University of British Columbia
Electrical and Computer Engineering
Electrical and Biomedical Engineering Design Studio
ELEC291/ELEC292

Making a Simple Oscilloscope with the EFM8LB1 Microcontroller

Introduction

Quite often, having access to an oscilloscope is one of the most important factors in the development phase of an electronics project. Frequently though, an oscilloscope is not readily available and we basically 'flight blind' during the development and testing phases of our project. This document describes a simple oscilloscope that can be achieved with an EFM8LB1 microcontroller, a computer, and a Python script.

Features and Limitations

This simple oscilloscope design includes several features found in many commercially available products, including:

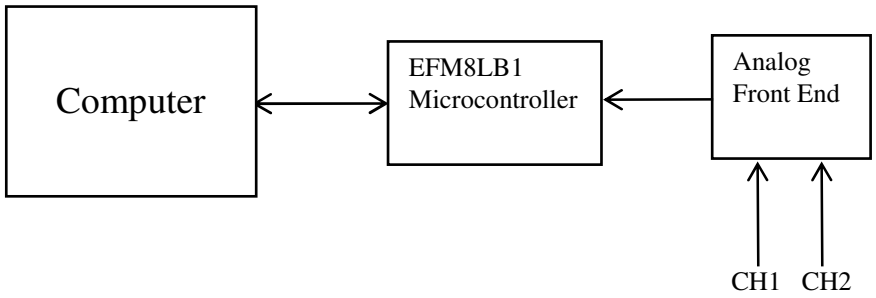
- 2 channels.
- Auto and Normal modes.
- Adjustable trigger: source, level, and edge.
- Python GUI, with cursors, visual trigger indication, and on-screen display of signal frequency.

Due to hardware limitations and in order to keep the design as simple as possible, this oscilloscope has these two important limitations:

- The sampling frequency is 500 kHz per channel. This limits the practical maximum frequency of the signals applied to the oscilloscope to about 50 kHz.
- No significant analog front end. This oscilloscope does not include either amplifiers or filters. The input voltage range is limited to 0.0V to 3.3V, which is the input range of the ADC in the EFM8LB1 microcontroller. This oscilloscope can not display negative signals.

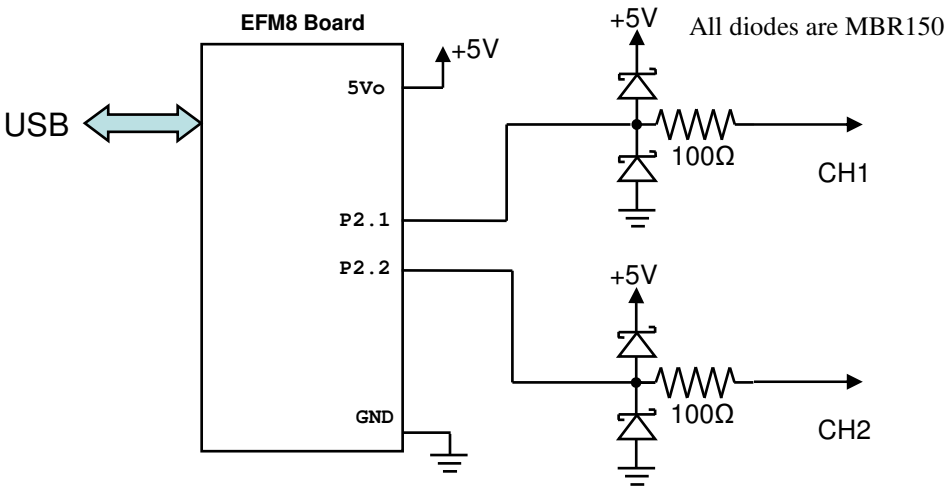
Block Diagram

The oscilloscope requires a computer with Python 3.x installed, an EFM8LB1 board with a USB to serial port, 4 Schottky diodes, and 2 x 100Ω resistors. The block diagram of the design is shown below.



