

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 is 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)*, *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. 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: (75 pts)

- a. (15 pts). The code fragment below performs I2C operations to a 24LC515 serial EEPROM. Answer the questions in the box in the right

```
char buf[3];
```

```
i2c_start();
i2c_put(0xA4);
i2c_put(0x78);
i2c_put(0x4A);
i2c_rstart();
i2c_put(0xA5);
buf[0]=i2c_get(0);
buf[1]=i2c_get(0);
buf[2]=i2c_get(1);
i2c_rstart();
i2c_put(0xAC);
i2c_put(0x40);
i2c_put(0x01);
i2c_put(buf[1]);
i2c_put(buf[0]);
i2c_put(0x30);
i2c_stop();
```

*bsel A1 A0*  
*0xA4 = 1010 0 1 0 0*  
*so, A1 = VDD, A0 = GND*

*Address is 0x784A, in lower 32K block as bsel = 0, so actual address 0x784A.*

*Read locations 0x784A, 0x784B, 0x784C*

*bsel A1 A0*  
*0xAC = 1010 1 1 0 0, block select = 1*  
*address is 0x4001 in upper 32K block, so actual address is 0xC001*

*content of location 0x784B written to location 0xC001*

*content of location 0x784A written to location 0xC002*

*value 0x30 written to location 0xC003*

1. What values are the A1, A0 lines on the EEPROM tied to (give as either VDD or GND for each)?

2. What locations in the SERIAL EEPROM are read? Give answers in hex.

3. What locations in the SERIAL EEPROM are modified? Your answer(s) must be in the form of either "The value 0x?? is written to location 0x????", or "The content of location 0x???? is transferred to location 0x????". In each case, 'location 0x????' is a location in the EEPROM. Give all values in hex.

- b. (15 pts) Write a C function named `char adc_check()` that performs a PIC18 A/D conversion on both the AD0 and AD1 channels. The function return value is:
- '1' if AD0 > AD1 by 1.0 V or more
  - '2' if AD1 > AD0 by 1.0 V or more
  - '0' if  $|AD1 - AD0| < 1.0$  V.

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.

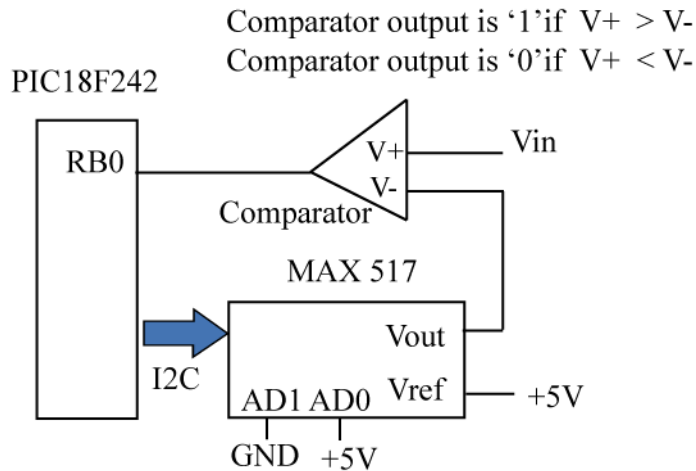
```
char adc_check() {
    unsigned char ad0_value, ad1_value;
    //select channel 0
    CHS2 = 0; CHS1 = 0; CHS0 = 0;
    DelayUs(20);
    GODONE = 1;
    while(GODONE); //do conversion
    ad0_value = ADRESH; //read the result
    //select channel 1
    CHS2 = 0; CHS1 = 0; CHS0 = 1;
    DelayUs(20);
    GODONE = 1;
    while(GODONE); //do conversion
    ad1_value = ADRESH; //read the result
    if ( (ad0_value > ad1_value) && ←
        (ad0_value - ad1_value) > 51) return(1);
    if ( (ad1_value > ad0_value) &&
        (ad1_value - ad0_value) > 51) return(2);
    return(0);
}
```

*Transform 1 V into an 8-bit code using a Vref of 5 V*

$$1V/5V * 256 = 51.2 = 51$$

*Comparing ad0\_value against ad1\_value before doing the subtraction as the subtraction is only valid if ad0\_value is greater than ad1\_value (if not, then will produce a negative number which is invalid in this case)*

- c. (15 pts) Reference the diagram below. Write a C function named **char range\_check()** that returns a '1' value if  $3.5V \leq V_{in} < 4.5V$  using the MAX517 DAC and the RB0 input as shown below. Use the I2C functions to communicate with the MAX517 DAC. The comparator output is '1' if  $V_{+} > V_{-}$ ; the comparator output is '0' if  $V_{-} < V_{+}$ .



