End Sub
sub test1(p As Any ptr)
dim i As Integer,c As String,b As Integer,d As toto
Static j As Integer
Print "val ";p
d.a=15
test2(p)
j+=1:b=j
    for i=1 to 2
       c="test ("+Str(b)+")"+Str(i)
       Print c
    next
j+=1
end sub
```

- Focus line : opens a special window pointing on the selected line.
- Add notes : adds the line under the cursor

[Summary]

Focus Line
This window, without wrapping, could be used to follow the executed line or an other part of a source module.

- To open it select the option in context menu of source code window or hotkey L.
- If "exec following" is selected the current line (between the two dashed lines) is updated with the execution.
- Up/down to navigate inside the file.

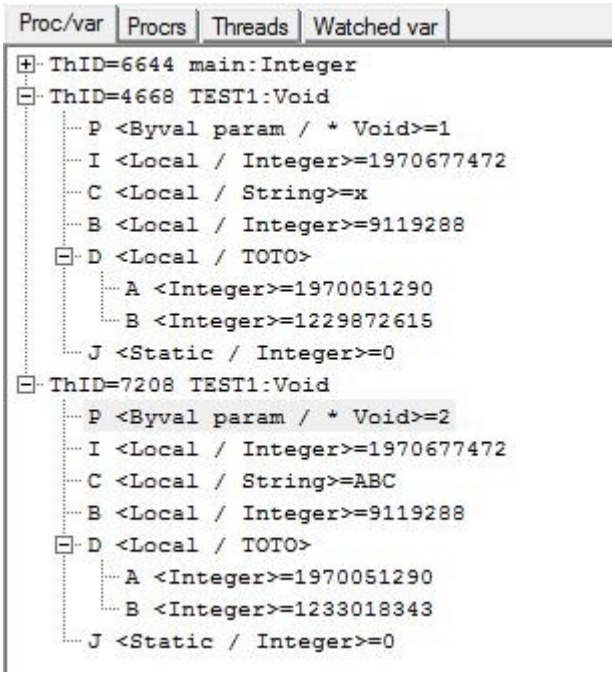


[Summary]

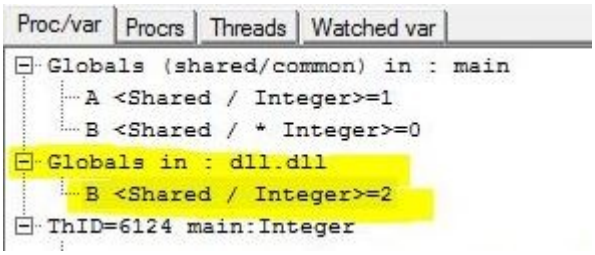
Procedures (subs / functions) and variables

Displays the running procs and datas of the associated variables : name, datatype, value, pointed value if pointer

Use verbose mode for getting more information as address.



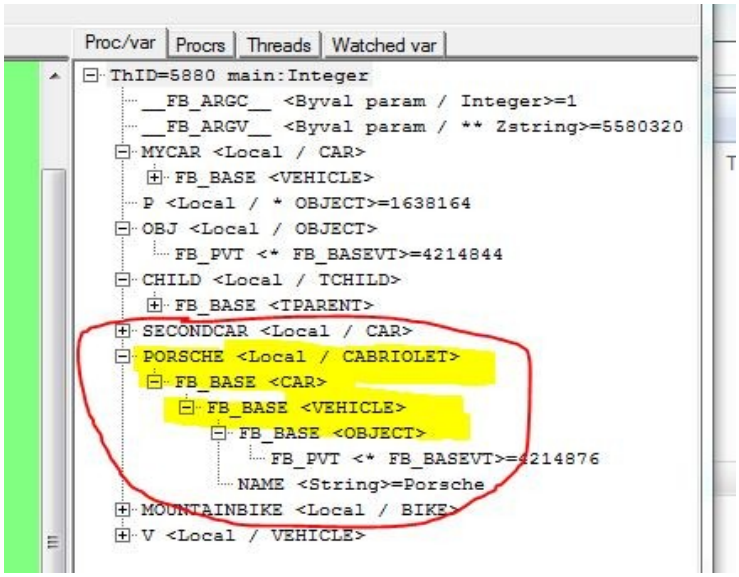
- Procs are sorted by thread.
- Shared and common variables displayed at the top (outside main proc). Globals from dll are in a separate part.



- Hovering on a variable (proc/var tab) few seconds automatically opens a shwexp window (by default disabled, enabled in Settings)
- 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 (only GAS version)
- Variables named lt_xxx are ignored

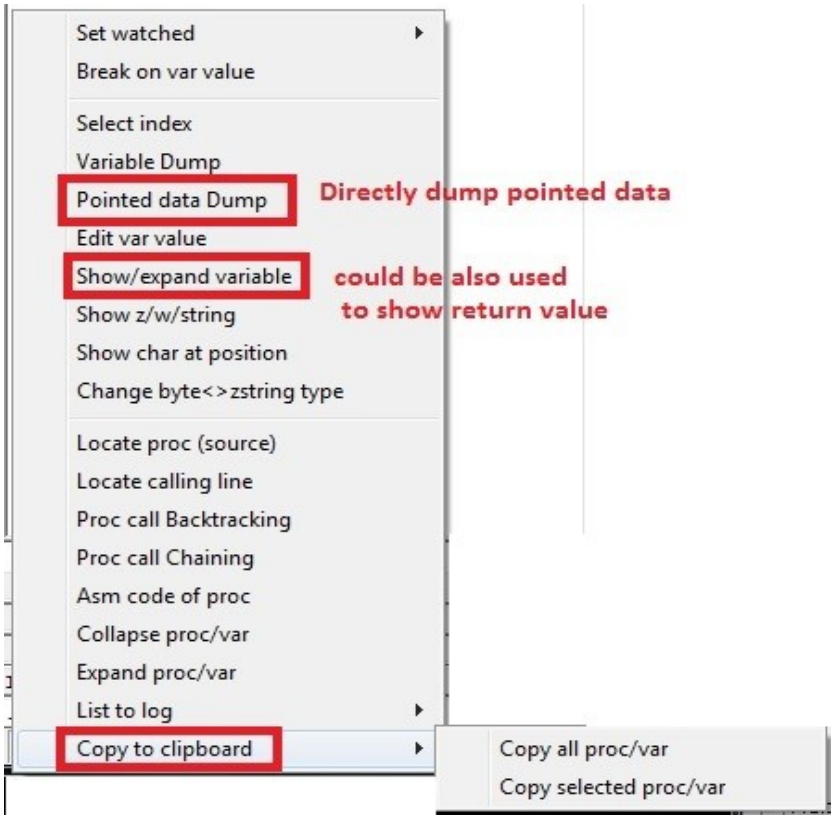
[Summary]

- Parents tree for inheritance are displayed with the variables added par the compiler



- The update is is only done when Fbdebugger is waiting, after each step or on stop.
- Dynamic arrays could be not defined before execution of some lines so some commands are not usable.

