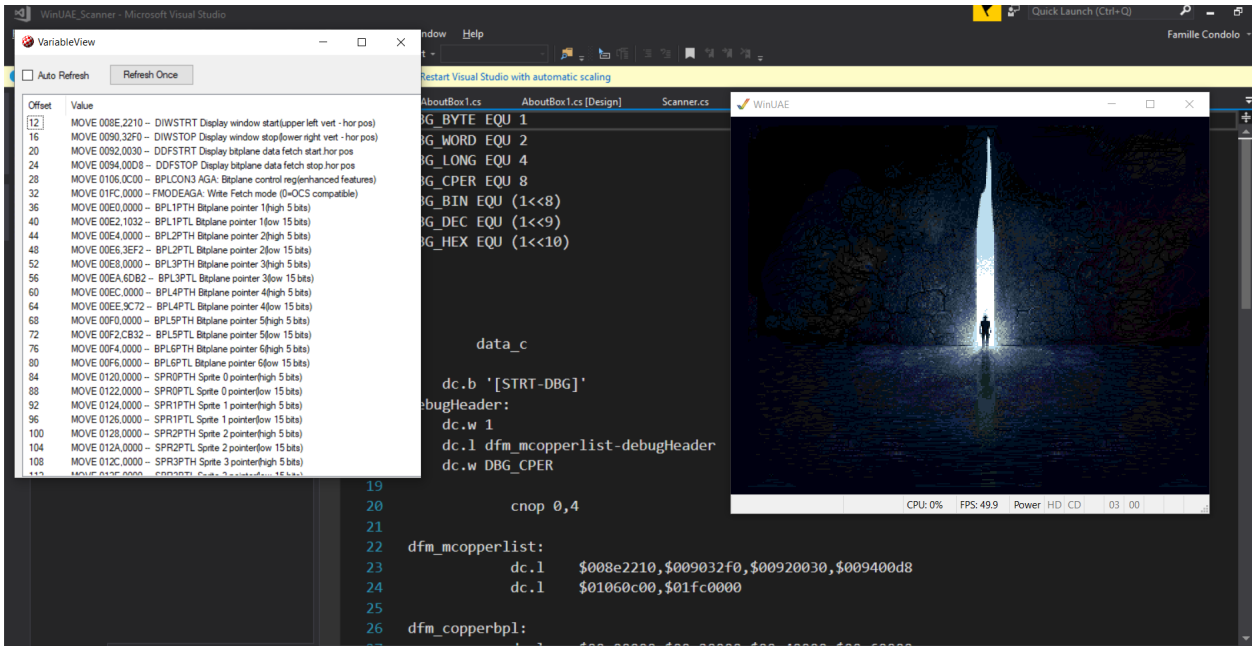
WinUAEScanner

V1.2 - Soundy

What is it?

WinUAE scanner is a memory watch. It helps following your variables to debug code when using WinUAE. The trick consists in reading the memory pages from WinUAE's application process and searching for data to watch. This is a very simple hack, yet helped me to quickly spot errors. Note it should work with other emulators than WinUAE.

You can access the GitHub project here:
https://github.com/fcondolo/WinUAEScanner



Amiga code side

We're going to describe how this example works:

```
DBG_BYTE    EQU 1
DBG_WORD    EQU 2
DBG_LONG    EQU 4
DBG_CPER    EQU 8

DBG_BIN     EQU (1<<8)
DBG_DEC     EQU (1<<9)
DBG_HEX     EQU (1<<10)

dc.b            '[STRT-DBG]'

debugHeader:
            dc.w    3
            dc.l    counterl-debugHeader
            dc.w    DBG_LONG|DBG_HEX
            dc.l    counterw-debugHeader
            dc.w    DBG_WORD|DBG_DEC
            dc.l    counterb-debugHeader
            dc.w    DBG_BYTE|DBG_BIN

counterl:   dc.l    $DEADBEEF
counterw:   dc.w    1000
counterb:   dc.b    10

dc.b            '[END-DBG] '
```

