ECE 3724 Fall 2005 Test #3 – Jones / Reese (**circle one to indicate your section**)

Net ID: __________________________ (no names, please)

You may use only the provided reference materials. All figures are on the last pages. 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.

Part I: (75 pts)

a. (5 pts) Write *C* code that configures PORTB for the IO shown in the figure for problem (a) on the Figure sheet. The internal weak pullup must be enabled. Do not assume any default bit values.

b. (15 pts) Assuming the IO configuration of the previous problem, write a *while(1){}* loop that implements the LED/Switch IO state machine shown for problem (b) in the figures. Either use a *switch()* statement approach or a *if-then-els*e approach. Assume you have available the *DelayMs()* function for blinking the LED; you do NOT have to include debounce delays for the switch input.

**c.** ( 20 pts) For the LED/Switch configuration shown in Figure (c) implement the actions listed in the figure in an *interrupt driven manner*. Divide your solution into two code segments -- an ISR, and *main()* code that includes the declarations of any variables used by the ISR initialization code for the interrupt system, initialization code for the ports, and code that initializes the LEDs to A=ON, B=OFF, C=OFF. You do not have debounce the switch inputs in your ISR. You cannot have any delay code in the ISR that waits for input. Your infinite loop in *main()* has to be an infinite loop that is an EMPTY infinite loop *while(1){};* your ISR has to do all of the work. **Your code must compile cleanly!!!!!!!**

1. (14 pts) ISR code (do not worry about debouncing the switch inputs).

2. (6 pts) *main()* code, your while(1){} has to be empty; the ISR must do all of the work.

d. (7 pts) Assume an asynchronous serial channel with a data format of 1 start bit, 8 data bits, and 1 stop bit between characters. What is the minimum time in microseconds that it takes to send 20 characters at 57,600 baud?

e. (7 pts) Write *C* code that implements the *char getch(void)* function that waits for a character to be available in the USART and then returns that character from the serial port. *No interrupts are enabled*.

f. (9 pts) Assume the definitions of a circular buffer that we have used in lab (i.e, the head pointer is used to place data into the buffer, the tail pointer is used to take data out of the buffer, the buffer is empty when head is equal to tail, and that pointers are incremented and wrapped before used to access the buffer).

f1. From figure F, how many characters are currently ***available*** in the buffer? (this is not the total number of locations in the buffer) _____________

f2. From figure F, what character is returned if the buffer is read?

_______________

f3. From figure F, what location is modified if one character is written to the buffer?

___________

g. (12 pts) Given the code below and Figure (g), answer the questions below. The instruction code produces a square wave on RB5 as the *main{} while(1)* loop sets RB5 high and the ISR resets RB5 low.

For the following execution time components, check whether or not they are part of the LOW PULSE WIDTH TIME or HIGH PULSE WIDTH TIME of the square wave on RB5 (or check both if you think they belong to both):

| | HIGH PW time | LOW PW time |
|---|---|---|
| a. execution time for instruction sequence A | ____________ | ____________ |
| b. execution time for instruction sequence B | ____________ | ____________ |
| c. exeuction time for instruction sequence C | ____________ | ____________ |
| d. execution time for instruction sequence D | ____________ | ____________ |
| e. interrupt service routine entry time | ____________ | ____________ |
| f. interrupt service routine exit time | ____________ | ____________ |

```
interrupt my_isr() {
  if (INT1IF) {
      INT1IF = 0;

     //instruction
     //....sequence
     //......C (not shown)
     RB5 = 0;
     //instruction
     // .....sequence
     // ........D (not shown)
  }
}//end my_isr()

main() {


//....code that initializes INT1 to be RISING EDGE triggered
// ...and enabled, and RB5 to be initially low
while (1) {
    //...instruction
    //.....sequence
    //.......A (not shown)
    RB5 = 1;
    //...instruction
    //     sequence
    //      B      (not shown)
 } // end while()
} //end main()
```

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. Assume that your PIC micro current consumption is 20 mA at 40 MHz and 5 V. If the voltage is kept at 5V, what frequency should produce a new current consumption of 10 mA? If the frequency is kept at 40 MHz, what new voltage should produce a current consumption of 10 mA? Use only the formula for dynamic power consumption to compute these results; ignore static power consumption.

