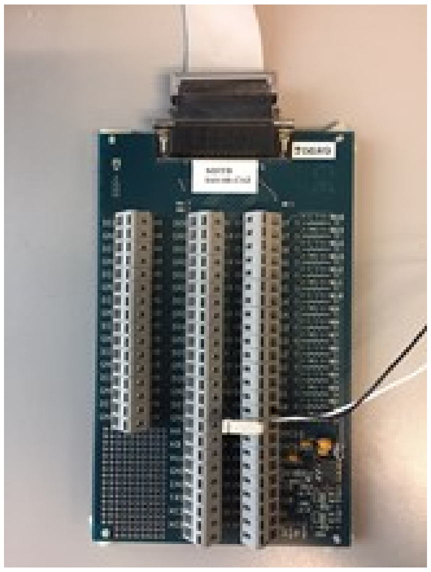
ENGR 325 | **Lab 3** | **Walla Walla University**
Winter 2021 | **Data Acquisition** | **Inst. Mark Haun**

The goal of this lab is to gain experience with a practical analog-to-digital converter (ADC) system and to explore the effects of time sampling.

Objectives:



- Gain experience with a real ADC.
- Review basic MATLAB skills.
- Explore the relationship between sampling rate, input signal frequency, and resulting output.
- Perform rudimentary analysis of digitized signals.

Equipment provided:

- Tektronix TBS2000 series oscilloscope
- Agilent 33250A signal generator
- Windows PC with ADC card and MATLAB
- ADC connection board

Procedure:

Your lab this week includes a new piece of equipment—the Microstar DAP840 analog-to-digital (ADC) interface board. The board is installed internal to the computer at your lab bench. You connect signals to the ADC via the ribbon cable and connector strip shown above. The first thing to do for this lab is to set up the Agilent signal generator to output a signal which you will capture via the DAP840 board and read into MATLAB using a script that has been written for you.

1. Connect the Agilent 33250A signal generator output to the DAP840 board by connecting the signal generator ground to the G0 terminal of the ADC interface board and the signal generator output to the S0 terminal of the A/D interface board. Use the little white "pry bar" to open the spring-loaded clamps, one at a time. You can see the pry bar on the long grey strip at the right. Take care that you do not lose this pry bar.

2. Set the signal generator to produce a sine wave with an amplitude of $2\,\mathrm{V_{pp}}$, $0\,\mathrm{V}$ offset, and a frequency of $500\,\mathrm{Hz}$.

3. Use the Tektronix scope to verify your waveform looks like it should at the connection terminals to the ADC. This is a good time to review some features of the oscilloscope, so put the scope in DC Coupling mode and then use the Measure function to display the peak-to-peak voltage, the frequency, and the RMS voltage. Record these values.

4. Calculate what the RMS voltage should be, and compare this with your measurement. Remember the definition

$$\sqrt{\frac{1}{T}\int_0^T (f(t))^2 dt}$$

(You may find the following trigonometric identity useful: $\sin^2 x = \frac{1}{2} + \frac{1}{2}\cos 2x$.)

5. Start MATLAB on the PC. Download the `lab32021.m` file from the class web site and save it to your STEM drive or to the desktop. This file uses a MATLAB function named `msgets0` to gather data from channel 0 on the DAP840 board installed inside the lab computers. The format of the `msgets0` function is

```
[data,time] = msgets0(rate, Npoints, gain);  % Reads A/D chan 0 "single-ended"
```

Inputs:

`rate`	number of samples per second
`Npoints`	number of data points
`gain`	optional selectable gain (1, 10, or 100 with a default of 1)

Outputs:

`data`	vector of samples from the ADC, as signed integers (see below)
`time`	time vector corresponding to the data samples

Here is a simple example which acquires 100 samples (at a 2-kHz rate) and plots them. Give it a try:

```
[d,t] = msgets0(2000,100);  % rate=2000 samples/s, Npoints=100, gain=1
plot(t,d); grid on;         % Plot the waveform and add a grid
```

(Notice that we use the time vector returned by `msgets0` when plotting. This shows real time on the x axis instead of "sample number" which is what `plot(d)` would do.)