Circuit Schematic

The analog front end of the oscilloscope is very simple; it consists of a voltage clipper built with Schottky[1] diodes and a resistor. There is a voltage clipper per channel. The circuit diagram is show in the figure below.

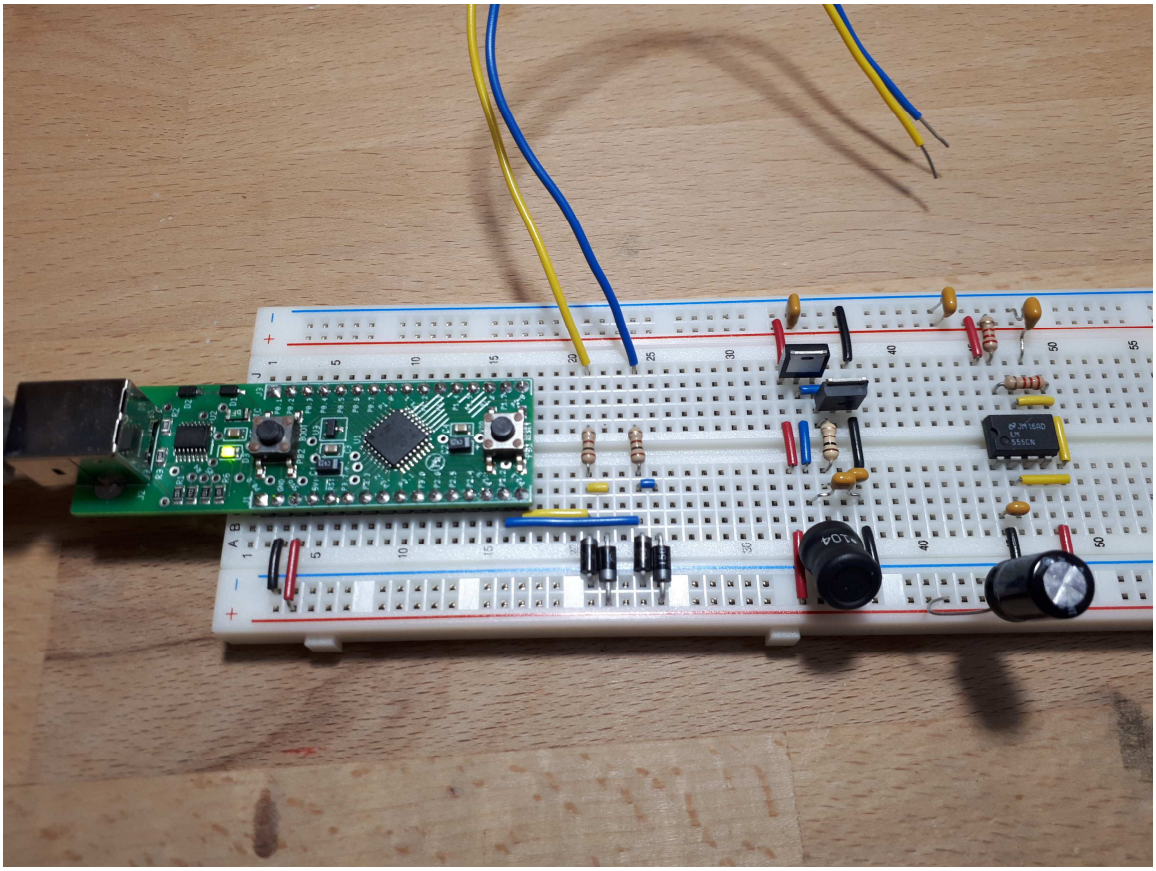


The purpose of the voltage clippers is to prevent input voltages outside the range of operation of the EFM8LB1 microcontroller. Applying voltages outside the operation range of the microcontroller may permanently damage the device, so using this design without a clipping circuit is **not** recommended.

