

maXbox



maXbox Starter 33

Start the Oscilloscope

1.1 Measure Time and Sound

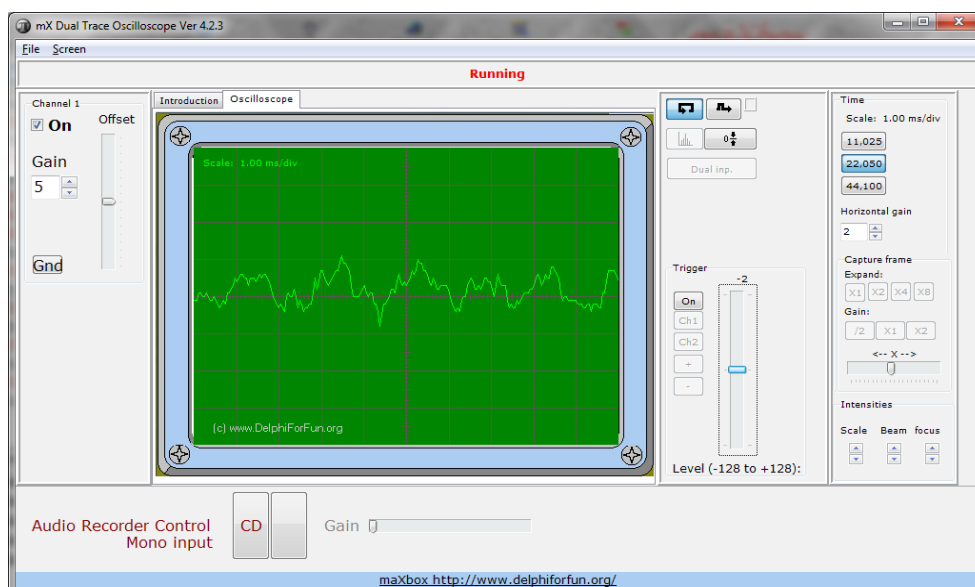
Today we step through the use of an virtual Oscilloscope and his visual representation in time and spectrum.

Oscilloscopes are one of the must of an electronic lab and are essential for anyone designing electronics, troubleshooting, or working with high speed electronics. Oscilloscopes are one of the few pieces of electronic equipment that plays multiple roles and can be used in the place of other electronics equipment. In maXbox we use a virtual one.

To picture a sound for e.g. you have to set a song first:

```
playMP3(mp3path);  
closeMP3;
```

In menu /Options/Oscilloscope v4 you start the device:



👉 Oscilloscopes include a rich set of math and measurement capabilities that are often underutilized. Knowing how to effectively use these capabilities can enable you to solve complex test problems quicker and better.

You can now start the scan with the button on the right side of the oscilloscope on the image above marked in blue light.