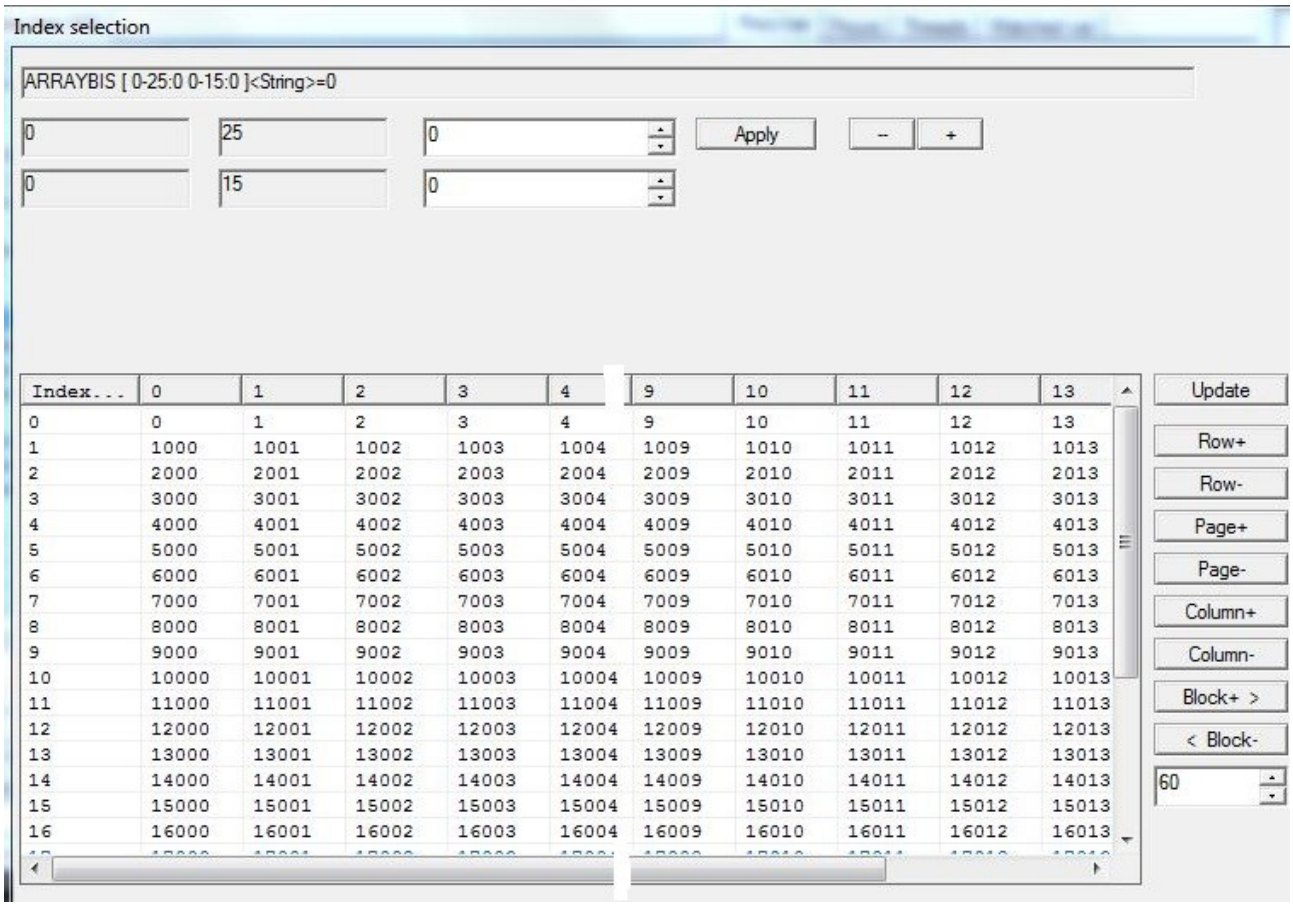
Right click to display the context menu, in most of case firstly select a variable before selecting the option menu



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

[Summary]

- Index selection : only for arrays, you can easily watch the elements one by one. Select the index and change the value then click apply or use the +/- buttons to incremente/decremente. The change is done in the window proc/var.
- allowing multiple index selects for showing different arrays (or the same one with different indexes)



Index...	0	1	2	3	4	9	10	11	12	13
0	0	1	2	3	4	9	10	11	12	13
1	1000	1001	1002	1003	1004	1009	1010	1011	1012	1013
2	2000	2001	2002	2003	2004	2009	2010	2011	2012	2013
3	3000	3001	3002	3003	3004	3009	3010	3011	3012	3013
4	4000	4001	4002	4003	4004	4009	4010	4011	4012	4013
5	5000	5001	5002	5003	5004	5009	5010	5011	5012	5013
6	6000	6001	6002	6003	6004	6009	6010	6011	6012	6013
7	7000	7001	7002	7003	7004	7009	7010	7011	7012	7013
8	8000	8001	8002	8003	8004	8009	8010	8011	8012	8013
9	9000	9001	9002	9003	9004	9009	9010	9011	9012	9013
10	10000	10001	10002	10003	10004	10009	10010	10011	10012	10013
11	11000	11001	11002	11003	11004	11009	11010	11011	11012	11013
12	12000	12001	12002	12003	12004	12009	12010	12011	12012	12013
13	13000	13001	13002	13003	13004	13009	13010	13011	13012	13013
14	14000	14001	14002	14003	14004	14009	14010	14011	14012	14013
15	15000	15001	15002	15003	15004	15009	15010	15011	15012	15013
16	16000	16001	16002	16003	16004	16009	16010	16011	16012	16013

- autoupdate in 'index select'
- autoupdate checkbox in index windows (default = no update)
- display of values for 2 dims or more arrays. if more than 2 dims only the 2 most right are shown (see select index)
- possibility of moving by row, column, page, block and of changing size of column
- select indexes by clicking on any cell
- table to display elements of arrays (see select index option), moving by line/page and select an index by clicking on

- watched var for dll : only basic types and pointers are kept between two debuggings

- Variable dump : the address of the variable is used as the beggining of the memory area.

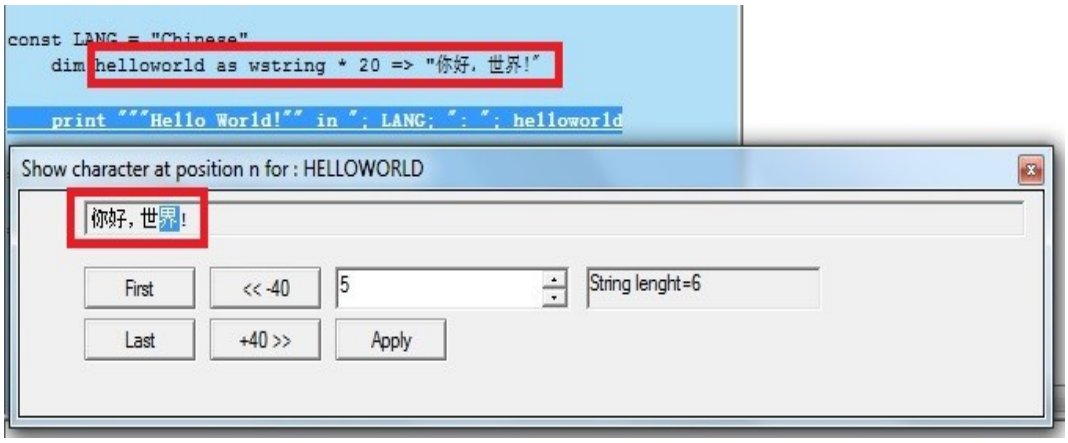
- To show the dump of deferenced data : select new option, pointed data dump, in contextual menu
- Edit : Enter a new value and click apply to modify the variable value. For pointers the value of data pointed is also displayed and could be changed.

- Expand : expands or shows variables. 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. Click on wrapping button to wrap/nowrap.

[Summary]

- 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.



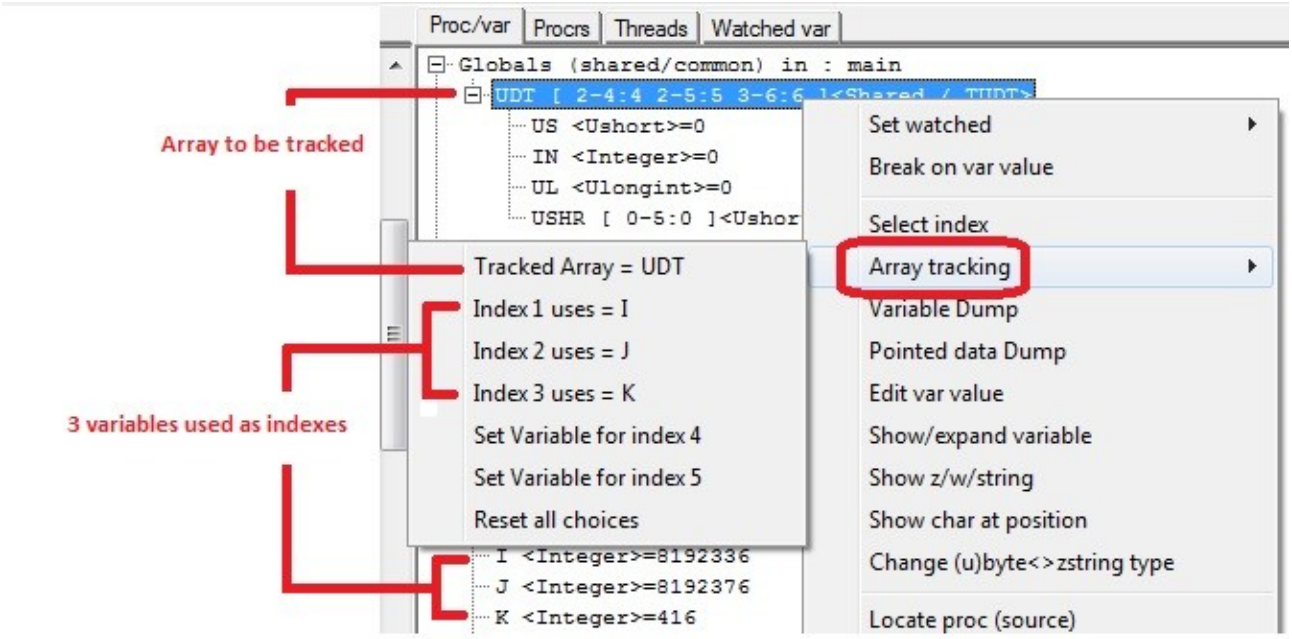
- Change ubyte<>zstring type : in case of gcc compil the zstrings are seen as byte. Change the type ONLY in Fbdebugger to easy manipulation.
- Locate proc in source : select the beginning of the proc in the sources window.
- Locate calling line : select the line that has called the proc.
- Collapse/expand : collapse or expand the tree.

