EXAM #4

- You may use a non-programming calculator only. You may use only the provided reference materials.
- When 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)`, `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.
- Unless stated otherwise, all multi-byte data values are stored in little-endian ordering.
- Please note the relative value of each problem in the table below.
- Answers should be clearly indicated. Placing them in a BOX would be ideal.
- Be as neat and well organized as possible. This is in your grade’s best interest.
- If you need additional space to work, do so on the backside of the page. Make sure it is clear where your work continues.
- Absolutely NO cheating is allowed. If you are caught in the attempt of, the act of, or the past action of academic dishonesty, you will receive the maximum punishment allowed by University policy.
- No panicking allowed!

<table>
<thead>
<tr>
<th>Page</th>
<th>Maximum</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6-7</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

MY CALCULATOR MODEL: ___________________________________________

«First_Name» «Last_Name» «User_ID»
Part I: (79 pts)

a. (9 pts) Complete the C code loop that causes the MAX 517 to output a voltage of VREF (or as close as you can get), and decrease all of the way to 0 V in steps of 1 LSb (least significant bit). You must use I²C functions when causing the MAX517 to perform a conversion. Assume A1 is wired to +5V and A0 is wired to GND.

```c
char dac_value;
dac_value = 0xFF; // initial value is as close to VREF //as possible
do {
```}

} while(dac_value != 0xFF); //exit when we wrap around back

b. (8 pts) Write a C code function `char getdata()` that waits for data to become available in a circular buffer named `buf`, takes data out of the buffer and returns it to the caller. Assume the buffer has a maximum of `BUFSIZE` characters, and pointers named `tail` and `head`. The `head` pointer is used for placing data into the buffer. Assume an ISR is responsible for placing data into the buffer.

c. (6 pts) Write a C code fragment to go in the beginning of `main()` to detect a software reset condition and prints the message ‘Software Reset’. Set or clear the appropriate status bit so that a software reset is not falsely detected on the next non-software-reset.
For the next several questions, consider the following C code for an ISR and main loop.

```c
interrupt my_isr() {
    if (INT1IF) {
        INT1IF = 0;
        RB5 = 1;
    }
}//end my_isr()

main() {
    TRISB5 = 0; TRISB1 = 1;
    RB5 = 1;
    INTEDG1 = 0; IPEN = 0; INT1IF = 0; INT1IE = 1; GIE = 1; PEIE = 1;
    while (1) {
        RB5 = 0;
    } // end while()
} //end main()
```

d. (6 pts) Describe the response of the system that runs the code above. Be specific in describing what stimuli cause what responses.

e. (6 pts) What determines the HIGH pulse width time in the interrupt code above? Be as specific and inclusive as possible.

f. (6 pts) What determines the LOW pulse width time in the interrupt code above? Be as specific and inclusive as possible.
g. (10 pts) Your PIC18 has “normally-open” pushbuttons connected to port pins RB1 and RB2 with internal pullups enabled. Assume both pins have been configured to generate rising-edge-triggered, high-priority interrupts. Write an ISR that increments a variable named `edge_count` by 2 if the INT1 interrupt occurs, or increments `edge_count` by 1 if the INT2 interrupt occurs. After an interrupt occurs, disable that interrupt using its enable flag, clear the interrupt flag bit, and set the semaphore variable `switch_pressed` to a ‘1’. Don’t debounce the switches.

h. (8 pts) The PIC18 automatically saves the registers BSR, W, and STATUS into the shadow registers when an interrupt occurs for either a low or high priority interrupts. Assume that you had a system with both low and high priority interrupts. What is the first thing that your low priority interrupt service routine should do with the values saved in the BSR, W, STATUS shadow registers? Explain why. Is this also necessary for the high priority ISR? Why or Why not?
i. (10 points) Write C code that will configure the A/D module for left justification, AN2, AN1 as analog inputs, AN3 as \( V_{REF+} \). VSS as \( V_{REF-} \). Use the internal FOSC clock, and configure the A/D clock such that it meets the minimum clock period constraint of 1.6\( \mu \)s assuming an FOSC of 12 MHz. (Use the fastest internal clock choice that meets this constraint). For the configuration code, use individual bit names ADCS2, ADCS1, ADCS0, ADON, ADFM, PCFG3, PCFG2, PCFG1, PCFG0. You do not have to configure the channel select bits, that is done next.

j. (10 points) Write a function called char analog_sum() that will perform a conversion on each analog input (AN2, AN1) and return the sum of these values as a char value. Do not let the sum exceed 255 (0xFF) (Hint: You will need to use an unsigned int variable to hold the sum, then clip this to 255). When changing A/D channels, use the DelayUs function to delay 20 \( \mu \)s to give the A/D input a chance to settle. Since you don’t know how often this function will be called, use this delay before starting each conversion.
Part II: (21 points) Answer the 7 out of the next 9 questions. Each question is worth 3 points.

YOU MUST CLEARLY CROSS OUT THE QUESTIONS YOU DO NOT WANT GRADED!!!!

a. What is the SPBRG value for a baud rate of 19200 assuming an Fosc of 15 MHz and high speed mode?

b. Can you hook the PIC TX and RX pins directly to pins 2, 3 of the DB9 connector to implement the serial port connection to the PC? Why or why not? Explain.

c. Your 7-bit DAC has an input code of 0x2B and a VREF = 8 V. What is the output voltage?

d. There are two important pieces of information specified in the first byte of any I²C transaction – what are they? Where are they located relative to each other?

e. 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!).

```c
void i2c_ack(ackbit) {

} // end i2c_ack
```
f. Explain EITHER the operation of a 4-bit successive approximation ADC or a 4-bit flash ADC. For both ADCs, use \( V_{in} = 1.7 \text{ V} \) and \( V_{ref} = 4 \text{ V} \). 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.

g. How many bit times are there in the \( \text{l}^2\text{C} \) transaction to the MAX517 DAC for a conversion? Count the start and stop conditions each as one bit time.

h. Assume the PIC18 A/D is configured with a \( V_{ref^+} = 4 \text{ V} \), and a \( V_{ref^-} = 0 \text{ V} \), and that two successive reads of the ADC are done, with the first value returning 0xA0, and the next value returning 0x1A (only the 8 MSb are read). What is the change in voltage?

i. In asynchronous serial communication, why can’t we send large numbers of bits at a time? (RS232 uses a maximum of 10 bits).
"DEJA VU" BONUS QUESTION (5 points)
You must answer this question completely correct to receive the bonus points.

After the execution of ALL of the C code below, fill in the memory location values. Assume little-endian order for multi-byte values.

```c
signed char a[2];
signed int b;
signed long c;
signed int *ptrb;

a[0] = 3;
a[1] = -7;
ptrb = &a[0];
b = *ptrb + 2;
ptrb++;
c = *ptrb >> 1; // Computes *ptr >> 1 as int, then converts to a long
```

Location   Contents (MUST be given in hex)

0x1A0    
0x1A1    
0x1A2    
0x1A3    
0x1A4    
0x1A5    
0x1A6    
0x1A7    
0x1A8    
0x1A9    

CBLOCK 0x1A0
         a:2, b:2, c:4, ptrb:2
ENDC