*To solve this problem, need to set  $V_{-}$  of the comparator to 4.5V to compare against  $V_{in}$ , then set  $V_{-}$  of the comparator to 3.5V.*

*4.5V as 8-bit code:  $4.5/5.0 * 256 = 230$*

*3.5V as 8-bit code:  $3.5/5.0 * 256 = 179$*

*This solution is only accurate to the accuracy level provided by the MAX517 DAC and rounding of the 8-bit calculations. Not really possible to check for equality to a particular voltage.*

```

char range_check() {
    //first, set DAC Vout to 4.5V
    i2c_start();
    i2c_put(0x5A); // 0101 1 0 1 0 MAX517 address byte
    i2c_put(0x00); //command byte of 0x00 says to do conversion
    i2c_put(230); //8-bit value corresponding to 4.5 V
    i2c_stop();
    if (RB0) return(0); // if RB0 =1, then Vin > 4.5, out of range, return 0 !

    //next set DAC Vout to 3.5 V
    i2c_start();
    i2c_put(0x5A);
    i2c_put(0x00);
    i2c_put(179); //8-bit value corresponding to 3.5 V
    i2c_stop();
    if (!RB0) return(0); // if RB0 =0, then Vin < 3.5, out of range, return 0 !

    return(1); // Vin must be between range of 3.5 V to 4.5 V
}

```

- d. (10 pts) Write C code that will configure the PIC18F242 TIMER2 to generate a periodic interrupt every 1 ms assuming an FOSC of 20 MHz. The interrupt must be fully enabled! Show the calculations first, then show the code.

$PR2 = (1 \text{ ms} / [4/20 \text{ MHz} * PRE * POST]) - 1$ .  
 One solution is  $PRE = 16$ ,  $POST = 2$ ,  $PR2 = 155$

*C code:*

```
TOUTPS3 = 0; TOUTPS2 = 0; TOUTPS1=0; TOUTPS0 = 1; //POSTSCALE = 2
PR2 = 155;
T2CKPS1 = 1; //PRESCALE = 16
TMR2ON = 1; // turn on the timer
IPEN = 0;
TMR2IF = 0; TMR2IE = 1;
PEIE = 1; GIE = 1; //enable the interrupt
```

- e. (10 pts) Assume that TIMER2 has been configured to generate a periodic interrupt and that a low-true pushbutton switch is tied to RB1 (the RB1 input is '0' if the pushbutton is pressed). Write an ISR that will set the semaphore BUTTON\_PRESSED to a '1' and disable the TIMER2 interrupt if the RB1 input is detected as low for five CONSECUTIVE TIMER2 interrupt periods.

```
char count;
interrupt my_isr() {
    if (TMR2IF && TMR2IE) {
        TMR2IF = 0; //clear interrupt flag
        if (RB1) count = 0; //button is not pressed, reset count
        else {
            //RB1 == 0, pushbutton is pressed
            count++;
            if (count == 5) {
                TMR2IE = 0; //disable the interrupt
                BUTTTON_PRESSED = 1; //set the semaphore variable
            }
        } //end if(TMR2IF...)
    } // end my_isr()
```

- 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 500 us period and a 25% duty cycle assuming an FOSC of 20 MHz. Show the calculations first, then show the code.

$PR2 = (500 \text{ us} / [4/20 \text{ MHz} * PRE]) - 1$ . Use  $PRE = 16$ ,  $PR2 = 155$ .  
 $CCPR1L = (PR2+1) * 0.25 = 156 * 0.25 = 39$

*C Code:*

```
T2CKPS1 = 1; PR2 = 155; CCPR1L = 39;
CCP1M3 = 1; CCP1M2 = 1; //select PWM mode
CCP1CON = CCP1CON & 0x0F; // Clear CCP1CON[5:4], duty cycle bits
TRISC2 = 0;
TMR2ON = 1;
```

Part II: (24 pts) Answer 6 out of the next 8 questions. Cross out the 2 questions that you do not want graded. Each question is worth 4 pts.

1. How many clock cycles will an N-bit successive-approximation ADC require for a conversion? How many clock cycles will an N-bit flash ADC require for a conversion?