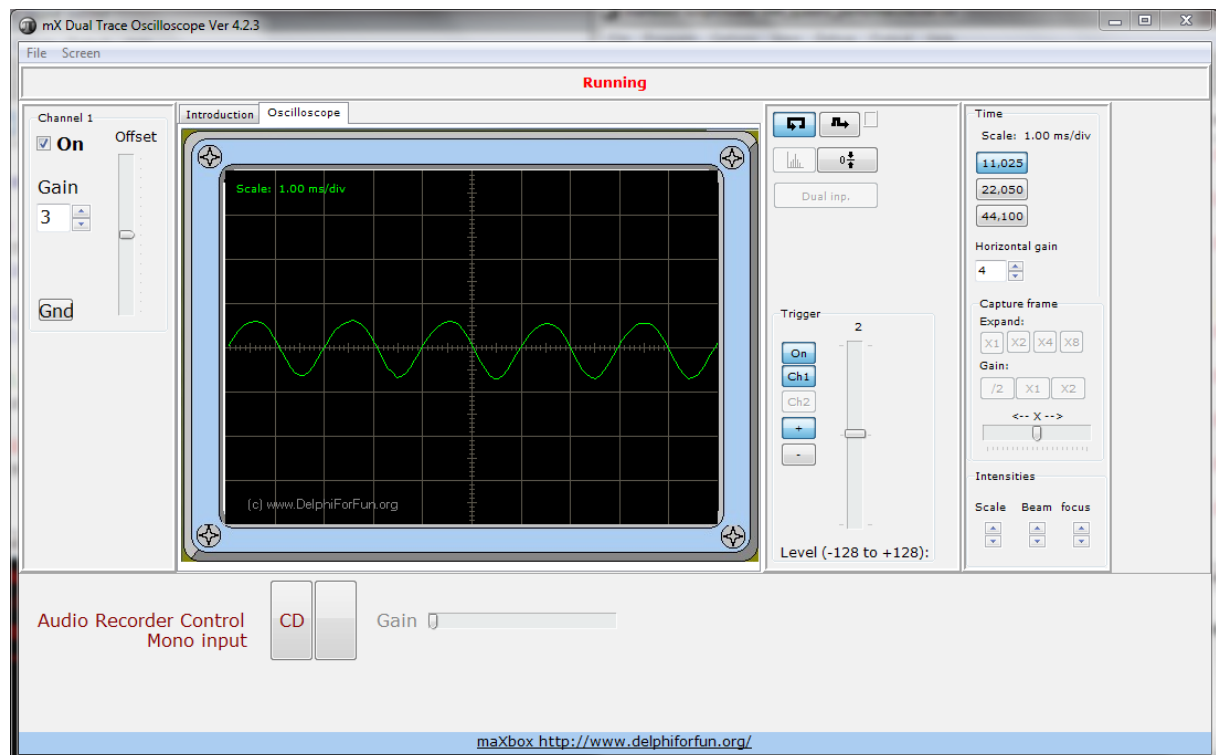
A trick is to capture the sound with the inbuilt microphone so the device can connect it by initialisation.

Next we measure a audio frequency of 500 Hz means 500 waves/signals per second or sec. (Hz = unit of frequency)

The second (symbol: s; abbreviation: sec.) is the name of a unit of time, and is the International System of Units (SI) base unit of time. Subdivisions of the second can be indicated by adding SI prefixes to second.

```
Tone (500, 10000);
```

With Tone() you can produce a sound of 500 Hz with a length of 10 secs.



You see the screen notion of scale: 1.00 ms/div. I must admit that needs an explanation.

The „Time/Div.“ caption determines, how long an electronic beam (or wave) which draws the curve needs for moving from the left to the right edge of a division. The button Horizontal gain controls the „time scale“.

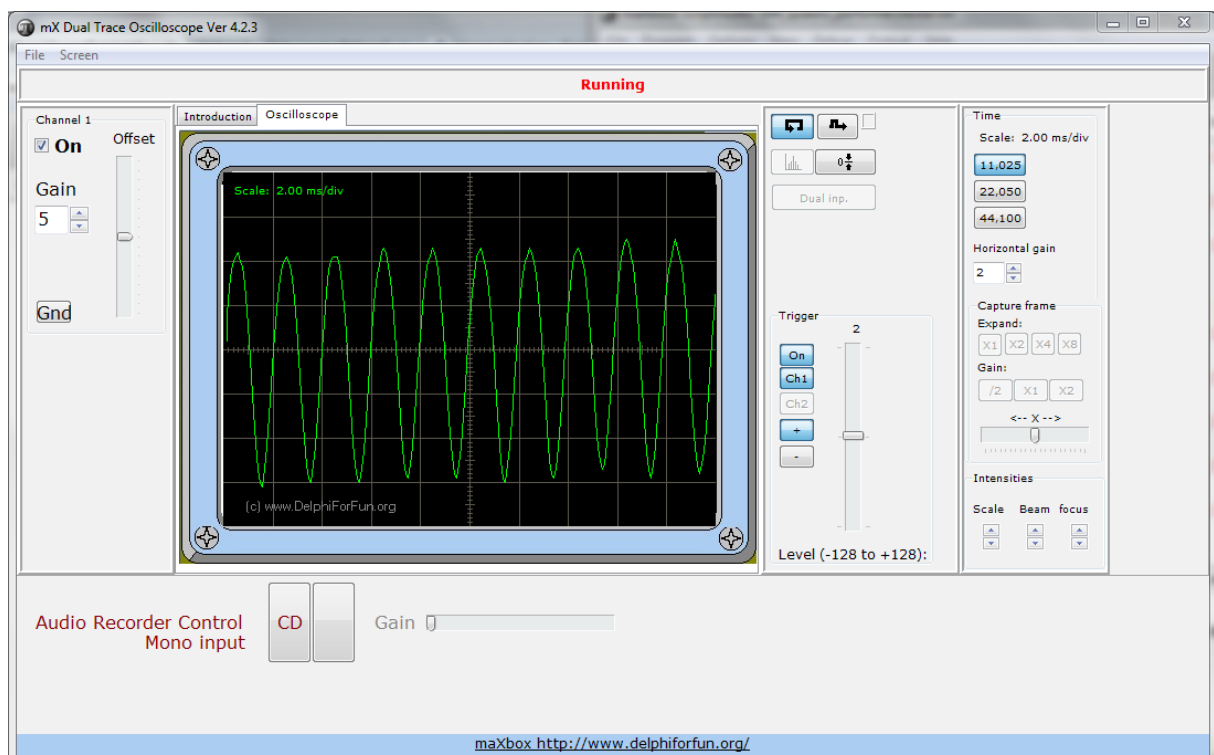
👉 A „division“ or div is one square in milliseconds on the screen. If our scale is set to „1 ms“ for example, it means that 1 ms is needed per square for the horizontal movement, i.e. 10 ms for moving the beam from the left to the right edge of the screen with 5 waves. ~

