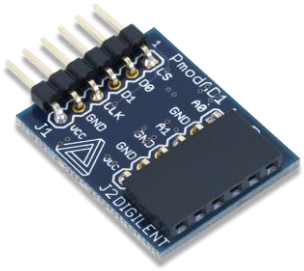
ECE3623 Embedded System Design Laboratory

Dennis Silage, PhD
silage@temple.edu

PmodAD1 Vivado SDK Project v2

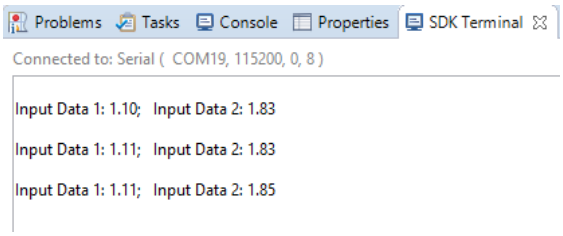
In this Lab 6 the PmodAD1 analog-to-digital converter (ADC) SPI peripheral from the Digilent Vivado IP Library is to be configured and used to perform measurements on an analog test signal. The *Zynq SPI Peripherals ADC and DAC* PPT posted on Canvas is the reference for the PmodAD1 ADC. The PmodAD1 Vivado SDK example project main.c is shown below and is also presented in the lecture.

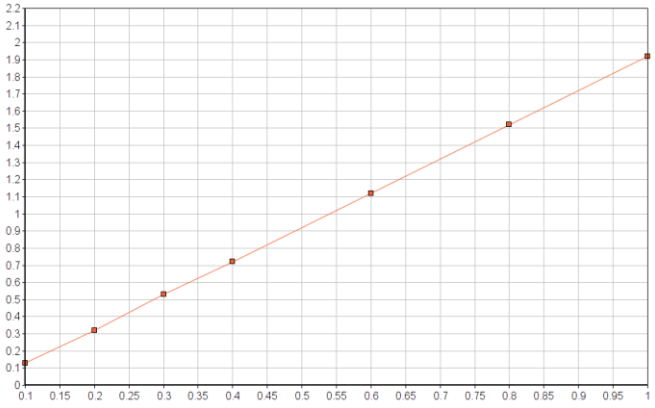


In this Lab 6 you are to use the Digilent Discovery 2 Board and the Waveforms application to generate an analog test signal interfaced to channel 1 of the PmodAD1 ADC. You should have installed and verified the performance of the Waveforms software. The Tasks for this Laboratory are as follows:

1. Execute the example PmodAD1 project in Vivado SDK. The PmodAD1 is connected to JE on the Zybo Board. Setup the Discovery 2 to input a constant voltage from the Arbitrary Waveform Generator (AWG) to channel 1 of the PmodAD1.

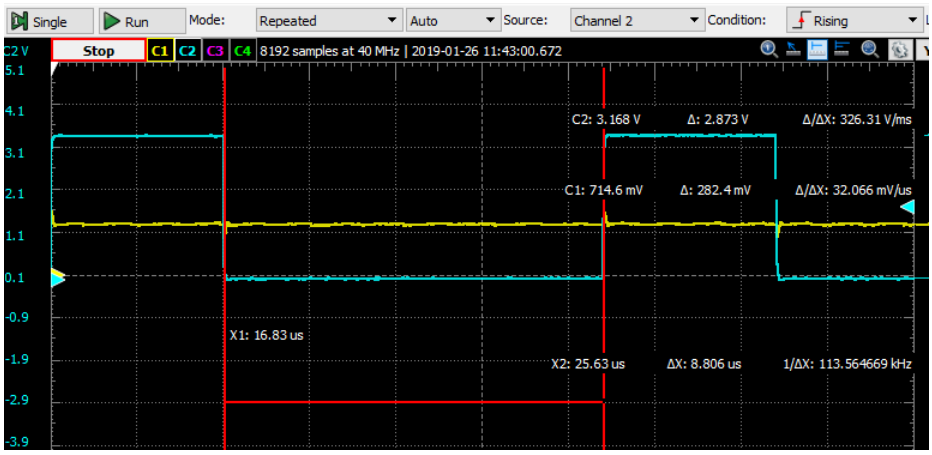
With the PmodAD1 example project executing in the SDK terminal, record the measured voltage for steps in the input voltage. In this example the range was 0.1 to 1 V input which produced the plotted data shown.





Using a calculation utility, obtain the *best fit* linear slope and y-intercept for the data. In the example here, the slope was 2 and the y-intercept was −0.08 V. Describe your results.

2. Use the oscilloscope to measure the Slave Select (or Chip Select CS) pin 1 of the PmodAD1 input. With the PmodAD1 example project executing in the SDK terminal measure the CS pulse. In the example here the CS pulse is active low for 8.8 µsec and inactive high for 4.83 µsec or an active duty cycle for ADC conversion of 8.8/(4.83+8.8) ≈ 64.6%. The data rate is 1/(13.63 µsec) ≈ 73.4 ksamples/sec.



3. Comment out the *xil_printf* and *sleep* statements in *main.c.* Be sure to be editing the *main.c* in the project and not the original version in the example. Save the *main.c* and evoke *Run As…Launch on Hardware* (4). Determine the active duty cycle and data rate. What can you infer by the changes in the measurements here?

4. Create a new project for the PmodAD1 in Vivado. The board design in Vivado is to use the PmodAD1 and a single GPIO with two channels. GPIO channel 1 is connected to the LEDs and GPIO channel 2 is connect to the slide switches (SW).