6. One task remains to get a properly scaled waveform plot: We need to figure out scaling for the y axis so it displays in volts. `msgets0` returns "raw" samples from the ADC, so the values in the `d` vector are the decimal equivalents of the ADC's binary output codes. Our DAP840 has a 14-bit ADC, but it returns 16-bit numbers where the 14 MSBs (most-significant bits) are the ADC conversion result and the two LSBs (least-significant bits) are always zero.

   Use the following steps to come up with a conversion factor k which we can multiply by the raw samples to get volts:

   (a) How many values can a 16-bit binary number represent?

   (b) The DAP840 input range is currently set to ±5V (a span of 10V). Because of this *bipolar* input, the output values are interpreted as signed numbers—half of the values negative, and half positive. What are the most negative, and most positive, values for a signed 16-bit number?

   (c) Armed with this information, derive your value for k, then test it by adding the line `d = k*d` before the plot command. Verify that the plotted waveform is correctly scaled with volts on the y axis. (You can double-check this against the oscilloscope if you are uncertain.)

7. A copy of the lab32021.m MATLAB script is shown below:

```
close all;
clear all;

sample_frequency = 50000;   % sampling frequency
number_points_to_gather = 1000;

[d,t] = msgets0(sample_frequency, number_points_to_gather);
d = k * d;  % Convert ADC values to voltage (replace k with your own factor)

knownF = 800;           % input frequency in Hz
numberCycle = 3;        % number of waveform periods to display

pointsPerCycle = sample_frequency / knownF;   % num data points per waveform period
timeToShow = numberCycle * pointsPerCycle * (1 / sample_frequency);

plot(t,d);              % plot with default (blue line, no dots)
hold on;
plot(t,d,'ok');         % overlay with black circles at data points
axis([0 timeToShow -1.1 1.1]);
hold off;
grid on;
title('Captured signal-generator waveform');
xlabel('Time (s)');     % Correct axis labeling always includes the
ylabel('Voltage (V)');  % variables being plotted *and* their units!

msstop;
```

8. Modify the script to display five cycles (periods) of your digitized sine wave on a MATLAB plot. Include the plot and your code in your report.

9. Now that you understand the basics of signal acquisition using an ADC board and MATLAB, the next step is to do some signal analysis from within MATLAB. Set your function generator to output a sine wave of 1000 Hz, 3 V_{pp}, and 0 V offset.

   Add MATLAB code to `lab32021.m` to replace the hard-coded value for `knownF` with a value computed from the captured data. Also compute the peak-to-peak voltage from the data, and use that to auto-scale the plot. (Once you have your peak-to-peak voltage, you can replace the ± 1.1 in the `axis` command with your actual zero-to-peak value.)

   (Hint: MATLAB provides `max` and `min` functions. Also, getting frequency or period gives you the other automatically. Can you count the samples between zero crossings—where the samples go from negative to positive, or vice versa?)

   Display the following parameters on your plot (see the MATLAB `text` command):

   (a) Peak-to-peak voltage

   (b) Frequency

   (c) Period
   (d) Stretch goal: RMS voltage

10. Change the script to capture the same signal-generator waveform with the following sampling frequencies. For each one, include the plot and comment on your observations.
    (a) 20,000 Hz
    (b) 2,000 Hz
    (c) 100 Hz

11. Using the Arb function on the signal generator, choose Select Waveform, then Built-in waveforms, then the Cardiac option to output an approximation for an EKG signal. What is the lowest frequency that you can sample at and still reliably see all parts of the waveform? Include a plot at this sampling frequency.

12. Run the function `msstop` to shut down the connection with the A/D board, then exit MATLAB before signing off.