Therefore the surface is divided into 10 divisions of 1 wave per 2 ms.

Hope you know the equation $f=1/D$ means 1 over Duration and the horizontal frequency in that example would be $1/2 \text{ ms} = 500 \text{ Hz}$, this means the electronic beam moves 500 times a second from the left to the right edge of the screen.

`maxcalcf('1/0.002')` ---> 500 Hz [2ms = 0.002 sec.]

The observer sees an image that seems to be continuous, non-flickering; now we change the scale to 2.00 ms/div with the same frequency:



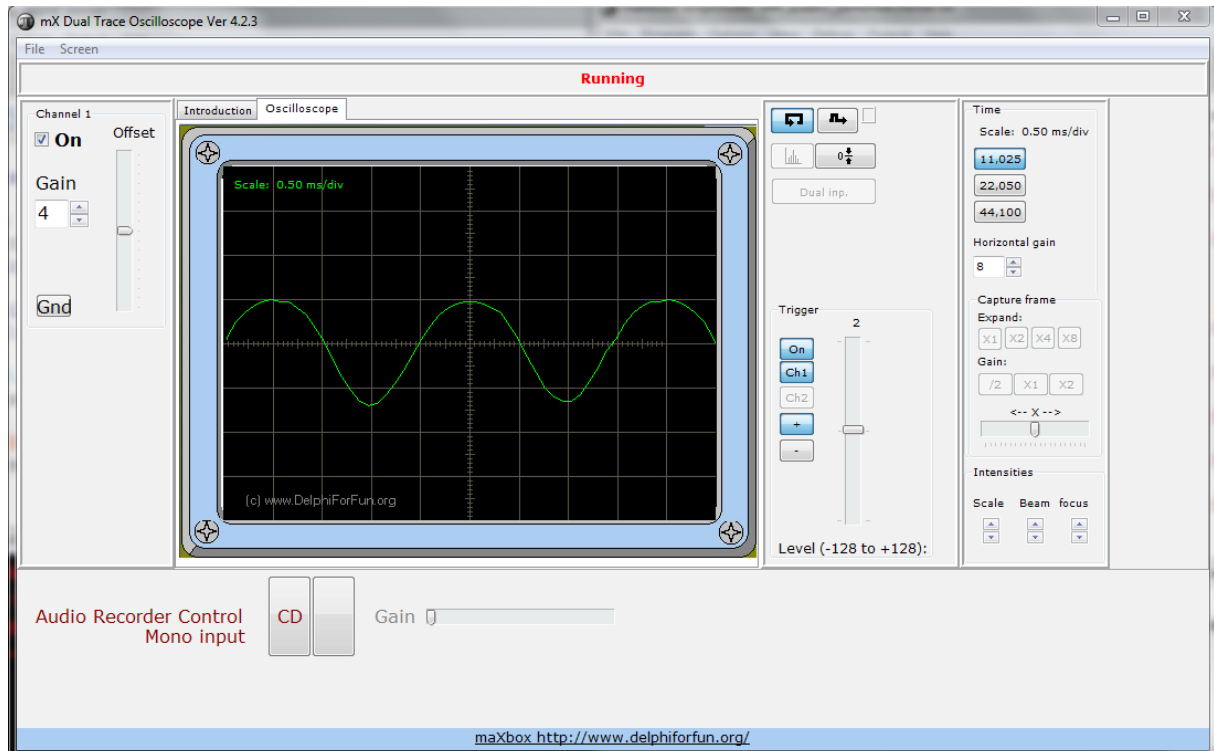
Now you see 10 waves in 20 ms and one second has $50 * 20 \text{ ms}$ that results in $50 * 10$ waves are the sum of 500 waves or 500 Hz! This is the same frequency as before but we can see more waves as the div is now 2 ms instead of 1 ms.