- For searching any text in the tree. Warning : the search is sensitive
- List proc/var : list in the log all the proc/var or only those from one proc (where is the cursor)

- To copy variable value in clipboard (all or only one with its childs) : select the option, copy to clipboard, in contextual menu

- List in log. all proc/var in log or just selected one

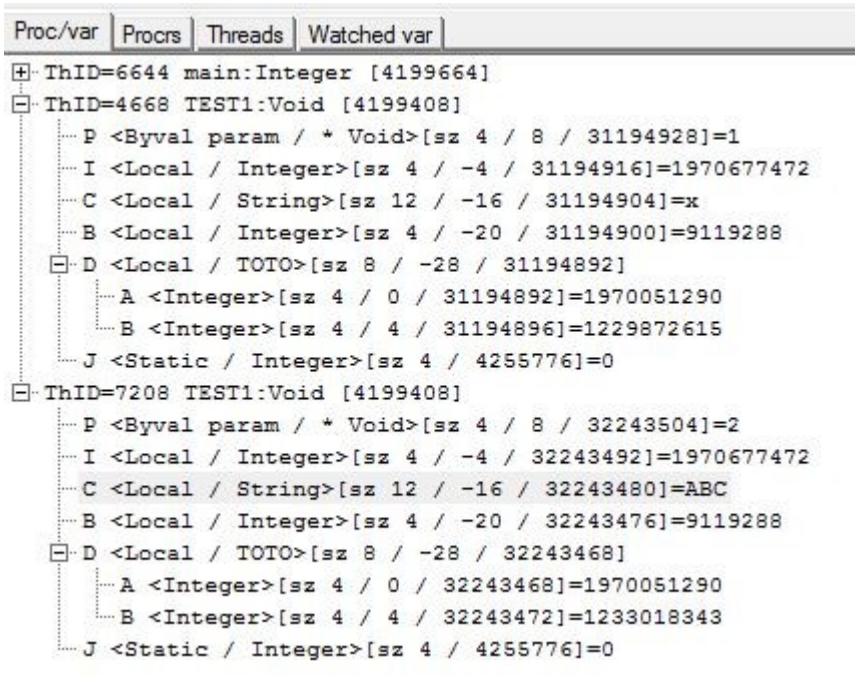
- tracking array, usefull to easily follow the values of an array with the help of variables used in loops.



Select the array then the variables used as indexes. Then when one index changes the corresponding value in the array is displayed.

[Summary]

Verbose mode
Use this option (Settings) to display more informations with variables and procs.



- Variables : size, offset and real address
- Procs : beginning address

[Summary]

Breakpoints

When a breakpoint line (10 max) is defined the running debuggee stops on that line before executing it. See also break on var/mem

- There are 3 types :
    - permanent (red by default) never deleted.
    - temporary (orange by default), deleted after have been passed
    - semi-temporary (red by default) using a counter decremented each time the line is executed and stopping when zero is reached eg if counter=10 brkp the 11th execution.

```
a=ThreadCreate(@test1,1)
b=ThreadCreate(@test1,2)
test1(3)
ThreadWait a
ThreadWait b
sleep
while inkey$=""
    print "s";:Sleep 1000
```

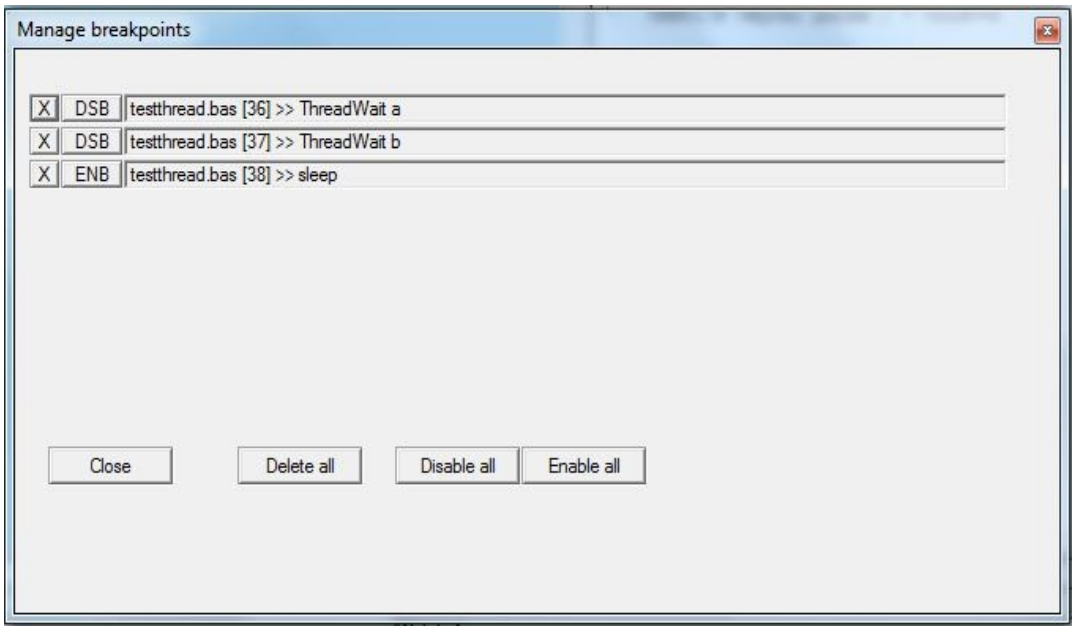
- Put the cursor on line,right click and select option or use F3 (permanent) or shift+F3 (with counter) or ctrl+F3 (temporary)
- They could be disabled (grey, fixed color) and enabled.- Disabling a line without breakpoint creates a disabled permanent one.

- The line must be executable.
- The line breakpoints are kept when restarting the debuggee. You can eventually 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.

The breakpoint lines are managed via a window (source codes context menu):

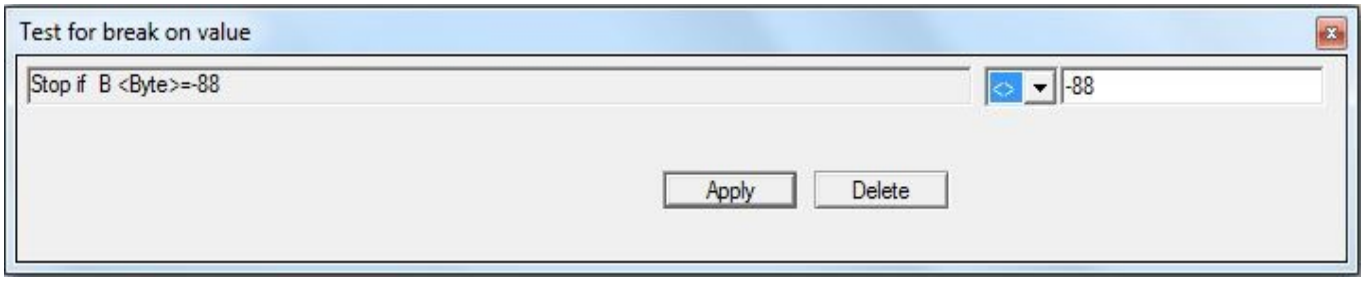


- Click on 'X' = delete
- ENB/DSB.= enabling:disabling/
- Close = No change and exit
- Delete all = delete all the existing breakpoints
- Disable/enable all = disable/enable all
- Click on a row = selection and display of the corresponding line in code sources

[Summary]

Break on var / on mem

Like a standard breakpoint, a break on var / on mem stops the debuggee but only when a condition is verified. Strings, except wstring, coud be use but the test is done only on the 25 first chars.