***Successive approximation takes N clock cycles (one for each bit)  
Flash A/D takes only one clock cycle.***

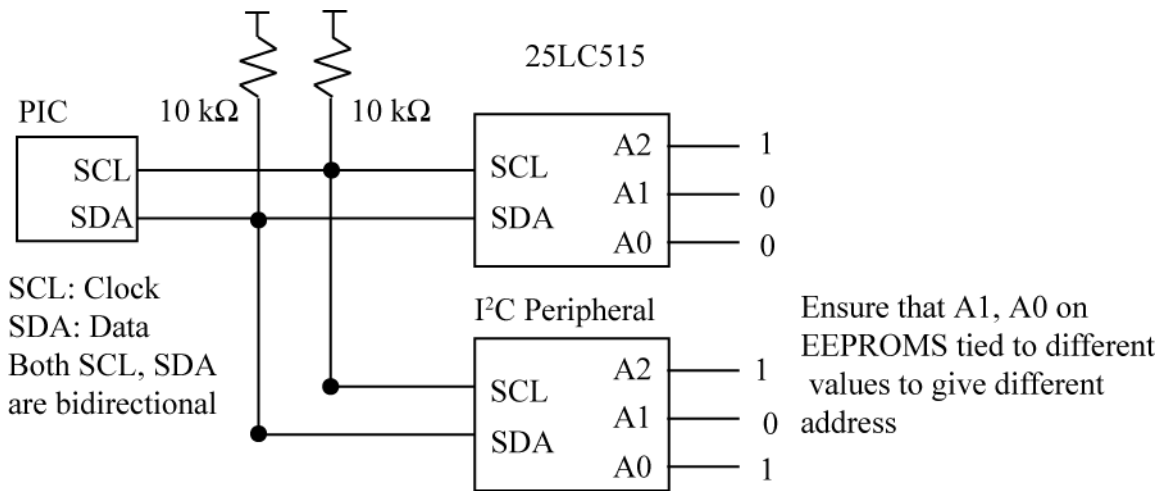
2. Write a C code fragment that will poll for end-of-write on the LC515 serial EEPROM. Use the I2C functions and assume that both A1, A0 on the EEPROM are tied low.

```
char ack_bit;  
do {  
    i2c_start();  
    ack_bit = i2c_put_noerr(0xA0); //try a write command, see what the  
    i2c_stop(); // returned ack bit is  
while(ack_bit);
```

3. How many comparators are used in a 4-bit flash ADC? How many resistors are used in the resistor ladder of a 4-bit flash ADC?

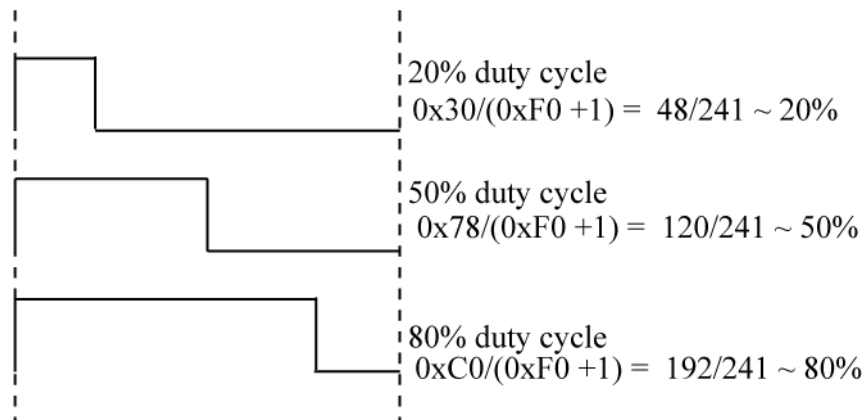
***2<sup>4</sup> resistors (16 resistors)  
2<sup>4</sup> - 1 comparators (15 comparators)***

4. Draw a diagram that shows a PIC microcontroller connected to two different LC515 serial EEPROMs devices. Label all of the pins.



5. On the PIC18, assume a PWM output is being generated using the PWM module. If the PR2 register contains a value of 0xF0, draw the PWM waveforms for CCPR1H values of 0x30, 0x78, and 0xC0. Your three waveforms must be drawn to the same scale and must illustrate the approximate duty cycles of the waveforms.

Duty cycle is  $CCPR1L / (PR2 + 1)$



6. Assume an I2C write transaction that consists of the I2C address byte, followed by four data bytes. What is the total number of I2C bit times involved? Assume that the start and stop conditions each count as one bit time. Show work done in these computations.

$$1 \text{ stop} + 1 \text{ start} + 5 \text{ bytes} * (8 \text{ bits} + \text{ack}) = 1 + 1 + 5 * 9 = 1 + 1 + 45 = 47 \text{ bit times}$$

7. Given a 9 bit DAC, and a reference voltage of 4.096 V, what is the expected output voltage change on the DAC output if the input code is changed by 1 Least Significant Bit? Give the answer in **millivolts**.

$$1/2^9 * 4.096 = 1/512 * 4.096 = 0.008 = 8 \text{ mV}$$

8. For an FOSC of 4 MHZ, what is the fastest ADC clock configuration (other than the internal oscillator) that can be selected and still not violate the 1.6 us minimum period constraint? Show your work.

$$1/1.6 \text{ us} = 625,000 = 625 \text{ KHz}$$

$$4 \text{ MHz}/4 = 4,000,000/4 = 1,000,000 = 1000 \text{ KHz (too fast)}$$

$$4 \text{ MHz}/8 = 4,000,000/8 = 500,000 = 500 \text{ KHz (this is ok)}$$