A bit weird but its also possible to measure such a wave of 2 ms:

```
with TStopWatch.Create() do begin
  try
    Start;
    //TimeOutThisFunction()
    Sleep(2); //2ms = 0.002 secs
  Stop;
finally
  Writeln('watchTime: ' + floatToStr(GetValueMSec/1000));
  Writeln('watchTime: ' + GetTimeString);
```

Free; //TStopWatch Class
end;

Next we shake the time back to a half of a ms that is only 0.50 ms/div, so what you expect to see on the screen, less or more waves?:



You see 2 ½ waves ~ on the whole screen in 5 ms that results in
 $200 \text{ ms} * 2.5 = 500 \text{ Hz}$. ($1000\text{ms}/5\text{ms} * 2.5$)

Time has long been a major subject of philosophy, art, poetry and science. Many fields use an operational definition in which the units of time are defined. Scholars disagree on whether time itself can be measured or is itself part of the measuring system¹.

But we can capture such a wave with a trigger!

The oscilloscope trigger facility is one of the key functions within the oscilloscope.

Triggering enables the scope, whether digital or analogue, to display a steady image on the screen that can be viewed by the user / engineer investigating a particular circuit or signal.

👉 Without a trigger, or other form of synchronisation, it would not be possible to display a steady signal on the screen.

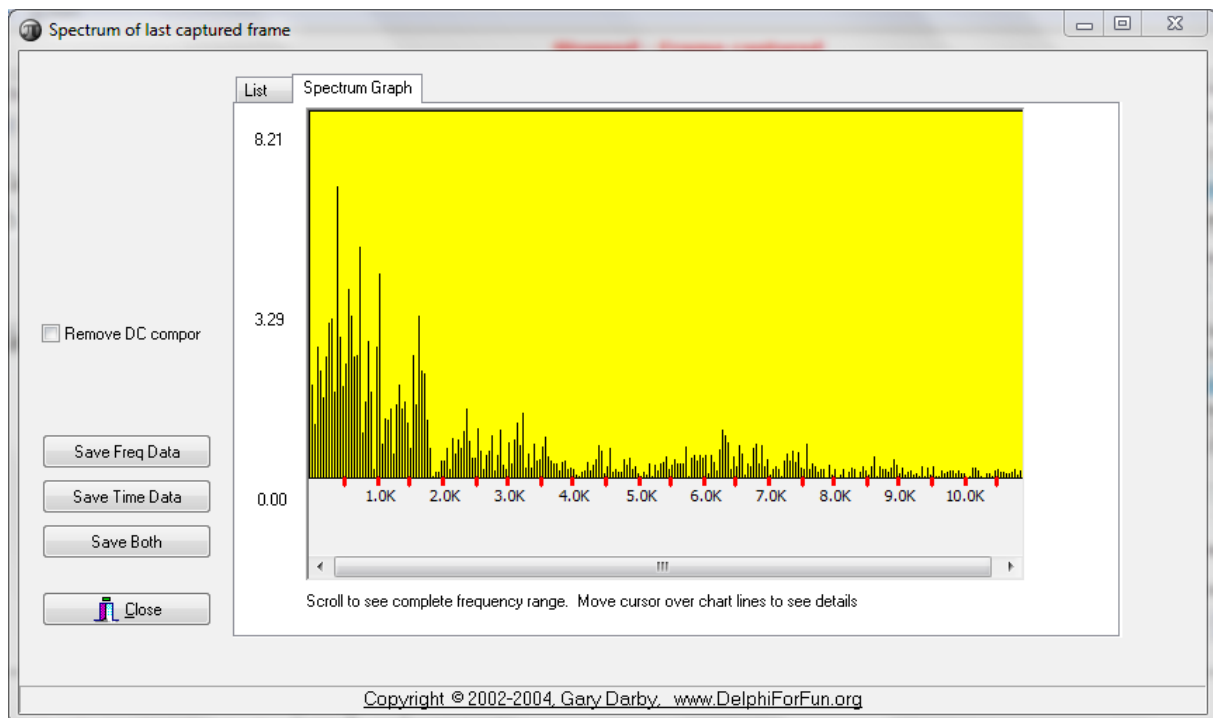
You can set a capture frame and a trigger level. As can be seen, the setting of both the trigger levels and frame determine the image that is seen on the screen. Careful adjustment of both controls ensures that the required portion of the waveform is seen.

