CSE 121L: Microprocessor System Design Lab
Fall 2020
Lab Project
TinyScope: A Dual-Channel Oscilloscope Based on PSoC-6

In this final project, you will use the PSoC-6 microcontroller along with the NewHaven Display 240x320 TFT LCD display to design a dual-channel oscilloscope. You will use the SAR-ADC of the PSoC-6 to sample the waveforms. The user interface will be through a terminal window on the host. You will also use two potentiometers to scroll waveforms.

Features

- Sampling rate of 250 ksamples/second.
- Single or dual analog channels
- Free-running or trigger modes
- Configurable trigger channel, trigger level and trigger slope
- Configurable X and Y scales
- Frequency measurement and display
- Potentiometers for scrolling waveforms

Recommended Reading

XYZs of Oscilloscopes by Tektronix (download from Canvas, Files/Lab Resources).

NewHaven 320x240 TFT display data sheet

emWin User Guide

User Interface

The user interface for TinyScope will be through a terminal window, connected to the PSoC through a USB-UART link. The user commands are listed below. The TinyScope should respond to each command with text indicating the command was accepted or there was an error. White spaces in command strings should be ignored.

1. Free-running/trigger mode: In the free-run mode, the TinyScope displays a free-running signal that may not be stable. The trigger mode aligns the sampled signal using a trigger level and trigger slope, so that repeating waveforms will appear stable on the screen. By default, the mode is set to the free-running mode. The command

   ```
   set mode free
   ```

   sets the scope to the free-running mode. The TinyScope responds with

```
mode set to free-running
```

To set the mode to trigger, the user enters the command

```
set mode trigger
```

The TinyScope responds with

```
Mode set to trigger
```

This command can be entered only when the TinyScope is in stopped state.

2. Setting the trigger level: The trigger level can be set from 0V to 3V in steps of 100 mV. By default, the trigger level is set to 1V. To set the trigger level to 2.1V, the user enters the command

   ```
   set trigger_level 2100
   ```

   where the trigger level is specified in milliVolts. The TinyScope responds with

   ```
   Trigger level set to 2100 mV
   ```

   This command can be entered only when the TinyScope is in stopped state.

3. Setting the trigger slope: The trigger slope can be either **positive** or **negative**. Positive slope indicates that the time sweep of the waveform should start on a rising edge when the signal crosses the trigger level. Negative slope indicates that the time sweep of the waveform should start on a falling edge when the signal crosses the trigger level. By default, the trigger-level is set to positive. To change it to negative, the user enters the command

   ```
   set trigger_slope negative
   ```

   The TinyScope responds with

   ```
   Trigger slope set to negative
   ```

   To change it back to positive, the command is

   ```
   set trigger_slope positive
   ```

   This command can be entered only when the TinyScope is in stopped state.

4. Setting the trigger source: The trigger source can be channel 1 or 2. The user can set the trigger source using the command

   ```
   set trigger_channel 1
   ```

or

```
set trigger_channel 2
```

The default is channel 1. This command can be entered only when the TinyScope is in stopped state. The TinyScope should print a confirmation message indicating that the command was executed successfully.

5. Setting x scale: The x scale defines the horizontal scale of the waveform display in microseconds per division. Its value can be 100, 200, 500, 1000, 2000, 5000 or 10000. Its default value is 1000 (1 millisecond per division). This can be changed using the *set xscale* command. For example, to change it to 200 us per division, the user enters the command

   ```
   set xscale 200
   ```

   The value of xscale can be changed any time, regardless of whether the TinyScope is stopped or running. The TinyScope should print a confirmation message indicating that the command was executed successfully.

6. Setting y scale: The y scale defines the vertical scale of the waveform in millivolts per division. Its value can be 500, 1000, 1500 or 2000. Its default value is 1000 (1V per division). This can be changed using the *set yscale* command. For example, to change it to 0.5V per division, the user enters the command

   ```
   set yscale 500
   ```

   The value of yscale can be changed any time, regardless of whether the TinyScope is stopped or running. The TinyScope should print a confirmation message indicating that the command was executed successfully.

7. Starting and stopping the oscilloscope: On power-on, TinyScope in the stopped state where its sampling and analog display are turned off. To start the oscilloscope function, the user needs to enter the command

   ```
   start
   ```

   The TinyScope will now go into the running state. To stop sampling and freeze the display, the user can enter the command

   ```
   stop
   ```

The program should check for any illegal commands and argument values entered and provide feedback to the user with a message to help them correct the input.