First, set some constants:

```
DBG_BYTE    EQU 1
DBG_WORD    EQU 2
DBG_LONG    EQU 4
DBG_CPER    EQU 8
```

The above constants define the supported “watchable” types (byte, word, long word, and copperlist).

```
DBG_BIN     EQU (1<<8)
DBG_DEC     EQU (1<<9)
DBG_HEX     EQU (1<<10)
```

The above constants define how watched variables should be displayed (binary, decimal, hexadecimal). Note that copperlists have a dedicated display mode.

Then, declare what you want to watch:

```
 dc.b            '[STRT-DBG]'
```

The above tag is what WinUAEScanner will look for in WinUAE’s memory. It defines the beginning of the watch info block.

```
debugHeader:
            dc.w    3
            dc.l counterl-debugHeader
            dc.w  DBG_LONG|DBG_HEX
            dc.l counterw-debugHeader
            dc.w  DBG_WORD|DBG_DEC
            dc.l counterb-debugHeader
            dc.w  DBG_BYTE|DBG_BIN
```

The above lines are an example of watch info block, or “debug header”. debugHeader MUST be right after he '[STRT-DBG]' tag in memory, and is structured as follows:

- dc.w Number of variables to watch

  Then, for each variable:

- dc.l offset (byte offset between the variable and the beginning of the debugHeader)
- dc.w type | display preference

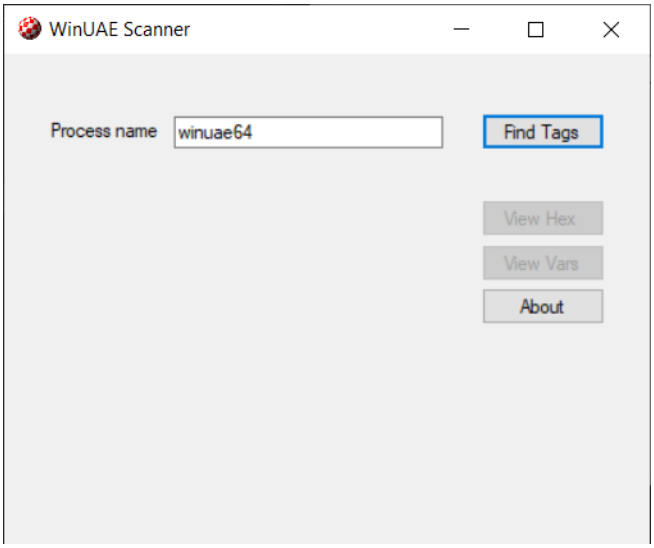
Finally, close the watch:

Just use `dc.b '[END-DBG ]'`. Beware, there is a space before the closing bracket in '[END-DBG]', just to make it 10 chars, an even number.

Remember: Only the memory block between `'[STRT-DBG]'and` '[END-DBG]' can be watched. Means all the variables you want to watch have to be within the same section, and the '[END-DBG]' tag must be placed at the end of the data of the last variable you want to watch. E.g. if you want to watch a copperlist, the '[END-DBG]' tag must be placed after the last byte of your copperlist.

Note that all the data between `[STRT-DBG]` and `[END-DBG ]` will be dumped in the hex view. You don't need to preference everything in the debugHeader. The debugHeader is just here to give a more user friendly formatting than pure hex dump for the variables you like.