¹ A clock shows just the position of the sun – we name that time!

Also a dual trace function using stereo input signals is on board.

- "Trigger" capability has been added.
Each scan is triggered when a signal rises above (+) or below (-) the preset trigger.
- To improve the image capture of transient events, there is now a "Capture Single Frame" button.
Use the "Trigger" feature to control when the frame will be captured.
- A "Set Zero Level" button will center the display vertically on the screen.
One of my sound cards has a "DC offset" large enough to noticeably move the display downward.

The voltage signals that an oscilloscope measures are generically called waves, which is any repeating pattern (sound waves, oceans waves, etc), and the signal displayed on the oscilloscope is called a waveform. Those waveforms can be split up in a frequency spectrum too:



Viewing the waveform on an oscilloscope can be used to determine several things about the signal including:

- The voltage value of a signal
- The frequency of a signal (Spectrum Analyzer)
- The DC and AC components of a signal
- The noise in the signal
- A distortion in the signal
- How a signal changes over time

Click the „Spectrum“ button after a frame has been captured and see the amplitudes of the frequencies contained in the sample. Fast Fourier

Transform (**FFT**) code has been added to analyze the signal. Results are displayed in a spectrum bar graph. Captured data in both time and frequencies domains can be saved to a file for further analysis. Also added user control of standard sampling rates (11,025 or 22,050 or 44,100 samples per second). In maXbox you have these predefined functions/object to use the internals and build at runtime your extensions:

```
maXform1.Oscilloscope1Click(Self);

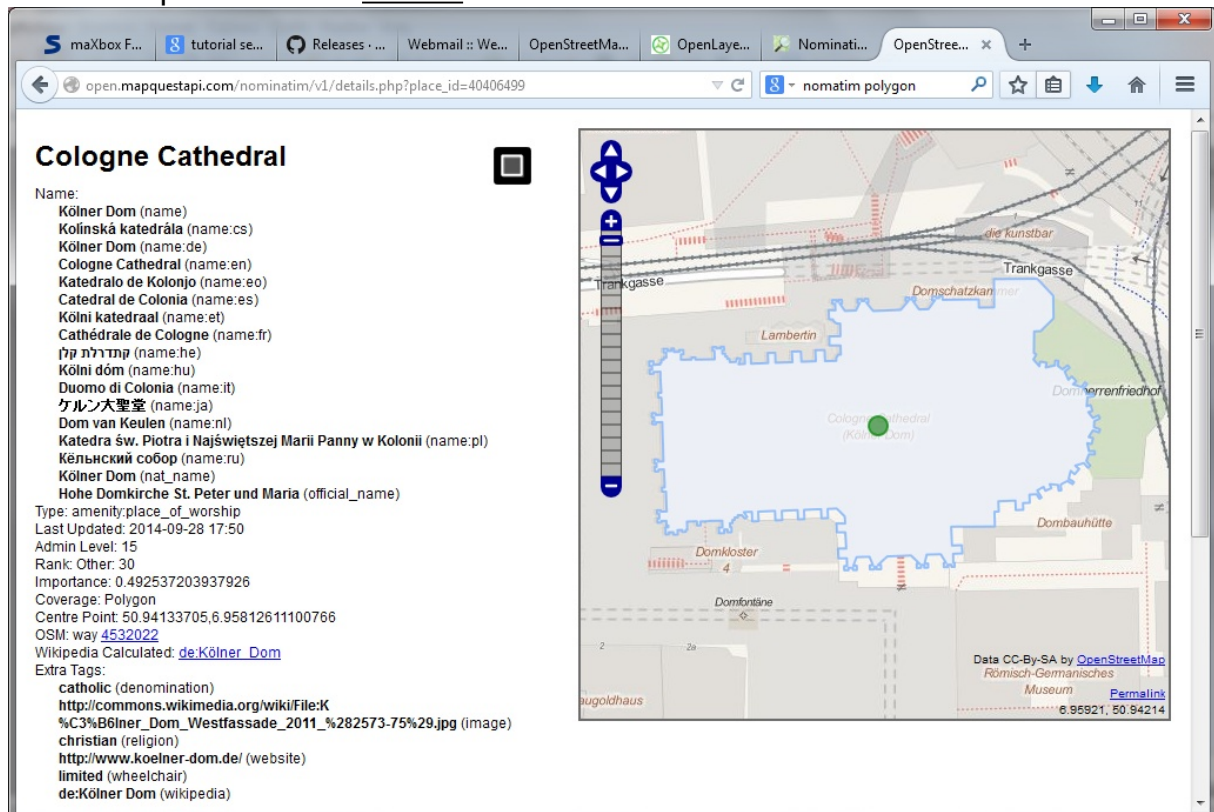
var oscfrmMain: TOscfrmMain;
oscfrmMain:= TOscfrmMain.Create(self);

procedure SIRegister_TOscfrmMain(CL: TPSPascalCompiler);
begin //with RegClassS(CL,'TForm','TOscfrmMain') do
  with CL.AddClassN(CL.FindClass('TForm'),'TOscfrmMain') do begin
```