In the SDK you will configure a *File…New Application*. The example *main.c* program can be incorporated into the project as before. However *main.c* will be modified to perform the following tasks:

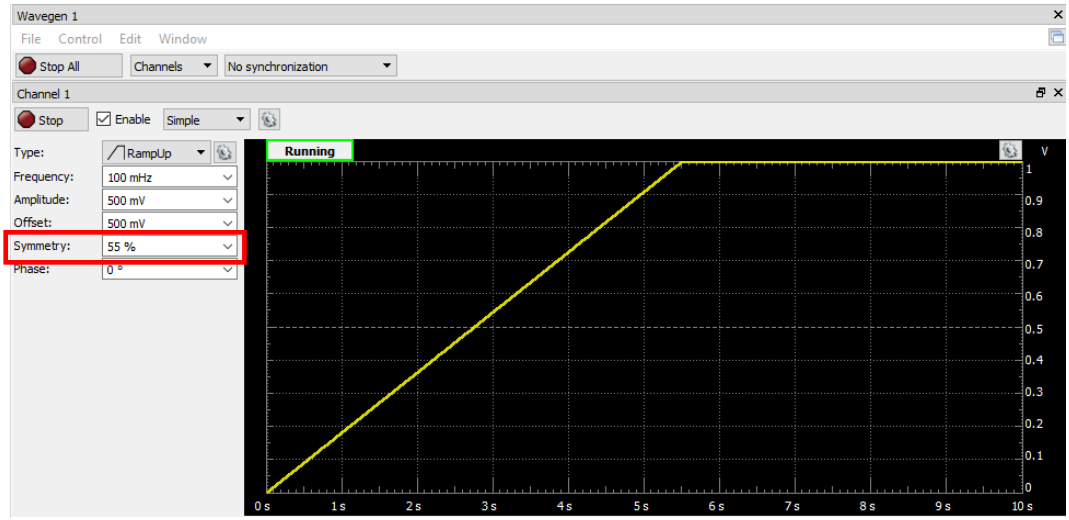
4a) SW0 OFF, SW1 OFF. The project prints *ECE3622 Lab 6* on the SDK terminal every 5 seconds.

4b) SW0 is ON, SW1 is OFF. The input voltage signal from the AWG is a unipolar ramp at 0.1 Hz (Waveforms indicates this as 100 mHz) with a range from 0 to 1 V. Use the oscilloscope to verify that the input signal is correct. The individual LEDs are to be ON if the voltage input exceeds the thresholds listed for the four LEDs:

LED0 0.2 V LED1 0.4 V LED2 0.6 V LED3 0.8V

If the voltage input is less than the threshold, the associated LED is OFF. There is no SDK terminal output, except for a single string that identifies this event. You are to use the results of the slope and offset determination from Task 1 to ensure accuracy.

4c) SW0 is OFF, SW1 is ON. The input voltage signal from the AWG is a unipolar ramp followed by a pulse from 0 to 1 V at 0.1 Hz, which then repeats. There are symmetry differences for these signals. For example, shown below is a unipolar ramp from 0 to 1 V at 0.1 Hz but with a symmetry of 55%.



Using multiple ADC samples of the unipolar ramp and pulse signal estimate the symmetry and print out the measured percentage every 10 seconds approximately. A threshold near 1 V can be used used to stop the sample count for the unipolar ramp (N_{ramp}) and the sample count from that point to the start of the next cycle is that for the pulse (N_{pulse}). The symmetry percentage then would be:

$$100 \times N_{ramp} / (N_{ramp} + N_{pulse}).$$

You are to use the results of the slope and offset determination from Task 1 to ensure accuracy. Verify your calculated percentages to that set by the AWG of the Discovery 2 setting at several instances.

Results to be described in the Project report must include the following:

1. Why is the gain indicated by the slope of the example PmodAD1 not unity?

2. Screen shots of the oscilloscope and the AWG verify the measurements made and are to be included. Result should be clearly indicated and discussed.

This Lab 6 is for the week starting March 9th and due no later than Tuesday March 16th 11:59 PM with an upload to Canvas and a hard copy to the Laboratory Assistant.

```
//AD1Pmod.c  PmodAD1 ADC example  ECE3622  c2019 Dennis Silage

#include "xparameters.h"
#include "xil_io.h"
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include <sleep.h>

//AD1Pmod from Address Editor in Vivado
#define AD1acq          0x43C00000    //AD1 acquisition
#define AD1dav          0x43C00004    //AD1 data available
#define AD1dat1         0x43C00008    //AD1 channel 1 data
#define AD1dat2         0x43C0000C    //AD1 channel 2 data

int main(void)
{
        int adcdav;                     //ADC data available
        int adcdat1;                    //ADC channel 1 data
        int adcdat2;                    //ADC channel 2 data


        xil_printf("\n\rStarting AD1 Pmod demo test...\n");
        Xil_Out32(AD1acq,0);
        sleep(5);

        while (1)
        {
                Xil_Out32(AD1acq,0);                    //ADC stop acquire
                adcdav=Xil_In32(AD1dav);                //ADC available?
                while(adcdav==1)
                                adcdav=Xil_In32(AD1dav);

                while(1)
                        {
                                Xil_Out32(AD1acq,1);    //ADC acquire
                                while(adcdav==0)
                                        adcdav=Xil_In32(AD1dav); //ADC data?
                                Xil_Out32(AD1acq,0);    //ADC stop acquire
                                adcdat1=Xil_In32(AD1dat1);
                                adcdat2=Xil_In32(AD1dat2);
                                while(adcdav==1)
                                        adcdav=Xil_In32(AD1dav); //ADC reset?
                                xil_printf("%d  Ch1  %d, Ch2  %d\n", adcdat1, adcdat2);
                                sleep(1);
                                }
        }
}
```