EEL 4742C: Embedded Systems
Homework 5

QUESTION 1. **(10 points)**

The analog-to-digital converter (ADC) module uses a clock signal that varies in the range [20 -40] Hz due to the operating conditions and error. The sample-and-hold time of the signal we're converting is 3 seconds. The ADC allows selecting a number of cycles from the list below.

Number of cycles: 10, 40, 80, 110, 150, 200, 250, 300

Part a) Show the analysis for choosing the suitable number of cycles.
Max = 40Hz, Total period = 3s, Total Cycles = 40 * 3 = 120, so we need 150 cycles.

Part b) The conversion takes 10 cycles. What is the total operation time (that includes the sample-andhold time and the conversion)?
150 + 10 = 160cycles over 40Hz, 160/40 = 4 seconds.

Note: the numeric values in this question are chosen to simplify the computations; an ADC usually uses a much higher clock frequency and the sample-and-hold time is usually in the micro-seconds range.

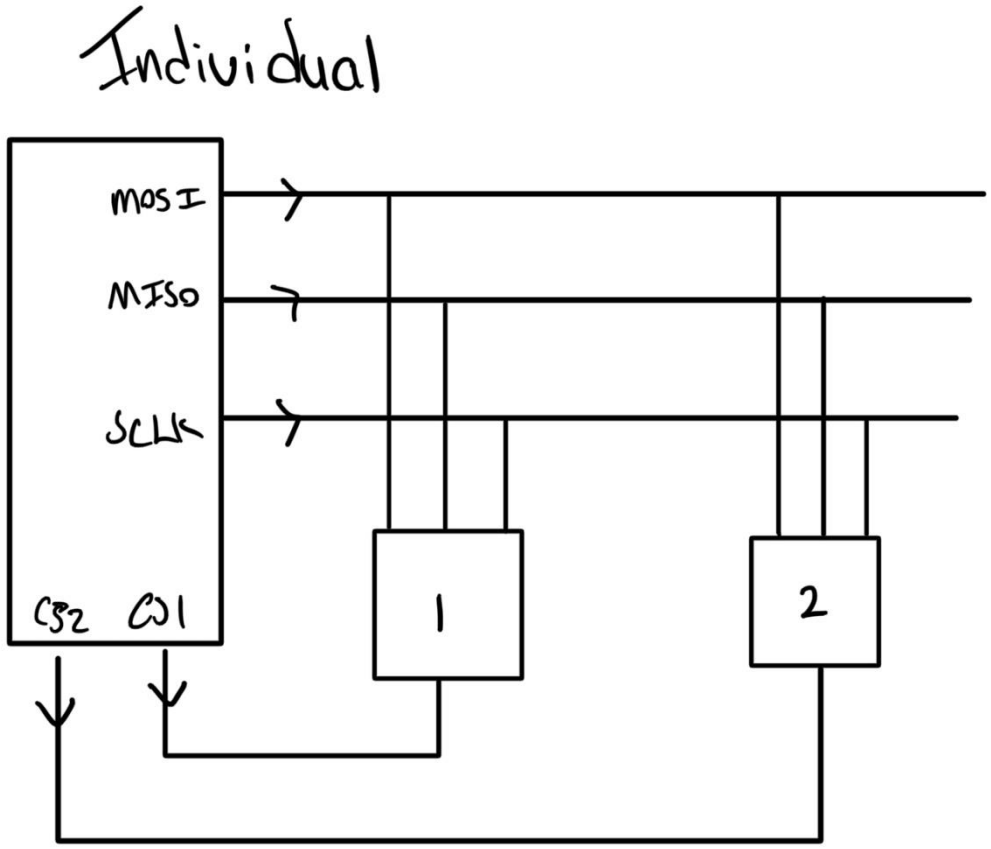
QUESTION 2. **(10 points)**

An SAR ADC has a 3-bit resolution (result). The reference voltages are Vr-=0 and Vr+=Vr. The signal we're converting is: Vin = 3.4/8 Vr+. Show all the voltage comparisons that are done by the ADC and show the conversion's binary result.