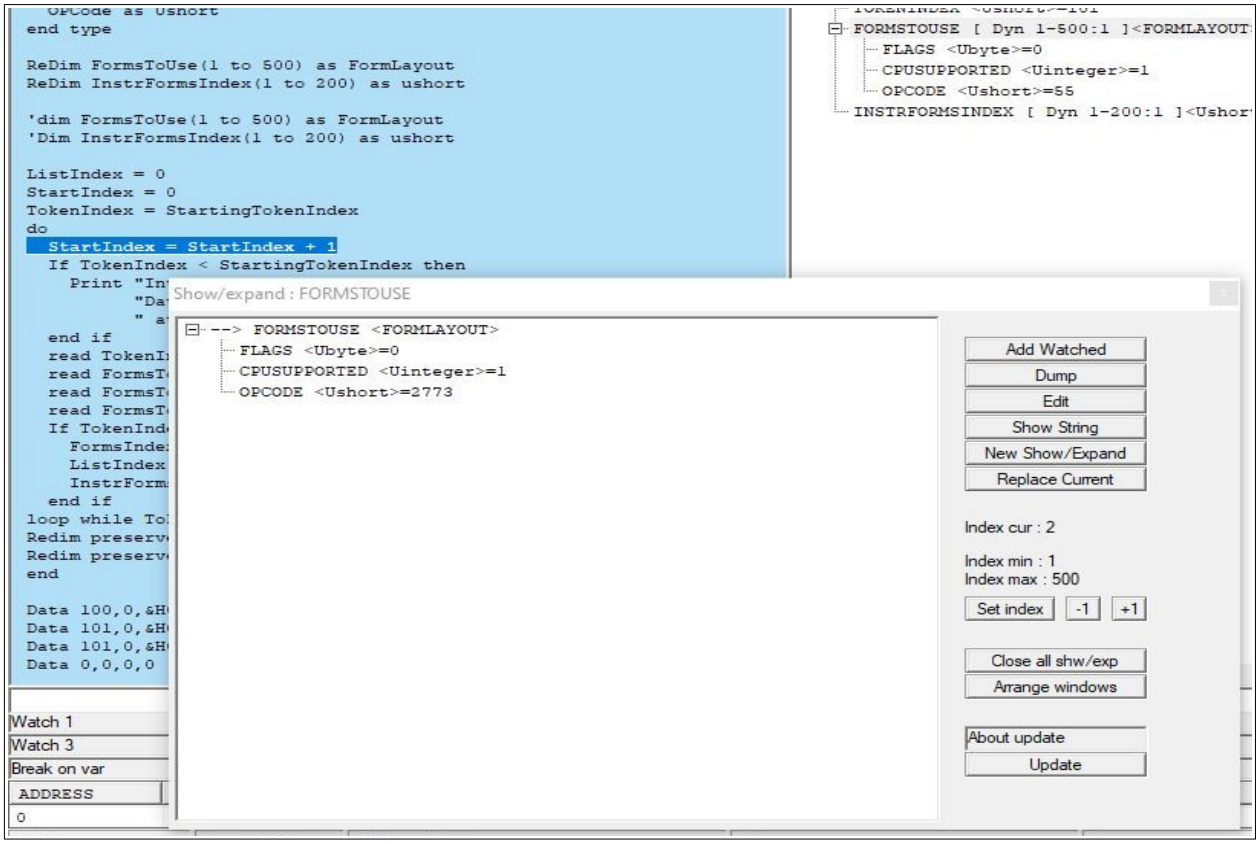


- Possible tests on value : change, equality, difference, etc.
- To set : contextual menu on proc/var breakpoint on var or in memory box.
- To delete : just left click on the line box.
- a message is displayed when a break on var occurs for better visibility and allowing to keep or remove it.

[Summary]

Show/expand

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



- Up to 20 windows displaying the data pointed if pointer and/or expanding if udt.

- Arrays inside structure are handled.
- There are the same possibilities than with the variables (add watched, dump, edit..)
- New expand : open a new window with the selected element
- Replace current : just replace the current root with the selected element, without opening a new window
- 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)

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

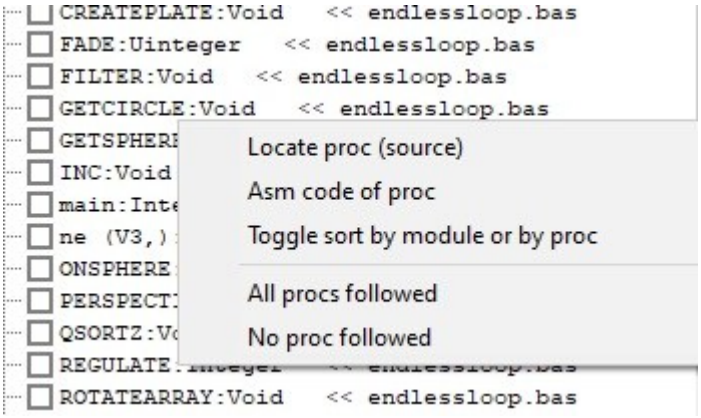
- When a dynamic array is redimmed use 'Update ' button. A message box is displayed for warning (full update or only indexes) , the min/max indexes are updated and the current is set in the new range.
Don't use +-1/set index before 'Update' as the address is not yet updated…..
Not done if the array is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

-To close/to arrange all the expand windows
- About update : the update is not automatic but done by clicking the update button. Be careful the data could have been deleted

[Summary]

Procedures

Displays all the procs (running or not) with file source and address (verbose mode). For operator Fbdebugger recreates the full names.

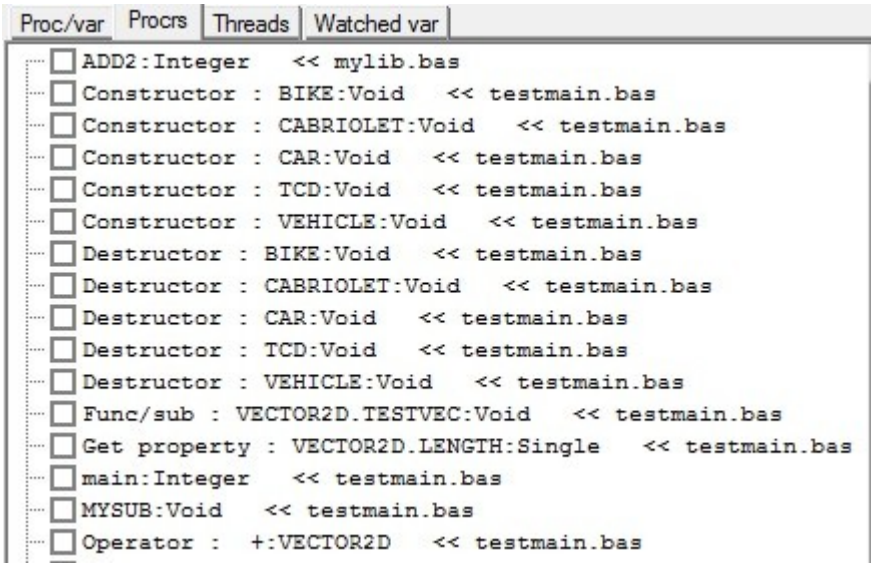


- Check one for disabling it : the proc is excluded from debugging. BE CAREFULL.

Right click to display the 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.
- dissassembly of proc with insert of each basic codeLine

- Proc call chaining to follow in one page
and Proc backtracking to follow in both direction the proc calls.: 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 except if the -gcc option is used.

[Summary]

Calling/called procedure chain

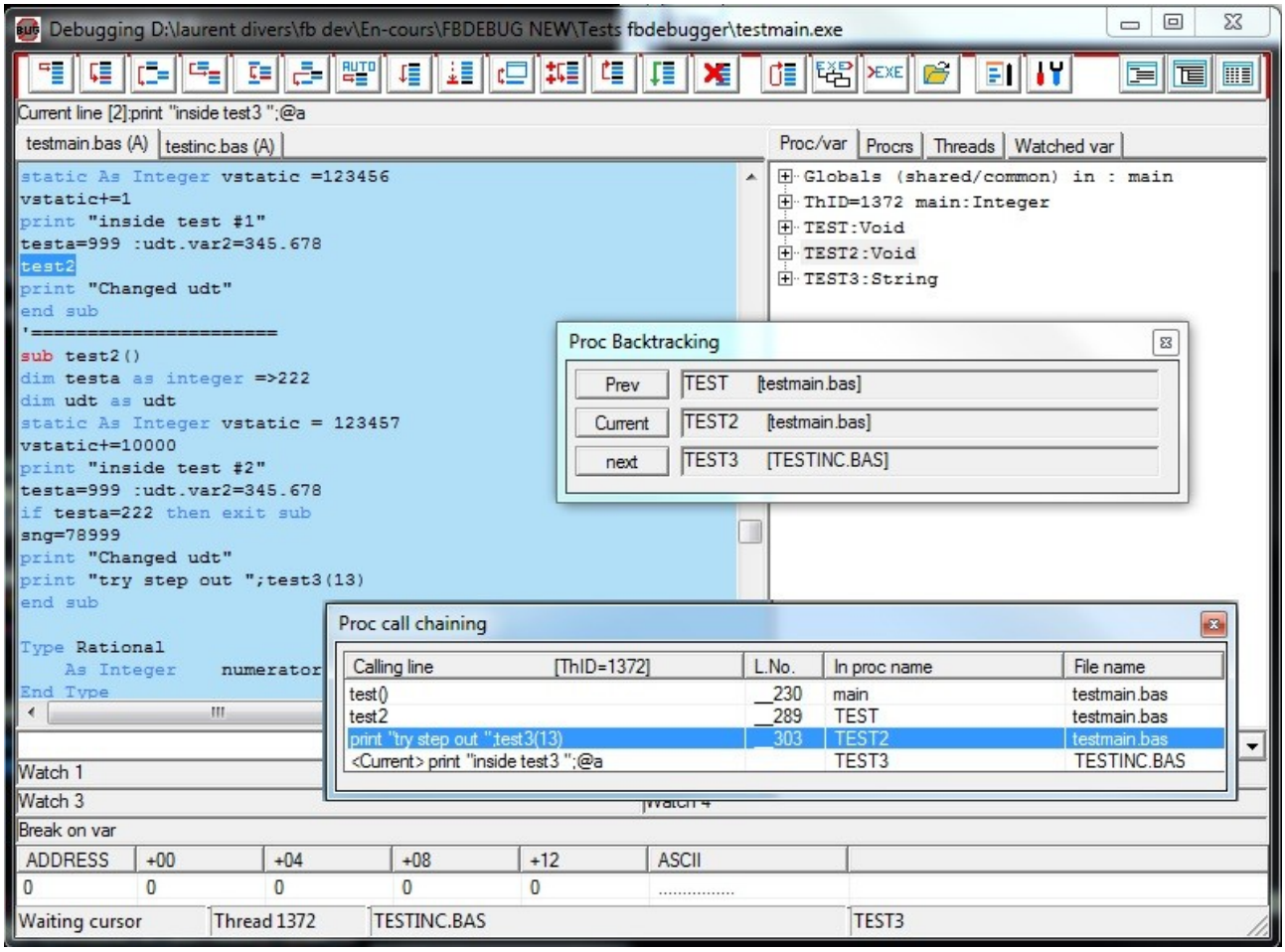
There are 3 ways to see the chain of all the calling/called procs.

