

ECE 3724 Summer 2006 Test #4 –Reese

Net ID: _____ (no names, please)

You may use only the provided reference materials. You may use a calculator, either a four-function or a scientific calculator. You may not use a programmable calculator. The test is worth 100 pts, you are given 1 pt for free. For any required I2C functionality, use subroutine calls *i2c_start()*, *i2c_rstart()*, *i2c_stop*, *i2c_put(char byte)*, *i2c_put_noerr(char byte)*, *char i2c_get(char ackbit)*. If you use *i2c_put*, you must pass in as an argument the byte that is to be written to the I2C bus. Recall that *i2c_put_noerr* returns the value of the ACK bit that is returned by the slave device, while *i2c_put* does a software reset if the slave device returns an ACK of '1'. 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 *DelayUs()* and *DelayMs()* functions available. Show all your work in any computations done or formulas used.

Part I: (80 pts)

- a. (10 pts). Write a function called *void write_rdy()* that does not returns until the 24LC515 serial EEPROM has finished with its last write. You must poll the device for 'end-of-write' to determine when the device has finished; the *i2c_put_noerr()* function will come in handy for this. Assume that both the A0, and A1 lines of the EEPROM are tied low.

- b. (5 pts) The function below is communicating with a 24LC515 Serial EEPROM. Give the range of addresses that are read in the EEPROM from the viewpoint of the 64K address space (the range must be within the range 0x0000 – 0xFFFF).

```
myfunc() {  
  
    int i;  
    char x;  
  
    i2c_start();  
    i2c_put(0xAE);  
    i2c_put(0x00);  
    i2c_put(0x00);  
    i2c_restart();  
    i2c_put(0xAF);  
    i = 0;  
    do {  
        x = i2c_get(0);  
        i++;  
    }while (i < 0x8000);  
    i2c_stop();  
}
```

- c. (15 pts) Write a C function named `char adc_check()` that performs conversions on channels AN1, AN2, and AN3 and returns a '1' if any of them are outside of the range 2V to 4V. Assume a VREF of 5 V, that the ADC is configured for left justification, and that we are only interested in the upper 8-bits of the conversion. Do not assume that a particular channel is selected on entering the function. Delay for 20 us before beginning a conversion after selecting a channel.

- d. (15 pts) Write a function called `dac_step(void)` that tests the MAX517 DAC by stepping its output value from 0 V to its full range in steps of 1 least significant bit. Assume that both AD1 and AD0 of the MAX517 are tied high.

- e. (15 pts) The delay function **DelayMs(char x)** that we have used in labs is a software delay loop function. Write an equivalent function that uses Timer1 and compare mode (CCPR1) to accomplish the same thing. Follow the approach of using a loop that zeros Timer1, then writes a value to CCPR1 that is equivalent to a 1 ms delay, then waits for the CCPR1IF flag to be set. To wait for **x** ms, execute this basic loop **x** times. Assume an FOSC of 40 MHz. In addition to the code, show calculations that determine the value to write to the CCPR1 register – you pick the prescale value that you will use for this calculation. You do not have to write any configuration code for compare mode, assume that this has already been done.

- f. (10 pts) Write C code that will configure the CCP1 output and the PWM module of the PIC18 to generate a square wave with 250 μ s period and a 40% duty cycle assuming an FOSC of 30 MHz. Show the calculations first, then show the code.
- g. (10 pts) Explain EITHER the operation of a 3-bit successive approximation ADC or a 3-bit flash ADC. For both ADCs, use $V_{in} = 1.7$ V and $V_{ref} = 6$ 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 3-bit output code.

Part II: (20 pts) Answer 5 out of the next 8 questions. Cross out the 3 questions that you do not want graded. Each question is worth 4 pts.

1. Assume an FOSC of 20 MHz. What is the maximum length (in milliseconds) of a periodic interrupt generated using Timer2?
2. What does the least significant bit of the first byte of every I2C transaction signify?
3. Write a C code fragment that returns the upper 8-bits of the PIC18 ADC result value in the char variable *c* regardless of whether the ADC is configured as left justified or right justified.

7. If the V_{ref} of the PIC18 is 4.1 V, and I read a 10-bit code of 0x200, what is the input voltage value?

8. For an FOSC of 6 MHz, what is the fastest ADC clock configuration (other than the internal oscillator) that can be selected and still not violate the 1.6 μ s minimum period constraint? Show your work.