Specifications

Here are some of the key specifications you should design to:

1. **Sampling rate**: The sampling rate determines the highest frequency that you can support for your input signals. The SAR ADCs in the PSoC have a maximum sampling rate of 1 Megasamples/second for a single channel. Because we will be using four of the ADC channels (2 for the signal channels and 2 for sampling the potentiometer settings), the sampling rate for each oscilloscope channel will be 250,000 samples/second.
2. **Sampling resolution**: The sampling resolution is always 12 bits per sample.
3. **Input voltage**: The range of the input voltage for each channel is 0 to +3.3V.
4. **Frequency Measurement**: The oscilloscope program should continually calculate the frequency of the signals on both channels and update the display periodically.
5. **Scrolling waveforms**: The design should include two potentiometers, which can be used to scroll the waveforms of the two channels up and down.

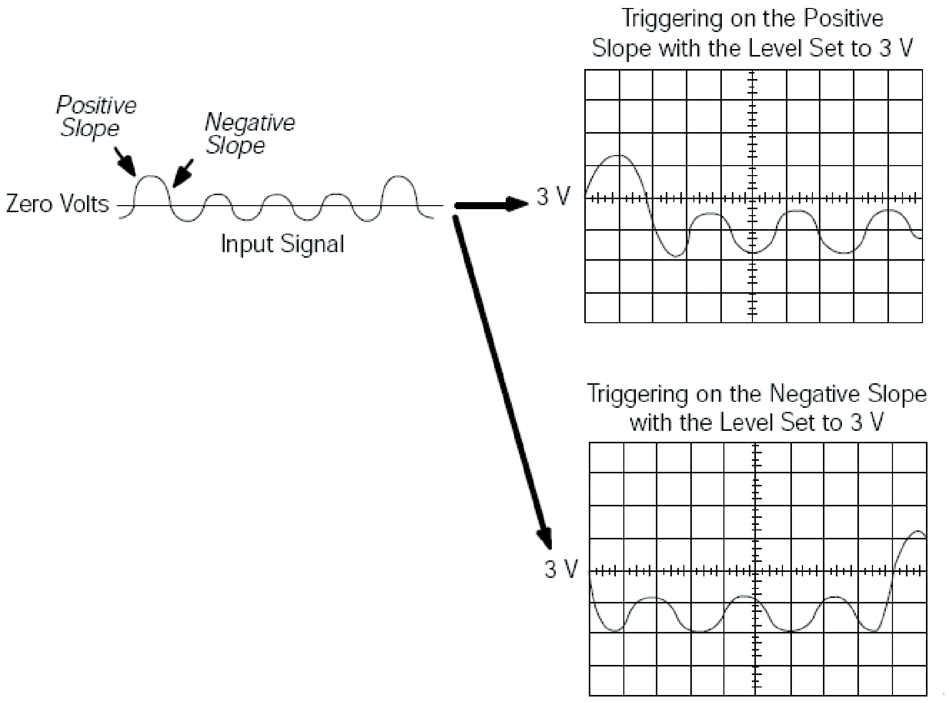

Triggering on the Positive Slope with the Level Set to 3 V
Positive Slope
Negative Slope
Zero Volts
Input Signal
3 V
Triggering on the Negative Slope with the Level Set to 3 V
3 V

Illustration of how the trigger level and slope affect the waveform display (source: Tektronix).

While many of the internal details of your design are left for you to work on, here are some guidelines that will get you started:

1. **Moving data from the ADCs:** You should use a separate DMA channel to move data from each of the two ADC signal channels into separate ping-pong buffers,

similar to the design in Lab 2. The two potentiometer channels can be sampled using polling.

2. **Detection of Trigger Condition:** Let a_n denote the amplitude of the *n*th sample of the signal. For rising edge trigger, the trigger conditions consist of
   (i) The previous sample $a_{n-1} <$ trigger level.
   (ii) The current sample $a_n \geq$ trigger level.
   (iii) The current slope of the signal is positive.

   For falling edge trigger, the trigger conditions consist of
   (iv) The previous sample $a_{n-1} >$ trigger level.
   (v) The current sample $a_n \leq$ trigger level.
   (vi) The current slope of the signal is negative.

   The samples may need some smoothing to detect positive and negative slopes of the signal reliably. False reading of slope from high-frequency glitches can be prevented by passing the signal through a low-pass filter. This can be done by maintaining a running average of the signal samples. For example, if a_n denotes the amplitude of the *n*th sample, we can compute a moving average

$$b_n = (a_{n-4} + 2a_{n-3} + 3a_{n-2} + 2a_{n-1} + a_n)/9.$$

   The slope can then be determined by comparing b_n with b_{n-1} (note that the division by 9 is not required as we are only interested in the sign of the result).

   The smoothing part is not required for the project, but it is recommended.