1/ In thread tab

2/ Backtracking (see context menu in proc/var tab and thread tab) : only the previous, current and next procs are listed.
Click on a button (Prev/Current or Next) to navigate in the chain of procs.

3/ 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.

Only one option could be used at the same time and the dialog box is canceled by any execution (step, run, etc) .

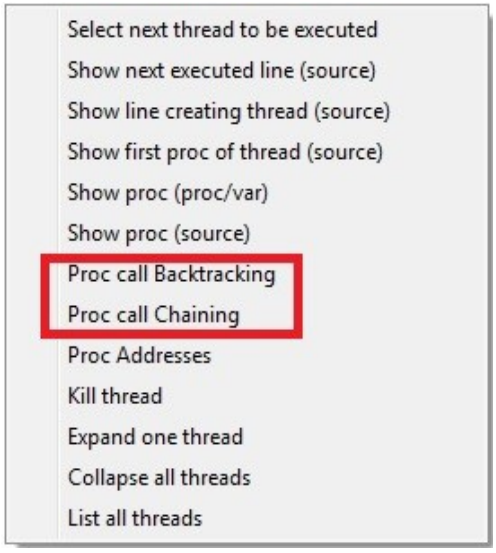


[Summary]

Threads

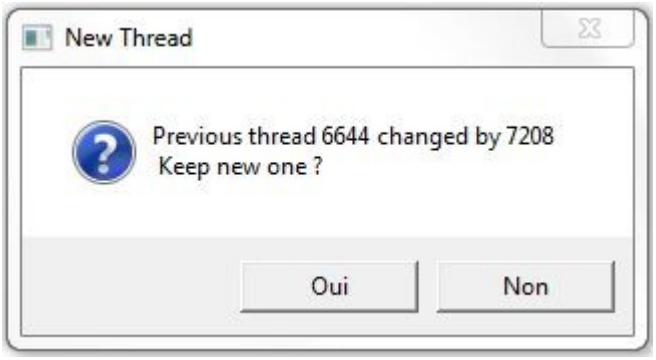
Displays all the threads (10 max) with ID and in verbose mode the handle (settings) then shifted every proc attached to the thread (call stack).
- The current thread is indicated by : (next execution).
- In case of automatic change of thread (mutex or threadcreate) a message is displayed giving the choice 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)

- Right click to display the 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 displays only the threads.
- List all threads : displays all the threads of the process included the threads not debugged.

- When a new thread is started a box is displayed allowing to keep the new one or to return to the older.



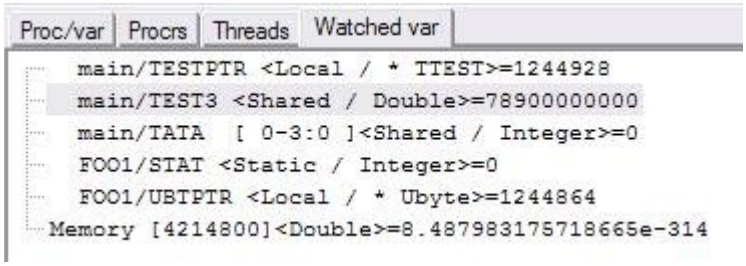
[Summary]

Watched variables/memory

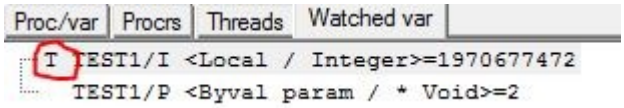
Watched variables are an easy way to keep an eye on some specific variables or part of memory.
Up to 20 watched values (variables/memory), updated also when auto mode

On the main window only the 4 first are displayed, removed by right clicking on.

All are displayed in the watched tab.



- Alt+left click on (or near) a variable in the source codes window to add it in the watched list. Also option in proc/var.
- For memory use the manage dump dialog.
- Variables or memory could be traced in the log file 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.

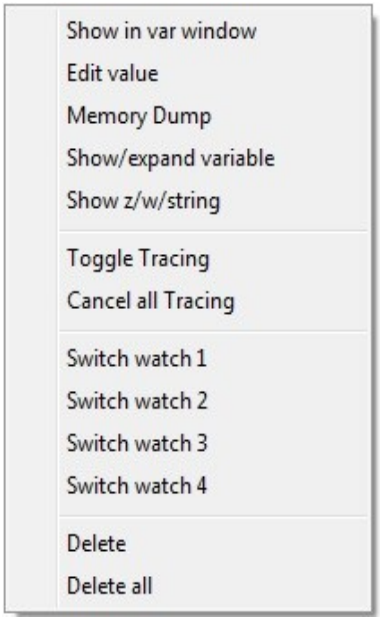


- Different formats (hex, binary,..) for watched vars, select twice or more to get a new format
- Local var watched are kept between proc calls but are displayed as non-existent and can just be deleted.
- If you rerun immediately the debuggee without closing Fbdebugger, all the watched are kept. In case 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.

[Summary]

- Right click to display the context menu

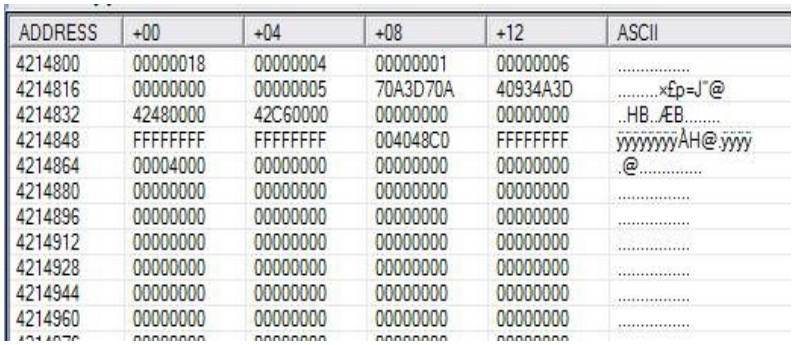


- Show in var window: selects the variable in the proc/var
- There are the same possibilities that with the variables (dump, edit..)
- Toggle tracing : set/reset the tracing mode in the log file.
- Cancel all tracing : the tracing is suppressed for every watched
- Switch watch x : switch the selected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

[Summary]

Memory Dump

- Displays a part of memory (16 bytes or more) with auto update (step or auto stepping)

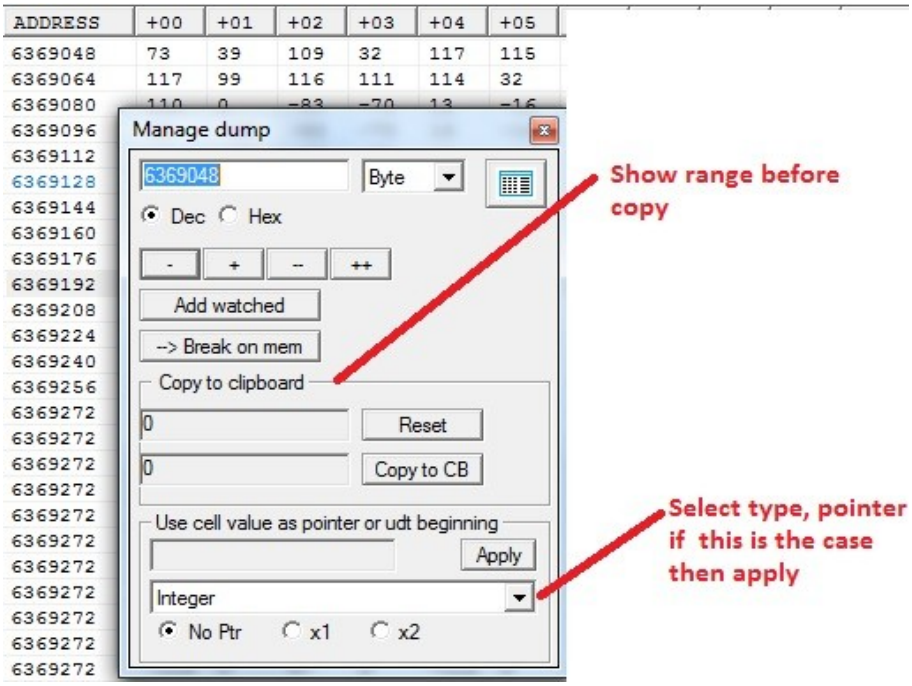


ADDRESS	+00	+04	+08	+12	ASCII
4214800	00000018	00000004	00000001	00000006	.....