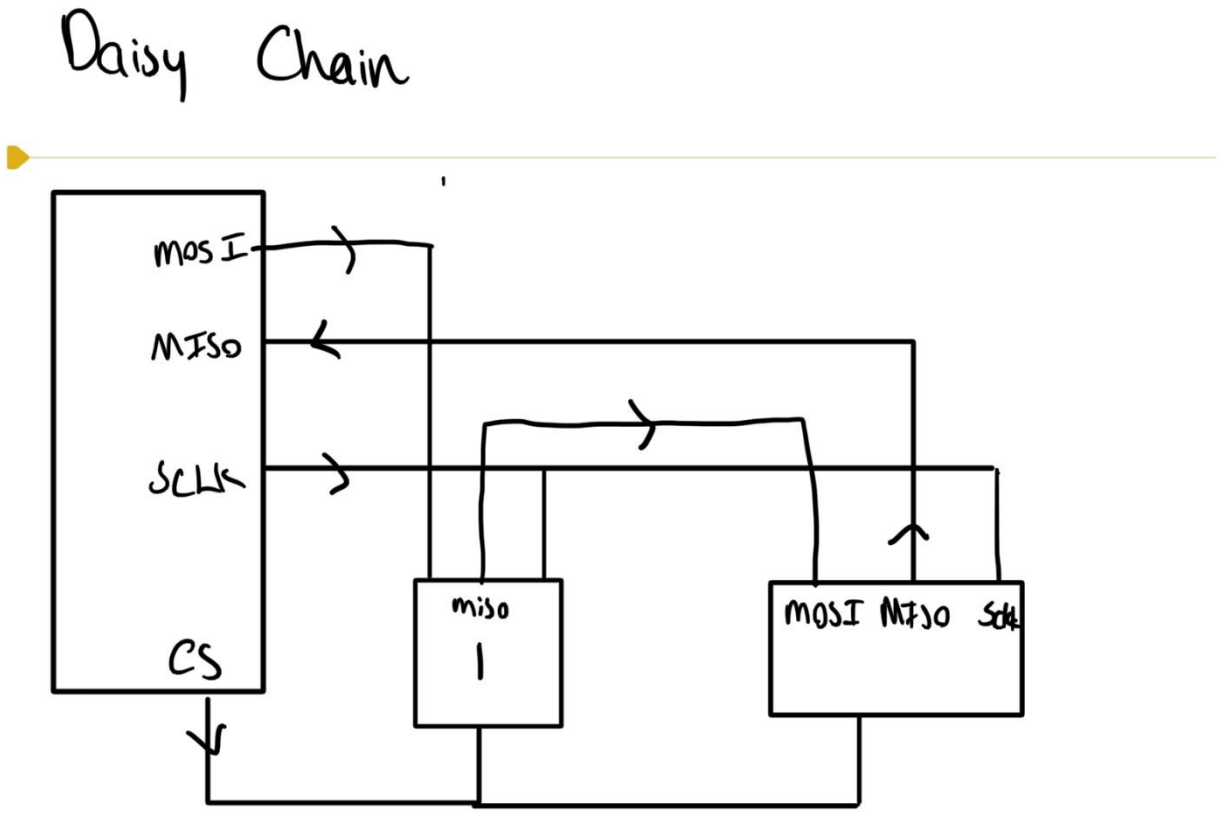
We need to compare for each bit so we compare at Vr/2 first, which gives us a 0, then Vr/4 which gives us a 1, and then 3Vr/8 which gives us another 1, so we get 011.

QUESTION 3. **(10 points)**

Part a) Draw the configuration where an SPI master is connected to two devices that are individually selected.



Part b) Draw the configuration where an SPI master is connected to two devices in a daisy chain.



Part c) What is the 3-pin SPI configuration? What's the difference between 3-pin SPI and 3-wire serial?
The 3 pin SPI configuration is MOSI, MISO, and SCLK, which are the master out station in, master in station out, and serial clock lines. MOSI is data from the master to the station and MISO is the opposite, while it is obvious what the system clock is. The CS pin for 3 pin SPI is always set to ground. For a standard 3 wire terminal there is data, clock, and ground, but this is a more general term since devices don't necessarily need to be set up in a certain configuration.