Finally, you need use a graphics library to design the waveform display. We will use the **emWin** library for this. Information on installing and using this tool, with example code, will be posted separately.

You can use the AD-2 waveform generator to generate various periodic waveforms and test your oscilloscope with them (sine, square, ramp). If the frequency of the test pattern is changed while the oscilloscope program is running in the trigger mode, the waveform should change on the display, while remaining stable.

Getting Started with the Design

1. Connect the TFT LED display to the PSoC board: This is explained in a video posted on Canvas.
    a. The table below specifies the connections between the PSoC and the LCD display. Create a schematic before wiring the pins.

PSoC-6 Signal Name	**PSoC-6 Port**	**Display Interface Signal Name**	**Display Interface Pin Number**
d[0]	P9[0]	DB8	22
d[1]	P9[1]	DB9	23
d[2]	P9[2]	DB10	24
d[3]	P9[4]	DB11	25
d[4]	P9[5]	DB12	26
d[5]	P6[2]	DB13	27
d[6]	P6[5]	DB14	28
d[7]	P6[3]	DB15	29
d_c	P6[4]	D/C	11
LCD_RESET_N	P5[2]	/RES	30
nrd	P5[5]	/RD	13
nwr	P5[4]	/WR	12

The following are the power/ground connections and tie-offs associated with the display.

Display Interface Signal Name	**Display Interface Pin Number**	**Connection**
GND	1	GND
VDD	7	VDD (3.3V)
IOVDD	8	VDD (3.3V)
/CS	10	GND
IM0	31	VDD (3.3V)
GND	33	GND
LED-K1	34	GND
LED-K2	35	GND
LED-K3	36	GND

LED-K4	37	GND
LED-A	38	VDD (3.3V)
GND	39	GND

In addition, you will need to allocate pins for a UART (ports P5[0] and P5[1]), and two analog input pins for potentiometers.

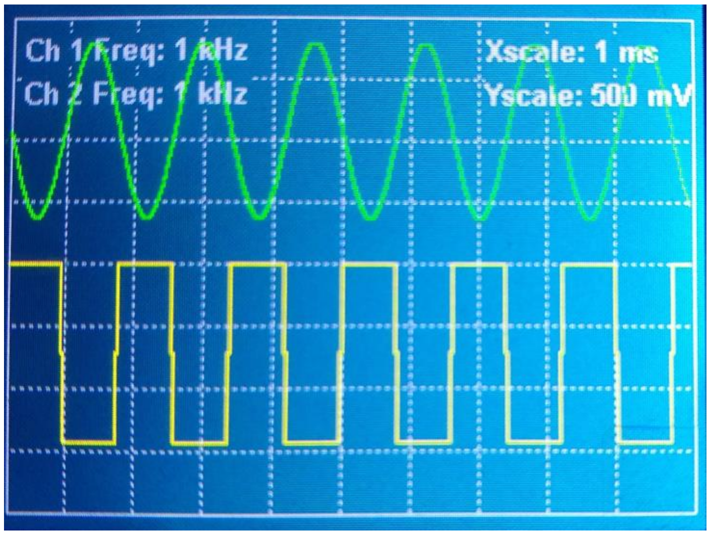
2. Download and run the emWin code example

    2.1 Change pin assignments to conform with the above table.

    2.2 In Project => Build Settings => PDL, check emWin Core and LCD Driver boxes. Select **nOSnTS** as the option for emWin and FlexColor as the option for LCD Driver.

    2.3 Build and run the example. The code demonstrates the use of various API functions of the emWin library.

3. You can use the example PSoC Creator project **Final_project_demo** as a guideline for designing the TinyScope screen. Do not copy this code. Instead, use it as a reference and design your own screen. The waveform display should be similar to a standard oscilloscope screen, showing the data waveforms against a reference grid with the parameter settings (horizontal and vertical scales, signal frequency, etc.) displayed in text. Here is an example for the dual-channel case.



Code Structure

We will not be using FreeRTOS for the project, so you will need to use the main loop to schedule the various tasks. You should keep the main loop short and break your design functionally into multiple C functions. For example, you may want to organize your code as follows:

- Header file: TinyScope.h
- main control loop
- command argument parser
- Communication functions
- graphical display
- data processing and trigger detection
- frequency measurement
- potentiometer input processing

You don’t need all of these functions to start testing your design. Start with a minimal design and keep enhancing it after testing each part.

Maintaining Files on GitLab Server

You must use the campus Git server to regularly back up your source files. During checkoff, you will be asked to pull the latest version of the code from the server and compile it. We will also use the version of the code in your repository at checkoff time for grading.

Checkoff Schedule

The schedule for checkoff and submission of reports is posted on Canvas. We will provide more details on the checkoff tests and report organization later.