4214816	00000000	00000005	70A3D70A	40934A3D	.....×£p=J"@
4214832	42480000	42C60000	00000000	00000000	..HB..ÆB.....
4214848	FFFFFFFF	FFFFFFFF	004048C0	FFFFFFFF	ÿÿÿÿÿÿÿÿ.@HÀ.ÿÿÿÿ
4214864	00004000	00000000	00000000	00000000	.@.....
4214880	00000000	00000000	00000000	00000000	.....
4214896	00000000	00000000	00000000	00000000	.....
4214912	00000000	00000000	00000000	00000000	.....
4214928	00000000	00000000	00000000	00000000	.....
4214944	00000000	00000000	00000000	00000000	.....
4214960	00000000	00000000	00000000	00000000	.....

- Formated (byte,ubyte,integer,etc/DEC,HEXA) according to datatype
- Beginning on address of variable or free address
- Click on a column header to change a value (only first row)
- 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.

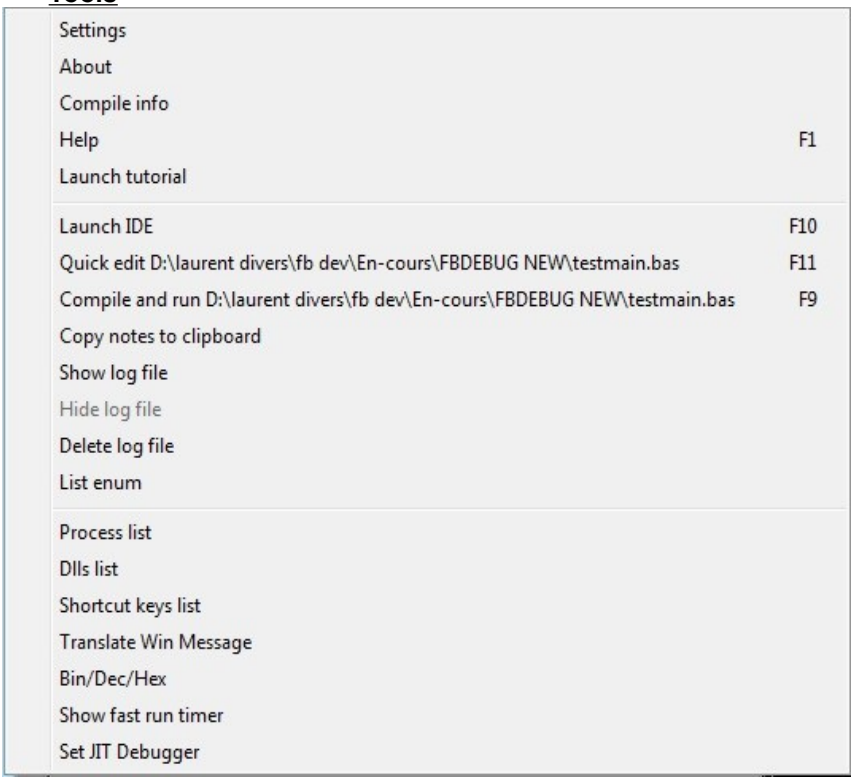
- to change the value of any cell in dump memory : left click on it
- to copy dump into clipboard : shift+left click on one cell / on header for whole page / address area or ascii area for one line or multilines

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.



[Summary]

Tools



Settings : to set parameters for Fddebugger.

About : version number and author (Laurent GRAS)
Feel free to send me (debug AT aliceadsl DOT fr) bugs, remarks, suggestions and improvement requests
And an option to launch Internet explorer with the dedicated page on the freebasic's forum
Compile info: displays informations about FBC used for compiling the debuggee.

Help: displays help, hotkey=F1

Launch IDE : launch an ide with the bas file as parameter if its path and file name are filled [settings]

Quick edit : uses an edit box to do some change
Option to kill running pgm when use of quick edit
2 options in quick edit : [save,compile and run] if only one source file / [reload]

Compile and run : compiles and starts the bas file

Copy notes to clipboard : notes

Show log file : shows the log file as a code source
Hide log : supress (close) the log in the source codes tab.
Delete log file : deletes the log file

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.
Shortcut keys list : just a list of all shortcut keys.

Translate win message : Enter the number of a Windows message to get the corresponding text.

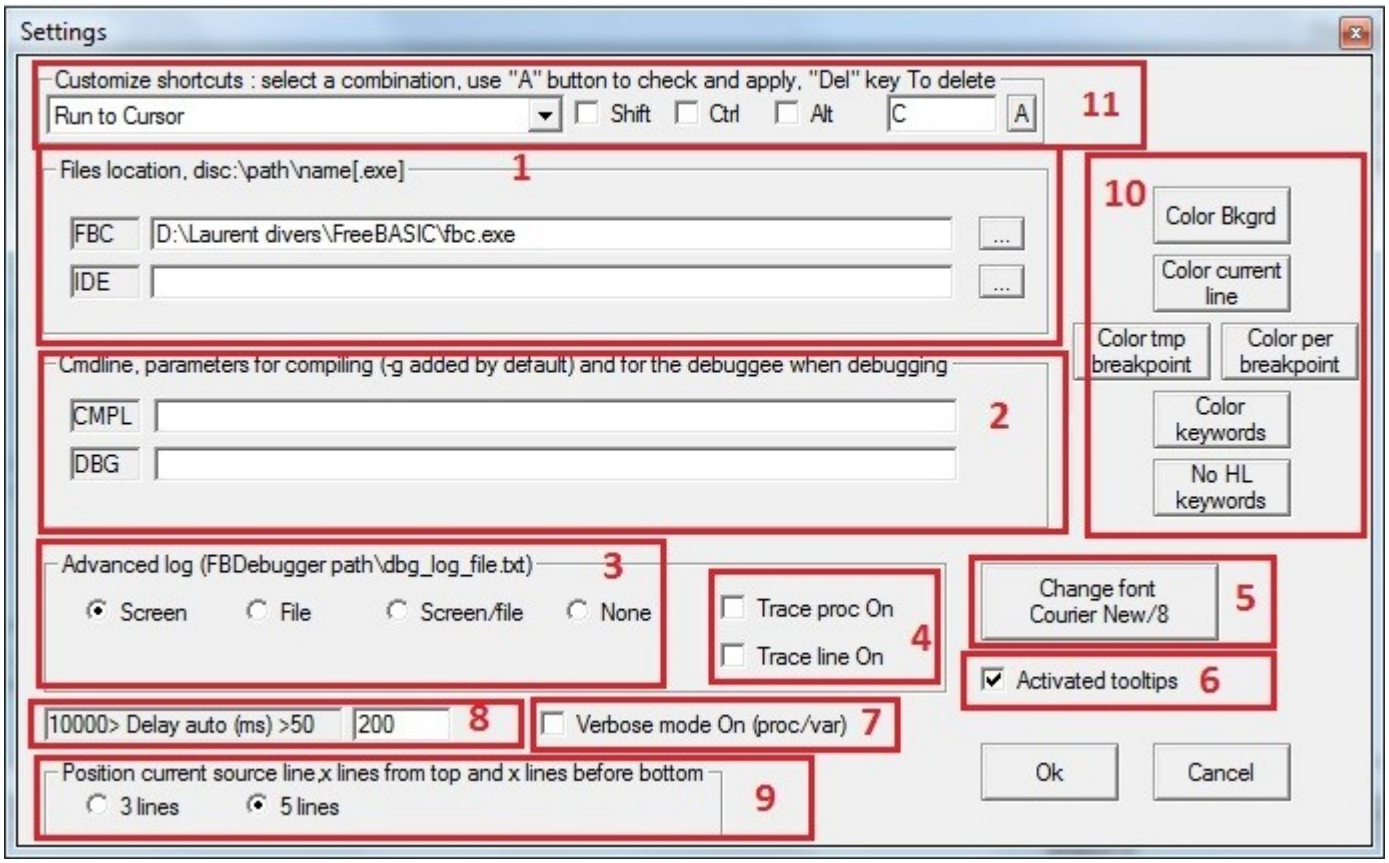
Bin/hex/dec : enter a decimal or hexa (&hxxx) value to get the value in dec/hexa/bin.

Show run fast timer : displays the elapsed time when using fast run.

JITdebugger : set the Just in time debugger

[Summary]

Settings

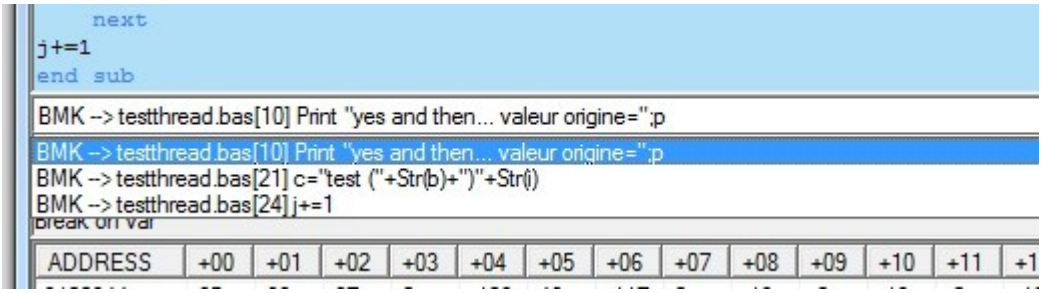


Option in the tools button.