QUESTION 4. **(10 points)**

Part a) Describe the graphics software stack in embedded systems.

The software top down goes application to graphics library to driver to display module.

Part b) What services does the display driver provide? To what party does it provide these services? Comment on the compatibility of a driver with multiple displays.

The driver gives low-level instructions to the display controller, this tells the hardware what the software is saying. Often a driver won't work with multiple displays since drivers are hardware specific.

Part c) What services does the graphics library provide? To what party are these services provided? Comment on the compatibility of a graphics library with multiple displays.

A graphics library can turn simple commands into more complicated commands like drawing a line or rectangle or printing text. What is nice about graphics libraries is that they can work with any display that has a compatible driver.

Part d) What is a graphics context?

The graphics context, as its name implies keeps the context of what is happening on the display. This is the logical display as seen by the graphics library and we can have multiple contexts on the same display.

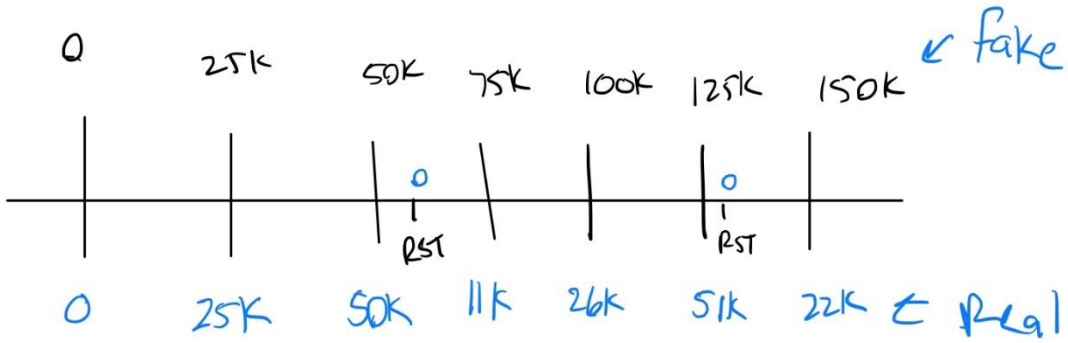
Part e) What is the clipping region of a graphics context?

The clipping region of the graphics context is the boundary of the graphics context. Nothing can be drawn outside of those boundaries on the display.

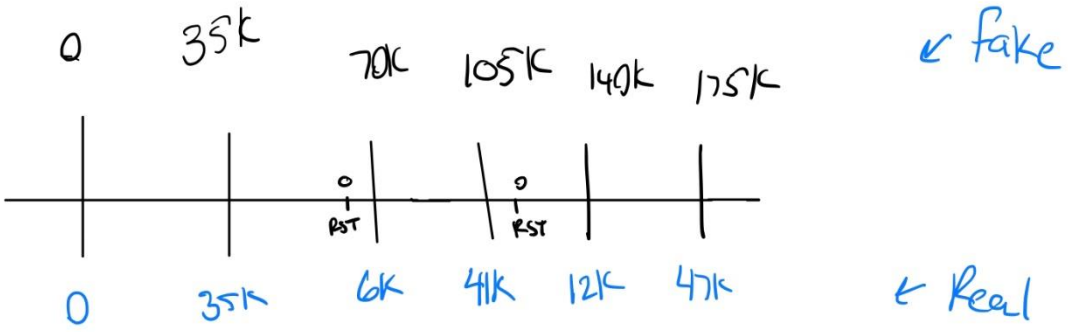
QUESTION 5. **(10 points)**

Part a) Channel 1 is scheduling periodic interrupts every 25K cycles (K=1024). Show how the interrupts are scheduled and demonstrate that the operation is still correct when a rollback to zero occurs on TACCR1. Show enough cases that span two rollbacks to zero.