Bread Boarded Circuit

The picture below shows the oscilloscope's bread boarded circuit. It includes the EFM8 board, two voltage clippers (one per channel) as well as two test circuits. Test circuit 1 is a Colpitts oscillator using a discrete CMOS inverter (C1=C2=0.1uF; L=75mH). Test circuit 2 is a 555 timer configured as an A-stable oscillator (R1=R2=2.2kΩ; C=0.1uF). The test circuits were explained in class and will not be covered here. The yellow wire is used as the probe for channel 1 and the blue wire is used as the probe for channel 2.

[1] If Schottky diodes are not available, regular signal diodes like the 1N4148 may be used as well.



EFM8 Board Configuration

To configure the EFM8 as an oscilloscope, compile and load the file "*Oscilloscope.c*" in to the board using the procedure described in class.

Python Script

The script "*scope_tk.pyw*" serves as the front end GUI for the oscilloscope. It has been tested with Python versions 3.5, 3.6, and 3.8. A working distribution of Python[2] with all the tools/classes required by the script is "WinPython". It can be downloaded from:

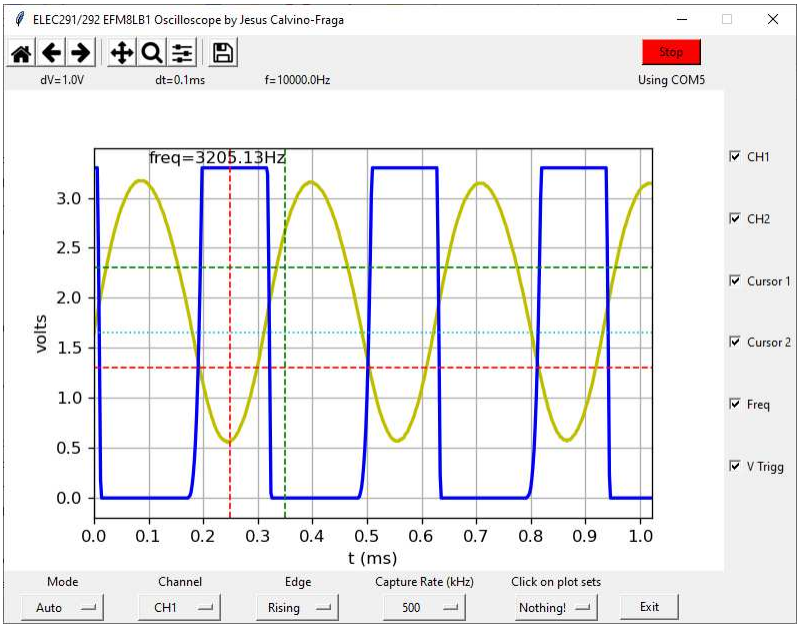
https://winpython.github.io/

The distribution used to develop the script was "*WinPython64-3.8.1.0cod*". If you install this file, remember to run the program "*WinPython Control Panel.exe*" as an administrator, click the 'Advanced' tab, and then "register distribution" so files with the '*.py*' and '*.pyw*' extensions are associated with this distribution of Python.

[2] For Microsoft's Windows operating systems. The python script was tested with Windows 10 and Windows 7.

Operation and Examples

After configuring the EFM8 board with compiled hex file of "*Oscilloscope.c*", the Python script "*scope_tk.pyw*" can be executed right away by double clicking the script icon. A window like the one below is displayed. The script determines which serial communication port to use automatically.



The figure above shows the two channel waveforms: yellow is for channel 1 and blue is for channel 2. The '**Mode**', '**Channel**', and '**Edge**' option menus control the type, source, and edge for the oscilloscope trigger source in a similar manner to the oscilloscopes in the lab. The vertical axis of the display is fixed as this oscilloscope lacks a sophisticated analog front end. That being said, the standard Python Matplotlib tool pushbuttons allow for panning and zooming of the displayed waveforms. The '**Capture Rate**' option menu allows for the selection of different sampling rates which determines the size of the horizontal axis.