With click 1 you call a GUI as found in menu Options\Oscilloscope V4
To define a sound to test you have several functions (among others):

1. MakeSound(440, 2000, 60,");
2. // MakeSound(440, 1000, 60,Exepath+'examples/kammertontest.wav');
3. Procedure MakeComplexSound();
4. Tone(440, 3000);

This picture below is just to say: An oscilloscope shows the micro world and a map shows the macro world!



1.1.1 Time Measure Code Behind

☝ Because Windows is multitasking and many processes and interrupts occur while your code is running, nanosecond timing would be not particularly useful. `GetTickCount` is also limited to the accuracy of the system timer (10 / 55 ms).

Also it is always possible that while that is the resolution, it is not the precision - the counter may not increment by one each time it increments. Finally, if a high-resolution performance counter exists on the system, you can use the `QueryPerformanceFrequency` Win API function to express the frequency, in counts per second. The value of the count is processor dependent!

`QueryPerformanceFrequency` returns the number of "ticks" per seconds - it is the amount that the counter, `QueryPerformanceCounter`, will increment in a second.

What about when you need to process millions of tree brunch leafs or generate 10 quadrillions (10^{16}) of unique random numbers? In such scenarios it is important that your code executes as quickly as possible. The calculation is almost self defining: cycles per second = cycles / secs, or, as reported, millions of cycles per secs = cycles / microseconds. The program sets priorities to high values before performing a calculation and restores them after to increase the accuracy of the timing.

Obviously the most important data required for this script to work is the time speed itself.

Test the script with **F9** / F2 or press Compile. So far now we'll open the time-sound example: `068_sound_oscilloscope.txt`

http://www.softwareschule.ch/examples/068_sound_oscilloscope.txt

```
if QueryPerformanceFrequency(Frequency) then
    QueryPerformanceCounter(Start1);
    //Sleep(2000); //2000 millisecs = 2 secs
    PerformanceDelayMS(2000000*10); //2000 micro secs = 2 milli secs
    //2000*1000 = 2 seconds

    QueryPerformanceCounter(Stop1);
    WriteLine('Delta: '+IntToStr(Stop1-Start1) + ' Freq: ' +
        IntToStr(Frequency) + ' ' +
        Format('%0.6f', [(Stop1-Start1)/Frequency])+ ' seconds')
```

Just note that you have to check the boolean return value from `QueryPerformanceFrequency()` which returns false if performance counters are not implemented in your hardware.

```
PrintF('Delta: %d Freq: %d of %0.6f micro secs!', [Stop1-Start1, Frequency,
    (Stop1-Start1)/Frequency ])
```