Part b) Channel 2 is scheduling periodic interrupts every 35K cycles (K=1024). Show how the interrupts are scheduled and demonstrate that the operation is still correct when a rollback to zero occurs on TACCR2. Show enough cases that span two rollbacks to zero.



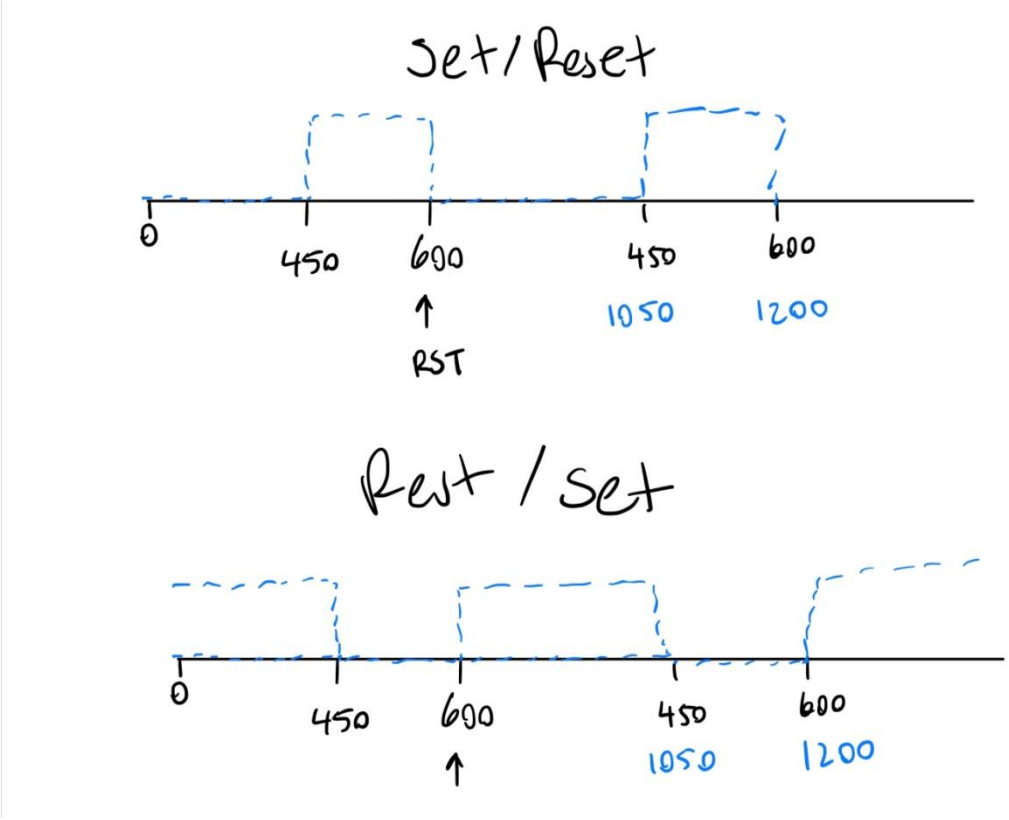
Clock Resets @ 64K



QUESTION 6. **(10 points)**

Part a) The timer is running in the up mode with a period of 600 cycles. Channel 1 engages the output mode (Set/Reset) and sets its register TACCR1 = 450. Explain what happens. Draw a timeline.

Part b) Repeat the previous part based on the (Reset/Set) mode.



QUESTION 7. **(10 points)**

Each time an event occurs (e.g. button push), the program needs to time a duration of 0.25 seconds (using ACLK=32 KHz) and raise an interrupt when this duration elapses. Note that this interval is timed only when the event occurs, not repeatedly back-to-back.

Part a) Can we set this up using the up mode? If yes, explain how.

Yes we can. We simply need to reset the channel and then set TACCR0 to 8kHz – 1, this will then raise the interrupt.

Part b) Can we set this up using the continuous mode? If yes, explain how.

Yes we can. Simply set the TACCR0 to 8kHz above the current value for TAR.

Part c) If both modes are possible, explain the preferable mode.