Warning some parameters are directly applied no need to click on the OK "button". Most of settings are saved in the ini file.

1. Enter path and file names (fbc.exe, IDE)
2. Enter parameters for compiling and debugging
3. Activate different use of log (screen display, file) default is none. Saved in the ini file.
4. Trace of executed procs and lines
5. Change the font size (8,10,12)
6. Desactivate the tooltips
7. Toogle verbose mode on/off, to get address and advanced information
8. Select the delay between every line execution. Min (50) or to max (10000).
9. Select the position of displayed current line, 3 or 5 lines from top
10. Select the colors for all the elements. (current line, background, breakpoint, keyword)
 and also define if keywords are **H**igh**L**ighted (not and blue by default)
11. Customize existing shortcut key or create new one for every menu option. List of existing shortcut keys.
 Click on the "A" button to check and apply. If the combination is already used a message box is displayed to indicate the option in case.

[Summary]

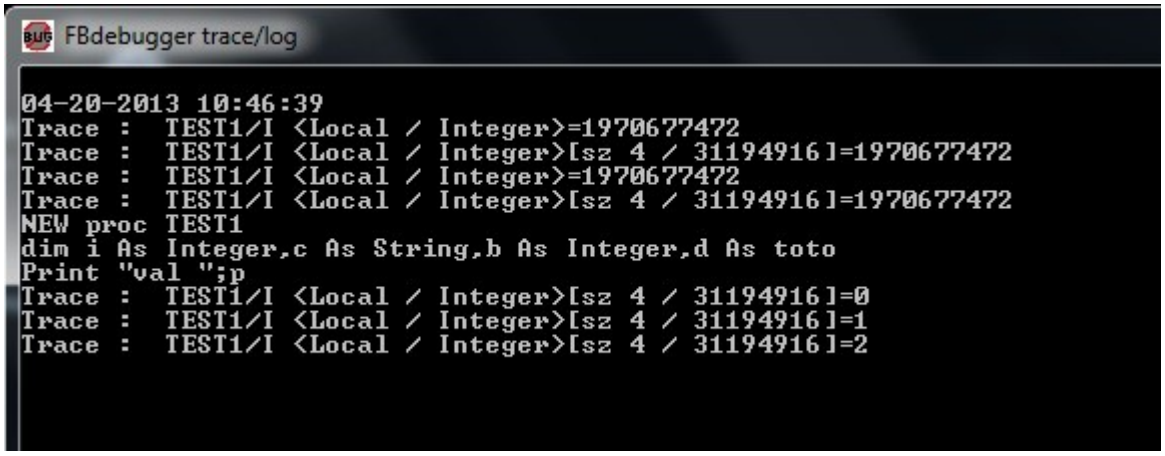
Bookmarks



- Up To 10 bookmarks
- Put cursor on line, right click and select 'bookmark' or ctrl+f2 to toogle only in the current file.
- Reached by F2 to Next and shift+F2 to previous. When beggining or end of file are reached a dialog box offers to restart the search.
- There is also a possibility to select by a combo box showing all the bookmarks (all files)
<file name>[Line number]beginning of the line

Log file

Some informations about the execution (system, procs and lines, procs and vars, watched var) could be displayed on a console window or stored in a file or both.[settings].



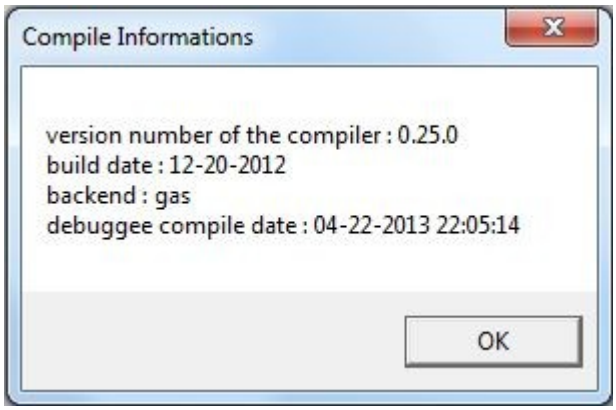
- Displays log file as code so find text is possible. Remove by hide log in tools.
- New data is appended so use delete log to reset the file.

[Summary]

ADVANCED FEATURES

Compile info

To displays the version and the build date of FBC, the used backend and the debuggee compilation date.



Just add these lines anywhere in the source :

```
dim shared FDBG_COMPIL_INFO As String
FDBG_COMPIL_INFO="$$__COMPILINFO__$
$"+__FB_VERSION__+"/"+__FB_BUILD_DATE__+"/"+__FB_BACKEND__+"/"+__DATE__+" "+__TIME__
```

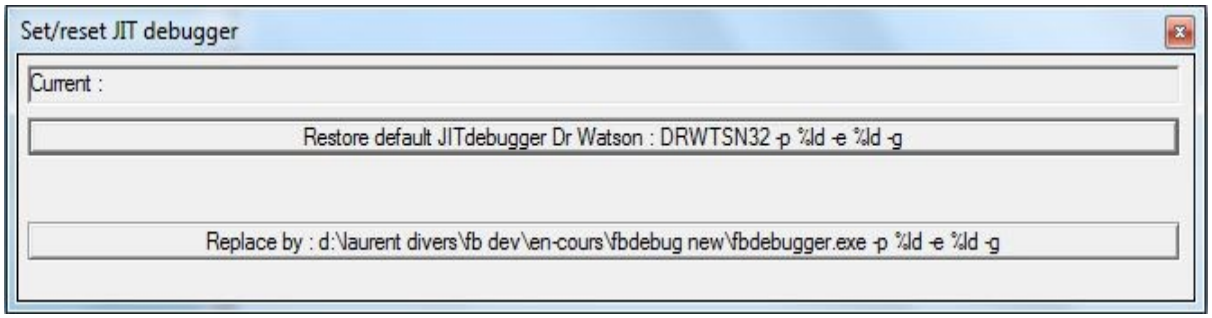
Fbdebugger seeks in the memory (data section) the "$$_COMP..." string and shows the informations in a message box, option in tools.

Change execution

Sometime it's interesting to skip or rerun a part code. This command or button changes the next executed line..
The change could be done only if the selected line is inside a same proc. That is to avoid problem with the stack. However use it carefully.. To do that a CPU register is changed.

JIT Just in time debugger

This option allows to define Fbdebugger to be automatically launched when the debugbreak instruction is met or an access violation (or an exception) is occuring, obviously your program has to be compiled with the -g option.



Click on the replace button to replace the default debugger (Dr Watson) by Fbdebugger in the registry. Restore is possible.

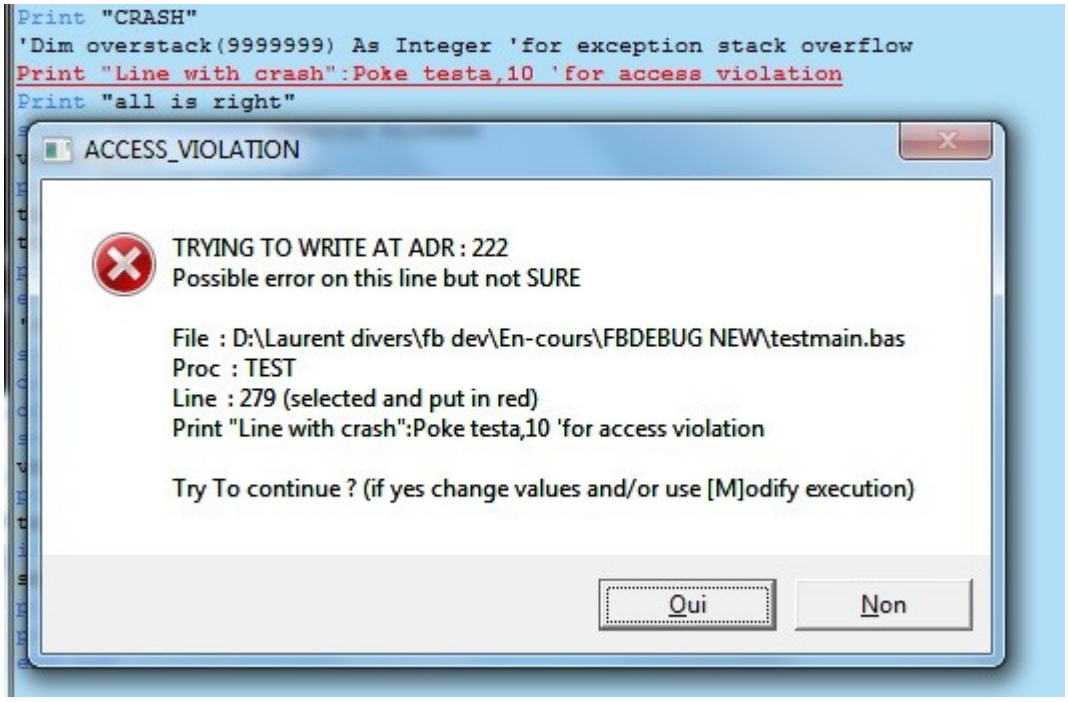
[Summary]

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.

