Pre-Lab 4

1.) The purpose of this week's lab is to understand the use of ADC and DAC by manually assigning timers to perform SOC, EOC, data fetch, data conversion and response. Another objective is to learn how to combine the learnings so far: watchdog timer, interrupts, GPIO, and now ADC to ensure unconditional and conditional execution of the code (case dependent). In the first part of the lab we learn how to generate analog signals to make the microcontroller act as a signal generator, and also learn how to read analog signals to make the microcontroller act as an oscilloscope. We also learn how to create a data logger to store the values read in from the GPIO pin and then plot it using the inbuilt graphing. We will also learn how to filter signals by accepting analog waveforms on the input and producing analog signals on the output.

2.) We use the generated by the ADC for two specific tasks:
(i) To conduct sample-and-hold at the true end of the EOC pulse
(ii) To issue the EOC pulse at the true end of conversation.

We can use it if we want to convert more than one pin voltage in response to a single SOC trigger – while multiple ISRs can be assigned for the same task but we can achieve the desired results through the use of just one ISR.

***Using the ADC interrupt allows us to avoid the need for continuous polling to determine if the conversion result registers have been updated with new valid data. ***

3.) First, in order to generate analog signals, we need to set up the DAC.

```
CpuSysRegs.PCLKR16.bit.DAC_A = 1; //enables DAC clock
asm("NOP");
asm("NOP");

DacaRegs.DACCTL.bit.DACREFSEL = 1; //we select use of VREFHI
DacaRegs.DACOUTEN.bit.DACOUTEN = 1; //enable DAC
DacaRegs.DACVALS.all = 4096*Vcmd/Vref;  //setting voltage output
```

Next, we need to be able read this output signal from a GPIO pin, so we need to set up the ADC

```
CpuSysRegs.PCLKR13.bit.ADC_A = 1; //enables ADC clock
asm("NOP");
asm("NOP");
```

(value -→ adcclk/sysclk = 50Mhz/200Mhz = ¼ ……NOT SURE OF THIS, since ratio is ¼ but the table in the DATASHEET only has ratio vals>1….. So I did sysclk/adcclk)

```
AdcaRegs.ADCCTL2.bit.PRESCALE = 6; // since ratio is 4.
AdcaRegs.ADCCLT1.bit.ADCPWDNZ = 1; // power ADC
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 1; //timer trigger
AdcaRegs. ADCSOC0CTL.bit.CHSEL = 2; //AIN for J3-9
AdcaRegs. ADCSOC0CTL.bit.ACQPS = 39; //since ADC ack time is 200ns for S-H stage
```

Next, we need to set up the ADC interrupts: we use interrupt one.

```
AdcaRegs.ADCCTL1.bit.INTPULSEPOS = 0;
AdcaRegs.ADCINTSEL1N2.bit. INT1SEL = 0;
AdcaRegs.ADCINTSEL1N2.bit.INT1E = 1; //enable ADCINT1

PieVectTable.ADCA1_INT = &AdcIsr;
PieCtrlRegs.PIEIER1.bit.INTx1 = 1;
```

Read and Map the conversion result

```
Vmea = AdcaResultsRegs.ADCRESULT0*Vref/4096;
```

Clear the Interrupt Flag

```
AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;
```

Acknowledge the interrupt

```
PieCtrlRegs.PIEACK.all = M_INT1;
```