By calling this function at the beginning and end of a section of code, an application uses the counter as a high-resolution timer.

```
Time1:= Time;  
PerformanceDelayMS(2000000*10); //2000 micro secs = 2 milli secs  
PrintF('%d %s',[Trunc((Time-Time1)*24),  
FormatDateTime("'h runtime:" nn:ss:zzz',Time-Time1)])
```

Hope you did already work with the Starter 18ff on Arduino topics:
<http://sourceforge.net/p/maxbox/wikimax/main/>

Conclusion Oscilloscopes provide the most basic, yet critical infos about how a circuit is actually behaving. Oscilloscopes display a signal in a graphical form, showing how the voltage at a point in a circuit changes over time. The vertical axis on an oscilloscope displays a voltage of the signal and the x axis represents the time.

Feedback @ max@kleiner.com

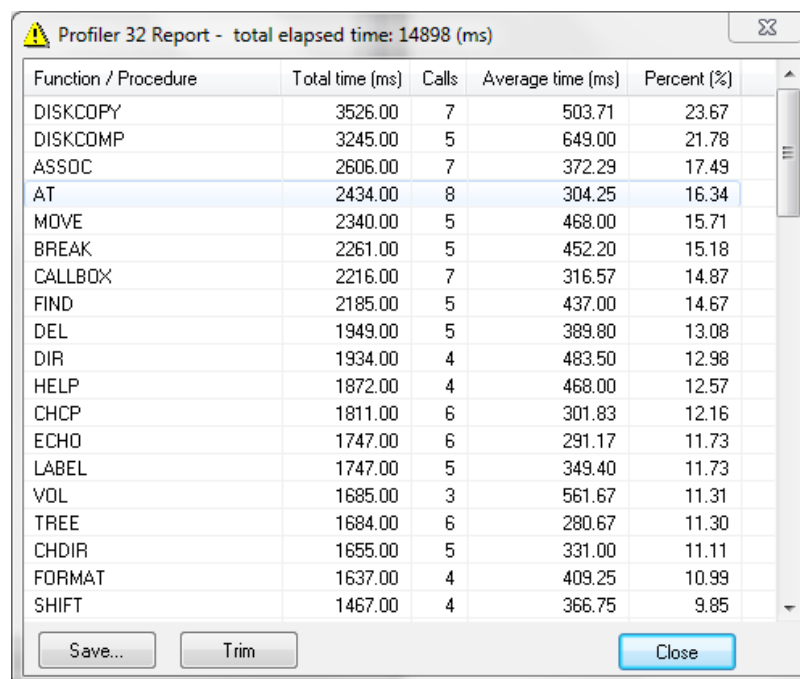
Literature: Kleiner et al., Patterns konkret, 2003, Software & Support

<http://www.virtual-oscilloscope.com/>

<http://www.delphiforfun.org/Programs/oscilloscope.htm>

http://www.delphiforfun.org/Programs/Delphi_Techniques/index.htm

1.2 Appendix Time Study with a Profiler



Profiler 32 Report - total elapsed time: 14898 (ms)

Function / Procedure	Total time (ms)	Calls	Average time (ms)	Percent (%)
DISKCOPY	3526.00	7	503.71	23.67
DISKCOMP	3245.00	5	649.00	21.78
ASSOC	2606.00	7	372.29	17.49
AT	2434.00	8	304.25	16.34
MOVE	2340.00	5	468.00	15.71
BREAK	2261.00	5	452.20	15.18
CALLBOX	2216.00	7	316.57	14.87
FIND	2185.00	5	437.00	14.67
DEL	1949.00	5	389.80	13.08
DIR	1934.00	4	483.50	12.98
HELP	1872.00	4	468.00	12.57
CHCP	1811.00	6	301.83	12.16
ECHO	1747.00	6	291.17	11.73
LABEL	1747.00	5	349.40	11.73
VOL	1685.00	3	561.67	11.31
TREE	1684.00	6	280.67	11.30
CHDIR	1655.00	5	331.00	11.11
FORMAT	1637.00	4	409.25	10.99
SHIFT	1467.00	4	366.75	9.85

Buttons: Save... Trim Close

Script Example: maxbox3\examples\070_pas_functionplotter_digital2size.txt