Example :
- Compile this piece of code.

```
Dim As Integer a(1)
Print "test"
a(5000000)=1 'access violation
Print "end"
```

- If Fbdebugger is not defined as JIT debugger, do it, start it then select/start the example exe. You get this box.



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 change exec button.

Use of debugbreak

When the instruction debugbreak in encountered the JIT debugger is started.

Example :

- If not done define Fbdebugger as jit debugger.
- Compile this piece of code

```
#include once "windows.bi" 'mandatory for debugbreak
Print "test line 1"
Debugbreak
Print "test line 3"
```

- Don't start Fbdebugger but just your program.
  The first print is done then Fbdebugger is automatically launched and waits on the second print.

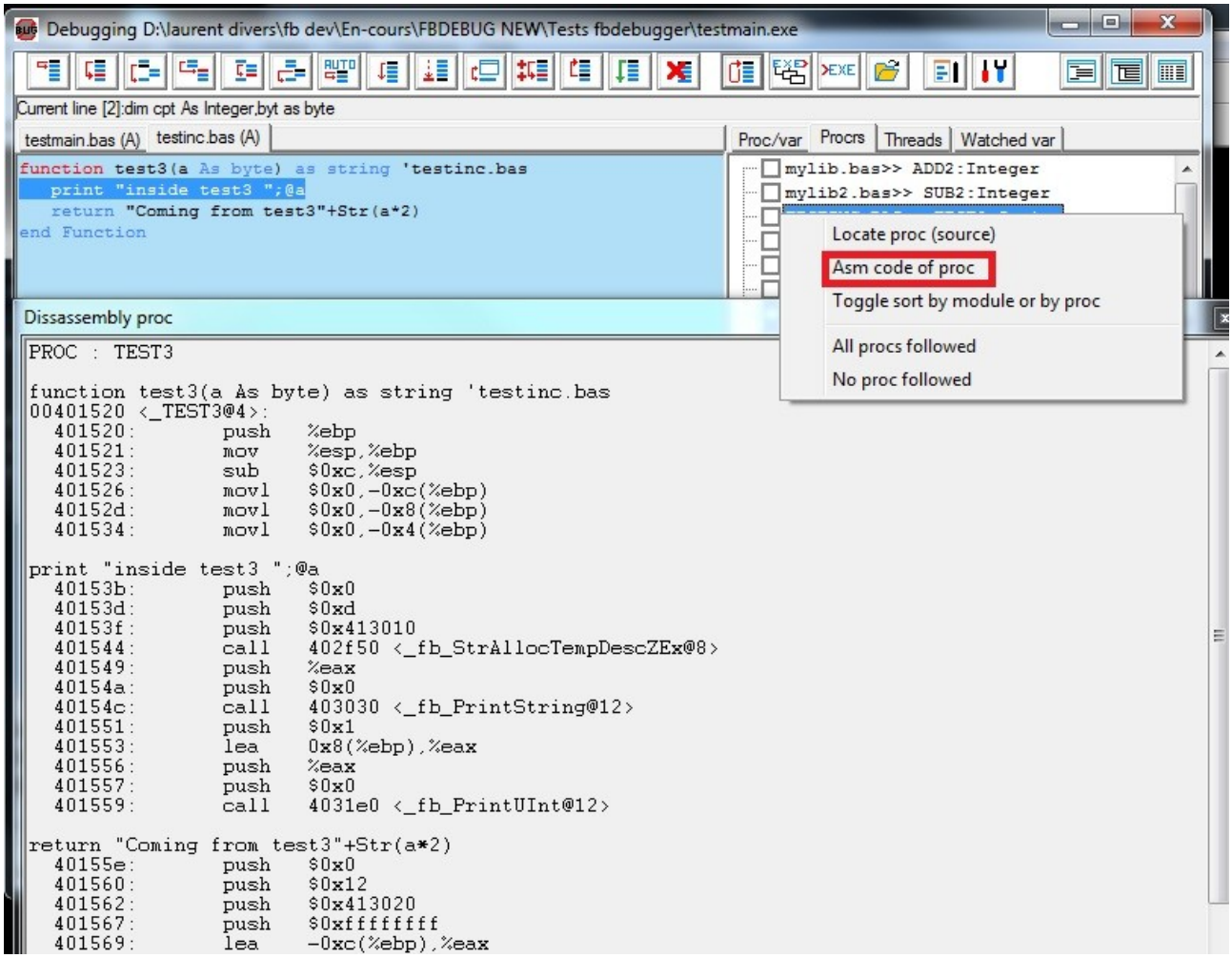
[Summary]

Dissassembly

The dissassembly could be done for one line or for an entire procedure. Objdump is used so necessay to be installed with Fbdebugger.

1/ In source code context menu (line or proc)
2/ In proc tab (complete proc)

Only the executable basic lines are inserted

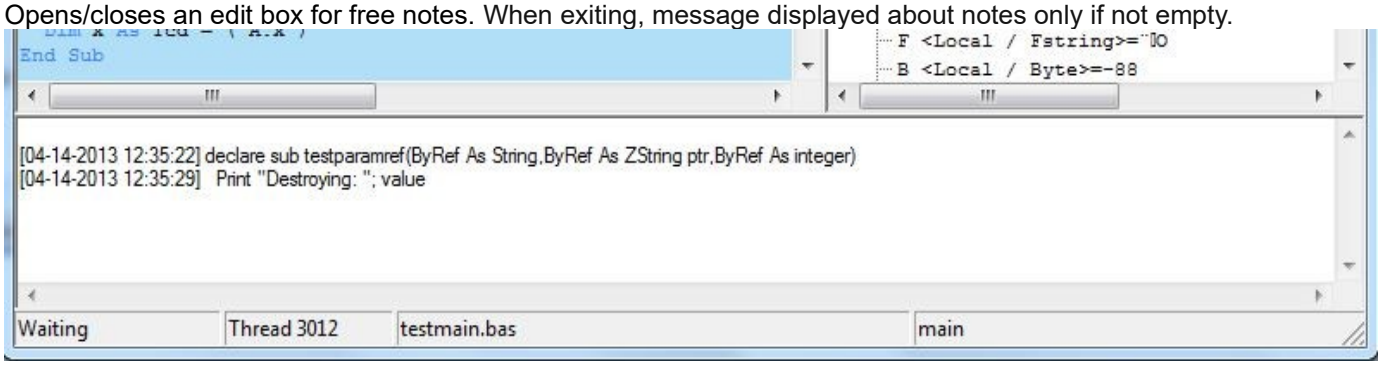


[Summary]

MISCELLANOUS

Notes

Opens/closes an edit box for free notes. When exiting, message displayed about notes only if not empty.



- Put the cursor on a line, right click and select add note to insert automatically this line with the date/time
- After eventually add comments.
- As usually ctrl+A / ctrl+C to copy it to the clipboard or with the copy notes to clipboard in the tools menu.

Used Files

Permanent files
- Fbdebugger.exe, fbdebugger.ini, Fbdebugger_help.chm

Temporary files in the same directory than the exe
- fbdebug_compil.log : compil log
- dbg_log_file.txt : trace file

Registry

No change in registry except when Fbdebugger is defined as JIT debugger

Objdump

- Objdump (tools) is necessary as Dwarf debug data is very crpytic, not easy to decode….
- When compiling with gas32 or gas64 or gcc32bit with gstabs no need.
- However is also used when dissassembling some parts of the executable.

- When objdump.exe tool is needed (compilation with gcc or for dessassembling) and not found (in the Fbdebugger's folder.) a message is displayed.

Provided 32bit / 64bit in the zip file but you can also grab them from the net
e.g tdm-gcc.tdragon.net/download and select binutils

GCC

- Due to the fact that C compilation changes things and some informations are missing using the option -gcc is possible but debugging should be strange.
- Naked procs are not correctly managed, they are not included in data debugging.
- The array defined (A to B) are internally changed by the compiler to zero based arrays eg 4 to 7 becomes 0 to 3.
In case of shared fixed-length there is no information in debugging data for retrieving the initial definition. So problems should happen.

- Lines from include files and not in a proc are not correctly associated with their file : they are tagged with the main file.
- If the version of gcc is >3 add this parameter "-Wc -gstabs+" in the FBC command-line. Otherwise the debug datas are not added.

- Tip : Debug with the -gas option then optimize with -gcc

[Summary]