2. Draw a diagram that shows how tri-state buffers (TSB) are used to implement a single-wire, bi-directional IO, half-duplex link. Show one diagram that has data flowing left to right, and a second diagram that has data flowing right to left. Show all of the terminals of the TSB and values for the terminals as appropriate.

3. In the code below, a student is trying to debounce a switch on INT1 that generates an interrupt by using a delay in the ISR as was done in lab. What is wrong? Correct the code. Explain the reasoning behind your answer.

```
interrupt my_isr (){
   if (INT1IF) {
      INT1IF = 0; //clear flag
      DelayMsISR(20) // delay 20 ms to debounce
      //rest of code not shown
      }
 }
```

4. What is the SPBRG value for a baud rate of 19200 assuming an FOSC of 20 MHz and low speed mode?

5. In the code below, give what is printed to the console assuming the standard PIC18 setup that you have been using in lab. Explain the reasoning behind your answer.

```
main() {

 serial_init(95,1);  // 19200 in HSPLL mode, crystal = 7.3728 MHz
 SWDTEN = 1;
while (1) {
  printf("Yawn");pcrf();
  DelayMs(30);  //give time for chars to finish printing
  asm("sleep");
  SWDTEN = 0;
  printf("Huh?");pcrlf();
 }//end while()
} //end main()
```

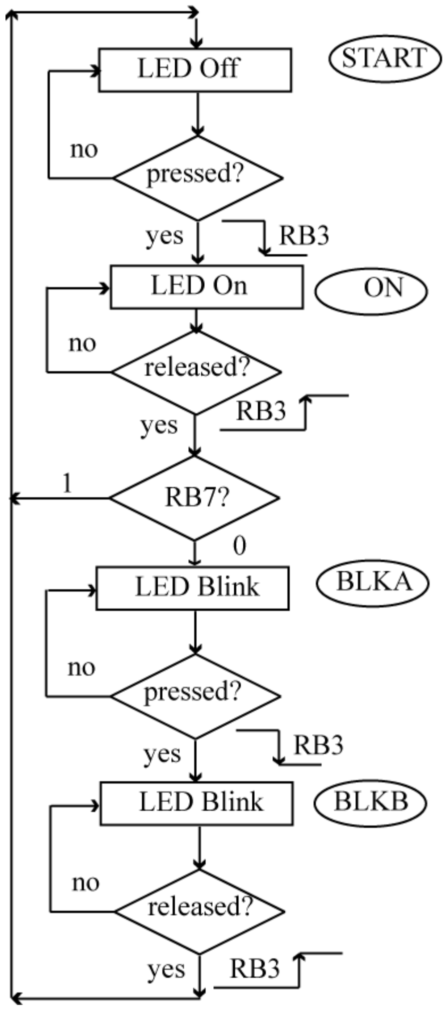
6. Draw a diagram that illustrates the basics of how your PIC micro is connected to the serial port of the PC.

7. Write a *C* code fragment that detects a power up of the PIC18 and prints the message "Power On". Be sure that you do not detect falsely detect a power up.

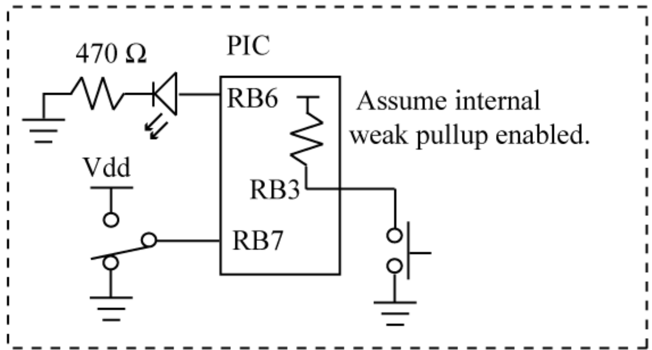
8. In an asynchronous serial interface, what would prevent me from having a data format of 16 data bits + start + stop? Or a format of 32 data bits + start + stop? Or a format of 1000 data bits + start + stop? , i.e., why can't I send an arbitrary number of data bits during an asynchronous data transmission?