The main window

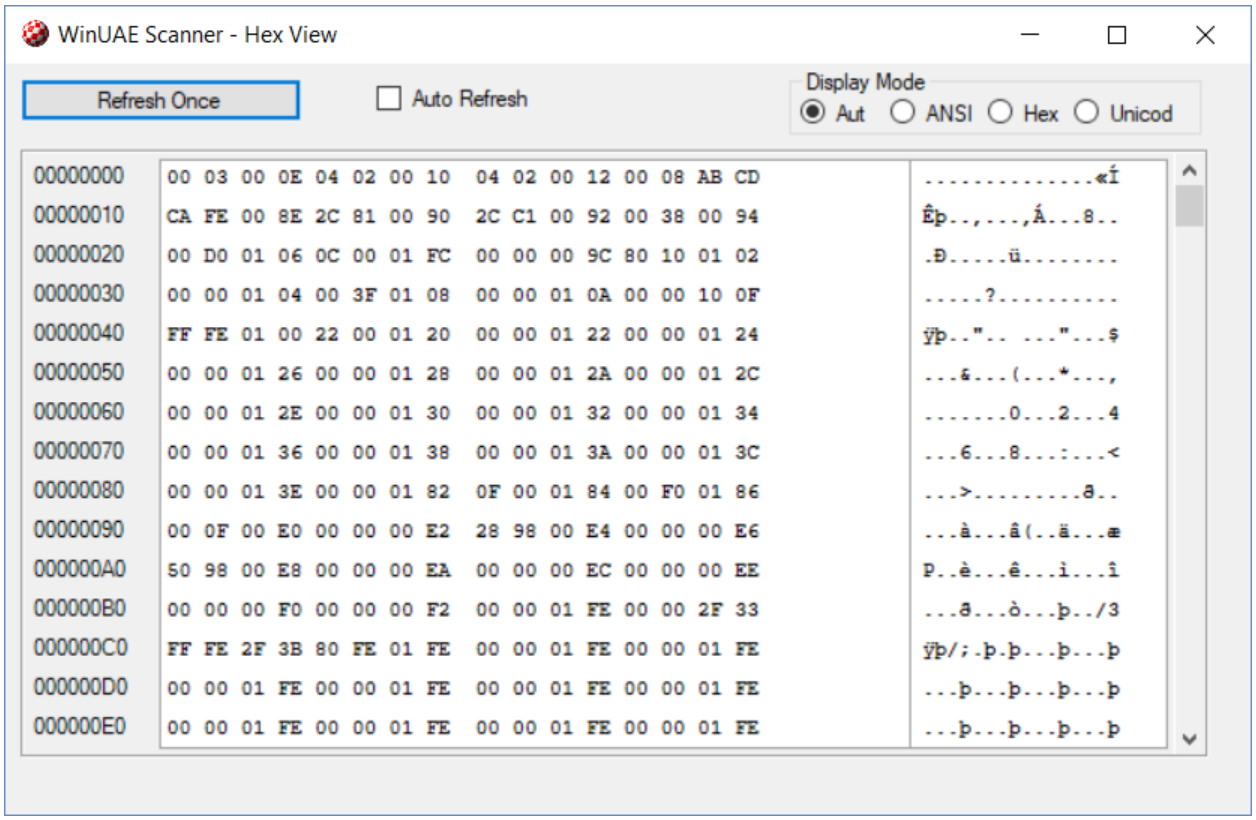


- **Process name:** At launch, the tool scans active processes, and finds the one that is the most likely to be WinUAE. If this field remains empty, please type WinUAE's process name here.
- **Find Tags:** When this button is clicked, the tool will scan WinUAE's memory and look for the '[STRT-DBG]' and '[END-DBG]' tags. All the memory between these two tags will be copied and processed by the tool.

Whenever tags are found, the “View Hex” and “View Vars” buttons will activate.

“View Hex” will show a hex dump of all the data between '[STRT-DBG]' and '[END-DBG]', while “View Vars” will show the data described by the debugHeader in your Amiga code.

