# EE2361 – Final Exam - 8/3/2012, 8:00-9:50

**Open book, open notes, computers OK. No communication devices or use of the internet. MPLAB allowed**

* **NAME: Patrick Barrett Email address: Barre217@umn.edu \_**

**Fosc=48MHz, unless otherwise stated.**

1. **(25 points) Use the highest resolution possible to measure the half-period of an input signal on the CCP1 pin. The range of the half-period of the signal is between 20ms .. 85ms. No polling is allowed.** 
   1. **Show your calculations  
      85 ms / (1/12Mhz) = 1020000 cycles  
      1020000/65535 = 15.6**
   2. **What is your strategy?  
      Use compare mode, switch between rising and falling edged trigger, and count overflows.**
   3. **Write the code in C. Remember: no polling.**

**#include <p18f4550.h>**

**#pragma config PLLDIV=2, CPUDIV=OSC1\_PLL2, USBDIV=2, IESO=ON, WDT=OFF**

**#pragma config BOR=OFF, PWRT=ON, LVP=OFF, FOSC=HSPLL\_HS, FCMEN=OFF, VREGEN=OFF**

**#pragma config MCLRE=ON, STVREN=ON, LPT1OSC=ON, PBADEN=OFF**

**//######### Function Declarations ################**

**void high\_isr(void);**

**void low\_isr(void);**

**void setup(void);**

**//######### Variables ################**

**//######### Interrupts ################**

**#pragma code high\_isr\_entry=8**

**void high\_isr\_entry(void){**

**\_asm goto high\_isr \_endasm**

**}**

**#pragma code low\_isr\_entry=0x18**

**void low\_isr\_entry(void){**

**\_asm goto low\_isr \_endasm**

**}**

**#pragma code**

**#pragma interrupt high\_isr**

**void high\_isr(void){**

**static unsigned int periodStart = 0;**

**static unsigned long int period = 0;**

**static unsigned char overflowCounter = 0;**

**if(PIR1bits.CCP1IF == 1){**

**PIR1bits.CCP1IF = 0;**

**CCP1CONbits.CCP1M0 = !CCP1CONbits.CCP1M0;**

**period = CCPR1 - periodStart + (((long int)overflowCounter) << 16);**

**periodStart = CCPR1;**

**overflowCounter = 0;**

**}**

**if(PIR2bits.TMR3IF == 1){**

**PIR1bits.TMR2IF = 0;**

**overflowCounter++;**

**}**

**}**

**#pragma interruptlow low\_isr**

**void low\_isr(void){**

**}**

**//######### Functions ################**

**void setup(){**

**TRISCbits.TRISC2 = 1;**

**//TMR3**

**T3CONbits.RD16 = 1;**

**T3CONbits.T3CCP2 = 1;**

**T3CONbits.TMR3ON = 1;**

**T3CONbits.T3CKPS1 = 0;**

**T3CONbits.T3CKPS0 = 0;**

**//CCP1**

**CCP1CONbits.CCP1M3 = 0;**

**CCP1CONbits.CCP1M2 = 1;**

**CCP1CONbits.CCP1M1 = 0;**

**CCP1CONbits.CCP1M0 = 1;**

**//Interrupts**

**PIE1bits.CCP1IE = 1;**

**INTCONbits.PEIE = 1;**

**INTCONbits.GIE = 1;**

**}**

**void main(void) {**

**setup();**

**while(1) {**

**}**

**}**

1. **(20 points) Write a program in C that generates a PWM signal with a frequency of 25KHz. The program should use a duty cycle of 13.42% for one minute, and then use a duty cycle of 89.85% for the rest of the program.**
   1. **Show your PWM calculations  
      (1/25KHz)/(1/12Mhz)/2 = 240 counts**

**13.42% = 32.21 counts  
89-85 = 215.6 counts**

* 1. **You are supposed to use polling on Timer 1 to generate the 1 minute delay. Show your calculations for generating a one minute delay using Timer 1.  
     60s/(1/12Mhz)/8 = 9\*10^7 counts  
     60s/(1/12Mhz)/8/65535 = 1373 overflows**
  2. **Write the C code to generate the specified PWM signal on the CCP2 pin.**