Up mode is preferable because continuous mode is always running the clock but up mode only runs the clock when desired.

Part d) If the timer is used intermittently, is it possible to turn the timer on/off throughout the program? How is this done?

An interrupt can be used to edit the timer register or simply set the msp to a lower power mode.

Part e) Timer_A is running in the continuous mode and none of its interrupts is enabled. TAR has cycled the full range (0 to 64 K) a few times and the program didn't interact with the timer's bit fields or registers. At this point, what is the value of each of the flags: TAIFG, CCIFG flags of Channels 0, 1, 2?
For Channel 0 the flags would be reset but for channel 1 and 2 the flags would stay set at 1.

Part f) In our program, when an interrupt event occurs, it's disabled for a little while (e.g. 10 seconds) and then it's re-enabled. When is the best time to clear the interrupt's flag: when it disabled or when it's reenabled?

The best time to clear the flag is before so the interrupt event isn't immediately ran again.

QUESTION 8. **(10 points)**

Part a) There are multiple levels of programming microcontrollers that are shown in the list below. Which level did we use in this course?

- Assembly level
- Register level
- Library level
- Operating system level

We used register level.

Part b) The Arduino line of microcontrollers is very popular thanks to the programming library known as the Wiring API. This API provides simple functions that can be used by the programmer. These functions are implemented for all the Arduino microcontrollers. Therefore, the user can use the same (portable) code for any Arduino device chosen. The Wiring API has been implemented for MSP430 microcontrollers in the Energia IDE.

To give you an idea of how an API is implemented, let's write a function that's similar in style to the Wiring API. The function finds the first available channel of Timer_A. A channel is considered available if its CCIE bit is disabled. This is the header of the function. It returns the first available channel out of Channels 0, 1, 2, in this order. If all the channels are used, the function returns -1. Write the code.

```
int get_available_timer_channel();
```

```
int get_available_timer_channel() {

   if( !(CCIE & TA0CCTL0) ){
      return 0;
   }

   if( !(CCIE & TA0CCTL1) ){
   return 1;
   }

   if( !(CCIE & TA0CCTL2) ){
   return 2;
   }

   return -1;

}
```

QUESTION 9. **(10 points)**

ARM 32-bit microcontrollers have a 32-bit CPU and a 32-bit memory address. The 32-bit address can reference 2^32 = 4 GB of memory, which is far more memory than a microcontroller usually has. Therefore, there is an abundance of unused addresses, which has inspired the bit-banding technique.

Modifying a single bit in the memory takes three assembly instructions, e.g. (load-OR-store to set a bit). There is interest in making this type of operation faster since it's used frequently. The bit-banding technique takes unused addresses and maps them to bits of a variable. As shown below, the variable Data is at address 100. Accessing address 100 references the whole 8 bits in Data. The bit-banding technique takes the non-used addresses in the 2000 range and maps them to individual bits of Data. That is, address 2000 maps to bit #0 of Data, address 2001 maps to bit #1 of Data, etc. With this technique, the load-ORstore operation to modify a bit can be reduced to a store operation. Writing 1 to address 2000 results in writing 1 to bit #0 of Data, leaving the other bits unchanged.

```
         Data:  __  __  __  __  __  __  __  __
Address: 100     #7  #6  #5  #4  #3  #2  #1  #0

Address:        2007           ...           2000
```

Part a) Write a piece of code that reads bit #4 of Data (into the variable temp) using the bit-banding technique.

```
unsigned int* ptr = (unsigned int*)0x2004;
temp  = *ptr;
```

Part b) Write a piece of code that sets bit #5 of Data using the bit-banding technique.

```
unsigned int* ptr = (unsigned int*)0x2005;
*ptr = 1;
```

Part c) Write a piece of code that counts how many bits in Data are equal to 1, using the bit-banding technique.

```
unsigned int* ptr = (unsigned int*)0x2000;
unsigned int counter = 0;
unsgined int i;

for (int i = 0; i < 8; i++){
   counter += ptr[i];
}
```