The '**Click on plot sets**' option menu tells the script what happens when the plot display screen is clicked on:

1. If the option '**V trigg**' is selected, clicking the screen sets the trigger voltage level. The trigger voltage level is displayed as a thin cyan dotted line in the screen.

2. If the option '**Cursor 1**' is selected, clicking the screen set the cursor 1 position. The cursor 1 position is displayed as two intersecting thin red dashed lines in the screen.

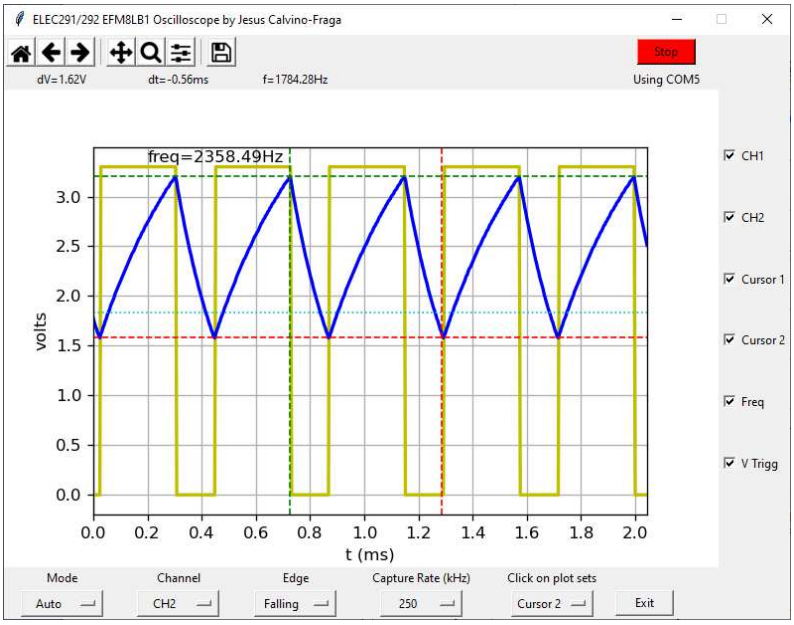
3. If the option '**Cursor 2**' is selected, clicking the screen set the cursor 2 position. The cursor 2 position is displayed as two intersecting thin green dashed lines in the screen.

4. If the option "**Nothing!**" is selected, nothing happens either to the cursors 1 or 2 or the trigger voltage level when clicking the plot display screen. This option must be selected when using the pan and zoom options from the Matplotlib tool menu pushbuttons.

Besides the standard Matplotlib tool menu pushbuttons the position of the pointer in the screen is displayed. Underneath the toolbar we have: '**dV**' which shows the difference in voltage between cursor 1 and cursor 2; '**dt**' which shows the difference in time between cursor 1 and cursor 2; and '**f**' which shows the inverse of '**dt**' when '**dt**' is not zero.

The **approximate** frequency displayed in the figure above (freq=3205.13Hz) corresponds to the signal from which the trigger is derived. The curves and indicators in the graph can be turned on/off using the check boxes to the right. The graph can be temporarily frozen by clicking the 'Stop' button. The previous figure was obtained while the Colpitts oscillator was connected to the oscilloscope. Channel 1 shows the signal at the 100Ω resistor before the gate pins of the CMOS gate. Channel 2 shows the signal at the drain pins of the CMOS gate.

The figure below was obtained while the 555 timer A-stable oscillator was connected to the oscilloscope. Channel 1 shows the output of the timer (IC pin 3), while channel 2 shows the voltage at the capacitor (IC pin 2). Notice that all the option menus have been adjusted from the defaults.



The next figure shows a zoom of a portion of the figure above. Pressing the 'Home' push button returns to the regular display as shown in the previous figure.