**#include <p18f4550.h>**

**#pragma config PLLDIV=2, CPUDIV=OSC1\_PLL2, USBDIV=2, IESO=ON, WDT=OFF**

**#pragma config BOR=OFF, PWRT=ON, LVP=OFF, FOSC=HSPLL\_HS, FCMEN=OFF, VREGEN=OFF**

**#pragma config MCLRE=ON, STVREN=ON, LPT1OSC=ON, PBADEN=OFF**

**//######### Function Declarations ################**

**void high\_isr(void);**

**void low\_isr(void);**

**void setup(void);**

**//######### Variables ################**

**//######### Interrupts ################**

**#pragma code high\_isr\_entry=8**

**void high\_isr\_entry(void){**

**\_asm goto high\_isr \_endasm**

**}**

**#pragma code low\_isr\_entry=0x18**

**void low\_isr\_entry(void){**

**\_asm goto low\_isr \_endasm**

**}**

**#pragma code**

**#pragma interrupt high\_isr**

**void high\_isr(void){**

**}**

**#pragma interruptlow low\_isr**

**void low\_isr(void){**

**}**

**//######### Functions ################**

**void setup(){**

**TRISCbits.RC1 = 0;**

**CCP2CONbits.CCP2M3 = 1;**

**CCP2CONbits.CCP2M2 = 1;**

**PR2 = 240;**

**CCPR2L = 32;**

**CCP2CONbits.DC2B1 = 0;**

**CCP2CONbits.DC2B0 = 1;**

**T2CONbits.T2CKPS1 = 0;**

**T2CONbits.T2CKPS0 = 1;**

**T2CONbits.TMR2ON = 1;**

**T1CONbits.RD16 = 1;**

**T1CONbits.T1CKPS1 = 1;**

**T1CONbits.T1CKPS0 = 1;**

**T1CONbits.T1OSCEN = 1;**

**T1CONbits.TMR1CS = 0;**

**T1CONbits.TMR1ON = 1;**

**}**

**void main(void) {**

**int i;**

**setup();**

**for(i = 0; i<=1373; i++){**

**while(PIR1bits.TMR1IF == 0);**

**PIR1bits.TMR1IF = 0;**

**}**

**CCPR2L = 215;**

**CCP2CONbits.DC2B1 = 1;**

**CCP2CONbits.DC2B0 = 0;**

**while(1) {**

**}**

**}**

1. **(15 Points) In a C program that interfaces a device using the SPI interface, SSPCON1 is set to 0x21. You are supposed to complete the program and transmit 0x59, then receive a byte and store in variable inpChar1, and then receive another byte and store it in variable inpChar2.**
   1. **Before writing the code, show your calculations for how long the process of sending a byte and receiving two bytes takes.**

**Each byte of send and receive: (1/12Mhz)\*8 = 666 ns**

**Total time = 666\*3 = 2 us**

* 1. **Write the C code here (only the parts doing the transmit and receive):**

**unsigned char rwspi(unsigned char spiOut){**

**SSPBUF = spiOut;**

**while(SSPSTATbits.BF == 0);**

**return SSPBUF;**

**}**

**void main(void) {**

**char inpChar1, inpChar2;**

**setup();**

**rwspi(0x59)**

**inpChar1 = rwspi(0x0)**

**inpChar2 = rwspi(0x0)**

**while(1) {**

**}**

**}**

1. **(15 Points) We would like to sample an analog input with a voltage range of 1.3v..4.1v using an 18F4550 PIC device running at Fosc=48MHz. Our sensor Rs=4.5KΩ and the maximum temperature would be 70°C.**
   1. **Show your calculations for TAD and TACQ.   
        
      Tc = -(25pF)(1Kohms + 2Kohms + 4.5Kohms)ln(1/2048) = 1.43 us  
      Tcoff = (70C-25C)(0.02 us/C) = 0.9 us  
      Tacq = 1.43 us + 0.9 us + 0.2 us = 1.53 us  
      TAD = FOSC/4 = 83.3 ns  
      20TAD > Taq**
   2. **What is the sampling resolution (in terms of volts) if we connect the VREF+ and VREF- pins to 4.1v and 1.3v respectively?  
        
      (4.1V - 1.3V) / 2^10 = 2.73 mV**
   3. **How many instruction cycles does it take to receive only one sample (including acquisition time)?  
        
      TACQ + 11 TAD + Discahrge&Load  
       = 20 TAD + 11 TAD + 1 TAD  
       = 32 \* 83.33 ns = 2.66 us**
2. **(15 Points) Use the CORDIC method to calculate Sin(56) with an accuracy of 2-3. Clearly show initial values, intermediate steps and what variable will have the final result. How far off is your answer compared to what your calculator shows for Sin(56)? (hint: make sure your calculator is treating 56 as degrees, and not radians)  
     
   Initial:  
   X = K; Y = 0; Z = 56  
   Round 1:  
   X = 0.607; Y = 0.607; Z = 11   
   Round 2:  
   X = 0.304; Y = 0.910; Z = -15.56   
   Round 3 ( Result ):  
   X = 0.531; Y = 0.835; Z = =1.52  
     
   Error:  
   sin(56) = 0.829; Difference of 0.006.**
3. **(10 Points) Assuming that variable *coeff*  is at FSR2 offset 5 on the stack, and variable *grade* is at offset 7, write an inline assembly code to perform *grade = coeff \* grade*. Both variables are of type *char* (i.e., only the lower 8 bits of the multiplication are going to be stored back).  
     
     
     
     
   movlw 0x05  
   movff PLUSW2, PRODH   
   movlw 0x07  
   movff PLUSW2, WREG  
   mulwf PRODH  
   movlw 0x07  
   movff PRODL, PLUSW2**