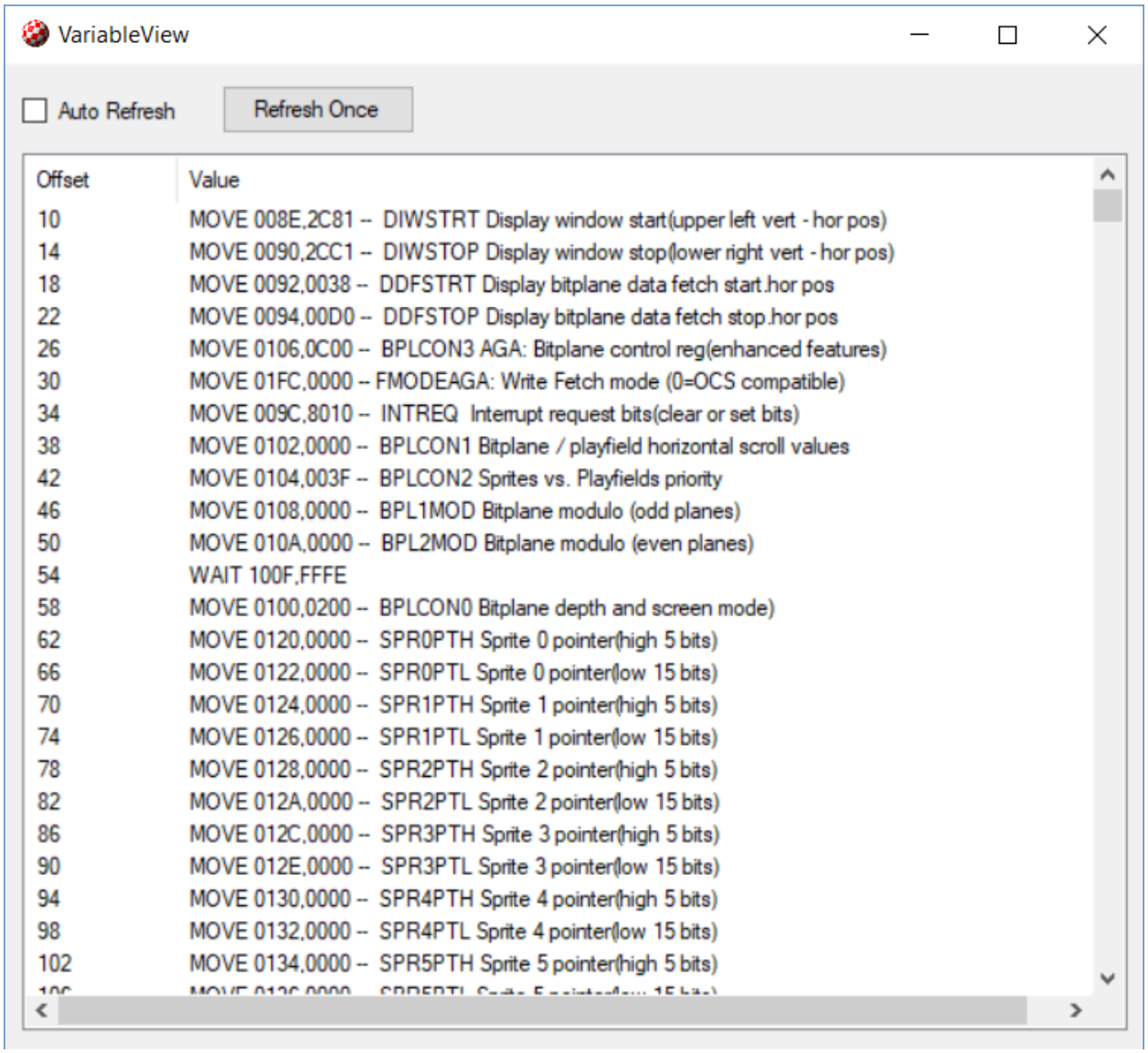
The hex view



Well, not much to say, this is a hex view 😉

- Click "Refresh Once" to refresh memory contents
- Click "Auto Refresh" to constantly refresh memory contents

The variables view



Displays the memory dump respecting the "debugHeader" structure. The hardware registers accessible by the copper are automatically documented (if you used `DBG_CPER`).

Support

For feedback, bugs, features, whatever, ping Frederic Condolo on Facebook or LinkedIn.