ECE 3724/CS 3124 Test #4 – Spring 2005- Reese.   Name:_____

You may use a calculator and the provided reference materials. If a binary result is required, give the value in HEX. For any required I2C functionality, use subroutine calls *to i2c_start(),i2c_rstart(), i2c_stop, i2c_put(char byte), i2c_get(char ackbit) ,char i2c_put_noerr (char byte)*. If you use *i2c_put*, you must pass in as an argument the byte that is to be written to the I2C bus. If you use *i2c_get*, you must pass in an as argument the bit value to be sent back as the acknowledge bit value.

You also have available the functions:
  i2c_memwrite(char i2caddr, unsigned int addr, volatile unsigned char *buf)
 i2c_memread(char i2caddr, unsigned int addr, volatile unsigned char *buf

Part I: (72 pts)
   a. (15 pts)Assume you have a 32.768 KHz clock as the clock source for TIMER1 with a prescale of 1 (timer1 overflows every 2 seconds). Every 30 seconds, do an ADC conversion and store the upper 8-bits in a buffer. Once the buffer has 64 entries write this to the 24LC515 EEPROM (A1, A0 on EEPROM both tied low). Do all of this work in an ISR, and use the *i2c_memwrite()* function to write the data to the EEPROM. Write the ISR that accomplishes this; do not assume that Timer1 is the only enabled interrupt. Assume the ADC has already been configured, and that it is left justified. Assume timer1 is already configured and that its interrupt is enabled. Each time a block of 64 bytes is written, increment the address passed to *i2c_memwrite()*. When the EEPROM fills up, disable the TIMER1 interrupt.

b. (12 pts) Using the I2C functions, write a *C* code fragment that loops, polling a 24LC515 serial EEPROM for end-of-write. Assume that both the A1, and A0 pins on the EEPROM are tied high. The loop should not exit until the 24LC515 indicates the current write operation is finished. The function *char i2c_put_noerr(char byte)* is needed as it returns the value of the acknowledge bit returned after the byte write to the I2C bus.

c. (6 pts) Assume FOSC = 40 MHz. Using the PWM module, give the PR2, PRE, and CCPR1L values for a 6 KHz square wave with a 20% duty cycle.

d. (12 pts) In the diagram below, a three position dip switch is connected to the lower three bits of PORTB. Asssume the weak pullups have been enabled on PORTB. Thus when all switches are CLOSED, the three bit value is read as "000" = 0; when all switches are OPEN the three bit value is read as "111" = 7. Assume TIMER2 has been configured to generate a periodic interrupt. Write an ISR so that on each TIMER2 interrupt, read the switches and cause the DAC output voltage to be the values shown below. Use the individual I2C function calls to communicate with the DAC; you cannot use the 'update_dac()' function. When you read PORTB, you CANNOT assume any values for bits RB3-RB7. Do not assume that TIMER2 is the only enabled interrupt.

Dip switch value	DAC Vout
"000",	0 V;
"001",	1/8 VREF
"010",	2/8 VREF
"011",	3/8 VREF
....	
"111",	7/8 VREF.
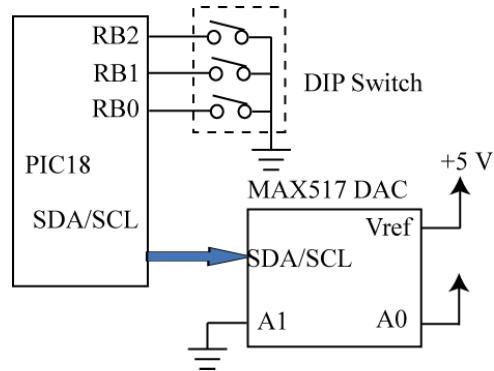
e. (6 pts) How many TIMER1 tics is contained in 1 ms given an FOSC = 40 MHz and a timer prescale of 2?

f. (7 pts) Write C configuration code that configures the ADC to select channel 4, left justified result, an ADC clock of FOSC/16, turns the ADC on, has VREF+=AN3, VREF- = AN2, and the other PORTA inputs as being analog inputs. Use individual bit assignments for clarity, do NOT assign 8-bit values to ADCON0 and ADCON1.

g. (6 pts) Assume the code used in lab to measure the pulse width of a pushbutton switch. On the falling edge (pushbutton pressed), the capture register captures the hex value 0x0700 from timer1. On the rising edge (pushbutton released), the capture register captures the value 0x0200 from Timer1, with TWO timer1 overflows between the falling and rising edge captures. Assuming a Timer1 prescale of 2, an FOSC = 40 MHz, and using the internal clock, how long is the pulse width in microseconds?

h. (8 pts) Explain EITHER the operation of a 4-bit successive approximation ADC or a 4-bit flash ADC. For both ADCs, use Vin = 1.7 V and Vref = 4 V. If you explain the successive approximation ADC, you have to give the internal VDAC voltage used at each comparison step, and list all steps. If you explain a flash ADC, you have to give the number of comparators and resistors, the output value (1 or 0) of all comparators. For either ADC, you have to give the final 4-bit output code.

Part II: (28 pts) Answer 7 out of the next 9 questions. Cross out the 2 questions that you do not want graded. Each question is worth 4 pts.

1. A 7-bit DAC has an input code of 0x3B and a VREF = 10 V. What is the output voltage?

2. There are two important pieces of information specified in the first byte of any I2C transaction – what are they?

3. Draw a picture that shows two devices connected to the I2C bus of the PIC18. Be sure to show everything!

4. How would PWM be used to control the illumination of an LED? Show waveforms for 25% maximum illumination, 50% maximum illumination, and 75% maximum illumination.

5. Draw pictures that show how a "1" is distinguished from a "0" using SPACE-WIDTH encoding. Your pictures must be VERY clear in how the "1" and "0" bits are different – you can choose the actual ratios of the waveforms yourself.

6. In class and in the notes, we discussed the proper way to WRITE a 16-bit value to timer1 so that it is updated correctly. Show C code that updates TIMER1 with the value from '*int my_value*' correctly.

7. Write C code that configures the capture system to capture every falling edge on CCP1, and to use TIMER3 as the timer register for CCP1 pin captures.

8. Draw pictures that show how a "1" is distinguished from a "0" using BIPHASE encoding. Your pictures must be VERY clear in how the "1" and "0" bits are different.

9. Write a C code fragment that performs an acknowledge condition on the I2C bus (do not use the **i2c_ack(char ackbit)** function, I want to know what is inside the **i2c_ack** function!).