Figures

Problem (b)

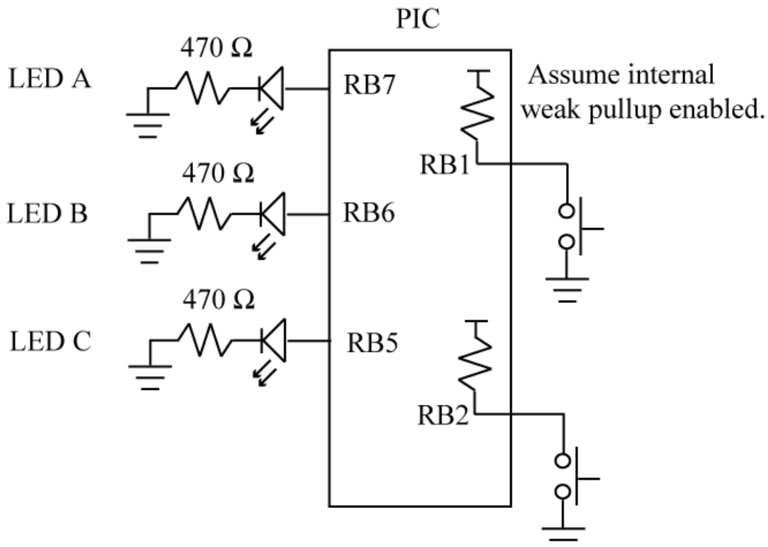



LED Switch IO Problem (a)




Pushbutton
input for RB3

Problem (c)




Actions for Problem (c). Assume LEDs are initally A = ON, B = off, C = off (initialized by main)

On each press and release of RB1, cycle through LEDs in forward direction

LED A = ON, LED B = off, LED C = off

LED A = off, LED B = ON, LED C = off

LED A = off, LED B = off, LED C = ON

On each press and release of RB2, cycle through LEDs in reverse direction

LED changes only happen on a press and release of either RB1 or RB2.
After a total of TEN button press/releases, disable both the RB1 and RB2 interrupts. Button presses of either RB1 or RB2 can happen in any order.
Do not worry about simultaneous button presses.
(Implementation hint: read the value of an LED output to determine what to do next)

An example sequence:

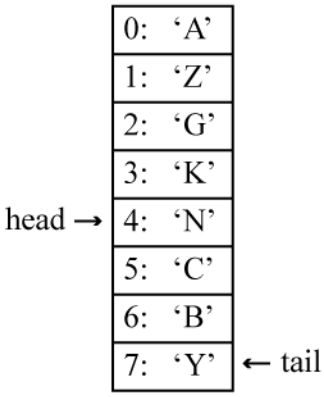
1. LEDs initially A=ON, B=off, C=off.
2. press and release of RB2
3. A=off, B=off, C=ON
4. press & release of RB2
5. A=off, B=ON, C=off
6. press & release of RB1
7. A=off, B=off, C= ON
8. press & release of RB1
9. A=ON, B=off, C=off

etc....

An example sequence:

1. LEDs initially A=ON, B=off, C=off.
2. press and release of RB1
3. A=off, B=ON, C=off
4. press & release of RB1
5. A=off, B=off, C=ON
6. press & release of RB1
7. A=ON, B=off, C= off
8. press & release of RB2
9. A=off, B=off, C=ON
10. press & release of RB2
11. A=off, B=ON, C=off

Problem (f)




Problem (g)