Jonathan W. Valvano Solution November 18, 2011, 2:00pm-2:50pm.

(5) Question 1. The starter file for Lab 6 has a 4.7 μF capacitor from 3.3V to ground. This capacitor is located after the regular; refer to the data sheet of your regulator (LM2937-3.3).

> **A)** The regulator needs capacitance at its output in order to stabilize the +3.3V supply. The manufacturer of each regulator will suggest appropriate capacitance values and capacitor types at its input and output.

(40) Question 2. The goal is to receive synchronous serial data using edge-triggered interrupts.

(5) Part a) `AddPointerFifo(SS, 64, unsigned char, 1, 0)`

FIRST SOLUTION IS SIMPLE: 8 interrupts on clock then put one byte into FIFO

(20) Part b) Show the initialization code that configures PD2, PD1, and PD0.

```
void SS_Init(void){
  SYSCTL_RCGC2_R          |= 0x0008;
  SSFifo_Init();
  GPIO_PORTD_DIR_R        &= ~0x07; // inputs
  GPIO_PORTD_DEN_R        |= 0x07;  // enable
  GPIO_PORTD_IS_R         &= ~0x04; // edge
  GPIO_PORTD_IBE_R        &= ~0x04; // not both
  GPIO_PORTD_IEV_R        |= 0x04;  // rise
  GPIO_PORTD_IM_R         |= 0x04;  // arm
  NVIC_PRI0_R             = (NVIC_PRI0_R&0x00FFFFFF)|0x00000000;
  NVIC_EN0_R              |= 8;
// initialization of variables

  EnableInterrupts(); }
```

(15) Part c) Show the edge-triggered interrupt service routine.

```
void GPIOPortD_Handler(void){ // rise of PD2 causes interrupt
int static Count=0;
unsigned char static Data=0;
  GPIO_PORTD_ICR_R = 0x04;    // acknowledge flag2
  if(PD0 == 0){               // PD0 low means active
    Data = (Data<<1)+PD1>>1;  // next bit
    Count++;
    if(Count==8){
      Count = 0;
      SSFifo_Put(Data);
    }
  }
}
```

SECOND SOLUTION IS MY FAVORITE: 8 interrupts on clock, rising edge on PD0 then put

(20) Part b) Show the initialization code that configures PD2, PD1, and PD0.

```
void SS_Init(void){
  SYSCTL_RCGC2_R          |= 0x0008;
  SSFifo_Init();
  GPIO_PORTD_DIR_R        &= ~0x07; // inputs
  GPIO_PORTD_DEN_R        |= 0x07;  // enable
  GPIO_PORTD_IS_R         &= ~0x05; // edge
```

```
  GPIO_PORTD_IBE_R     &= ~0x05; // not both
  GPIO_PORTD_IEV_R     |= 0x05;  // rise on PD2 and PD0
  GPIO_PORTD_IM_R      |= 0x05;  // arm both PD2 and PD0
  NVIC_PRI0_R          = (NVIC_PRI0_R&0x00FFFFFF)|0x00000000;
  NVIC_EN0_R           |= 8;
// initialization of variables

  EnableInterrupts(); }
```

(15) Part c) Show the edge-triggered interrupt service routine.

```
void GPIOPortD_Handler(void){ // rising edges of PD0, PD2 interrupt
unsigned char static Data=0;
  if(GPIO_PORTD_RIS_R&0x04){  // rise of PD2
    SSFifo_Put(Data);
    Data = 0;
  } else{                      // rise of PD0
    if(PD2 == 0){              // PD2 low means active
      Data = (Data<<1)+PD1>>1; // next bit
    }
  }
  GPIO_PORTD_ICR_R = 0x05;    // acknowledge both flag0 flag2
}
```

(10) Question 3. Calculate the **resolution**. ADC resolution is 4V/1024 = 4 mV. Gain is 100, so input resolution is 4mV/100 = 40 μV. Noise must be less than 40 μV. (or ½ resolution 20 μV)

(5) Question 4. What is the best definition for what the ADC sequencer?

C) When the ADC samples multiple channels, the sequencer determines which channels will be sampled.

(5) Question 5. What is the best definition for of a flash ADC converter?

B) An 8-bit ADC uses 256 analog comparators running in parallel so all ADC bits are determined at the same time.

(20) Question 6. Design an analog circuit with the following transfer function $V_{out} = 100*(V_{in}+0.02)$.

$V_{out} = 100*(V_{in}+0.02)$.

$V_{out} = 100*V_{in}+2$

Create a 2.00 V reference with LM4041

Vz=1.233 (1+R2/R1), (1.233 is the fixed voltage of the zener)

2 = 1.233(1+R2/R1), R2/R1= 0.622, R2=31.6kΩ, and R1=51.1kΩ

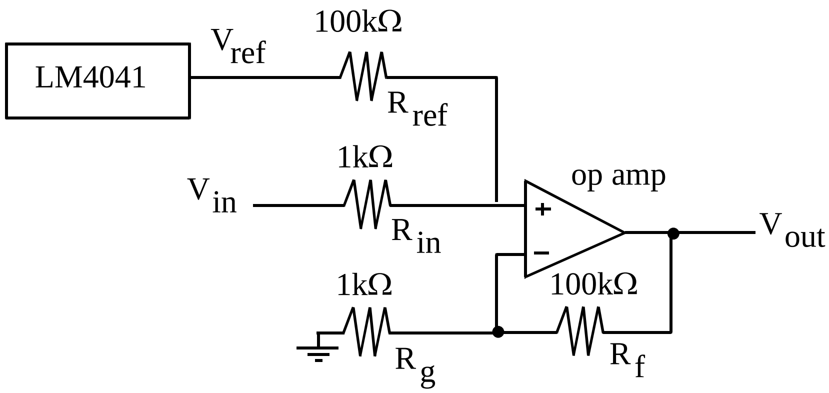
$V_{out} = 100*V_{in}+ V_{ref}$

Add ground gain of -100 to make all gains sum to 1

$V_{out} = 100*V_{in}+ V_{ref} - 100V_g$

Choose R_f to be common multiple of 1, 100 R_f = 100kΩ,

Choose other resistors to create needed gains, R_{in}=1kΩ, R_{ref}=100kΩ, R_g=1kΩ



(15) Question 7. Each SSI bit takes 1 μs, and there are 16 bits. Thus, each DAC output takes 16μs. Since there are 16 outputs in each interrupt, the ISR takes 256 μs to execute.