

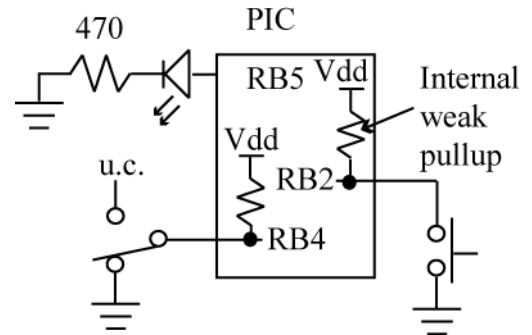
You may use only the provided reference materials.

Part I: (60 pts)

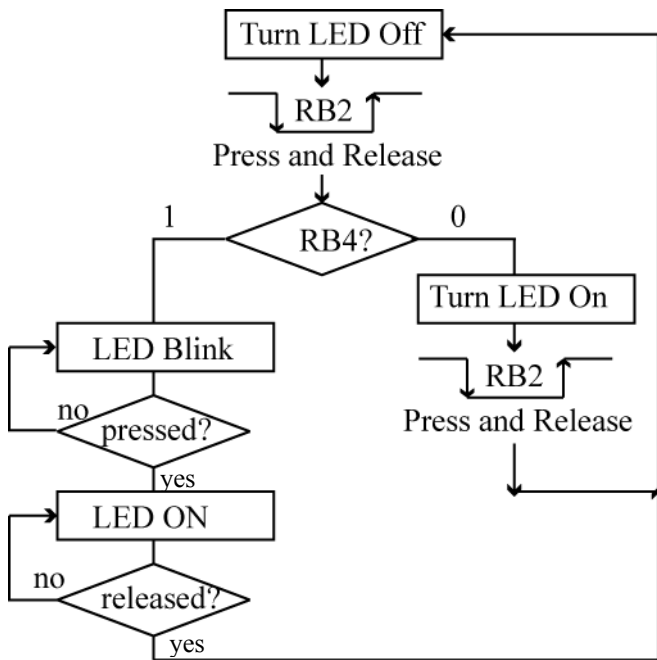
- a. (5 pts) Write C code that configures PORTB for the IO as shown below. The internal weak pullup must be enabled.

```
//one solution
RBPU = 0;
TRISB5 = 0;
TRISB4 = 1;
TRISB2 = 1;
```

```
//another solution
RBPU = 0;
TRISB = 0xDF;
```

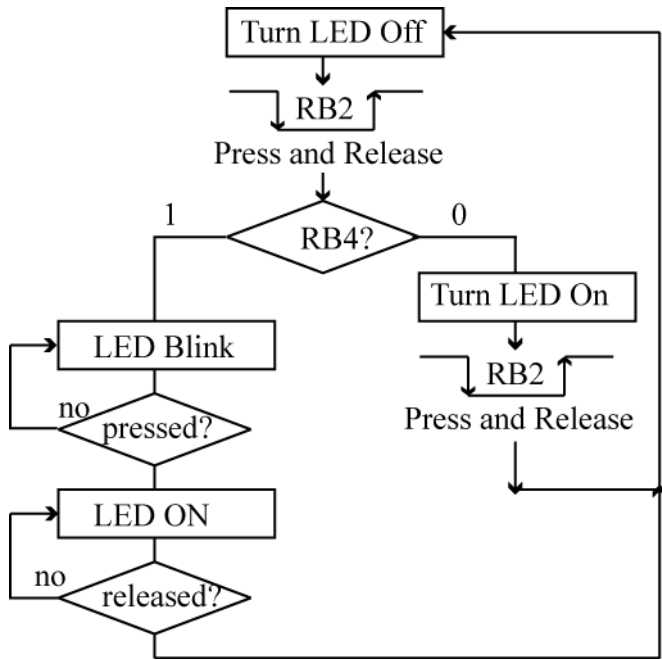


- b. (15 pts) Assuming the IO configuration of the previous problem, write a `while(1){}` loop that implements the LED/Switch IO state machine shown below. Either use a `switch()` statement approach or a `if-then-else` approach. Assume you have available the `DelayMs()` function.



```
#define OFF 0
#define ON_A 1
#define BLINK 2
#define ON_B 3
```

```
char state;
main(){
    //configure ports, not shown
    state = OFF;
    while (1){
        switch(state) {
            case OFF: RB5 = 0;
                while (RB2);
                while (!RB2);
                if (RB4) state = BLINK;
                else state = ON_A;
                break;
            case ON_A: RB5 = 1;
                while (RB2);
                while (!RB2);
                state = OFF;
                break;
            case BLINK:
                while (RB2) {
                    if (LB5) LB5 = 0; else LB5 = 1;
                    DelayMs(200);
                }
                state = ON_B; break;
            case ON_B:
                RB5 = 1;
                while(!RB2); //wait for release
                state = OFF; break;
        }
    }
}
```

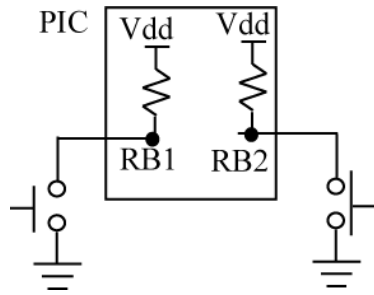


// if-then-else solution

```

main(){
  //configure ports, not shown
  while (1){
    // turn off
    RB5 = 0;
    while (RB2); //wait for press
    while (!RB2); //wait for release
    if (!RB4) {
      RB5 = 1;
      while (RB2); //wait for press
      while (!RB2); //wait for release
    } else {
      // blink
      while (RB2) {
        if (LB5) LB5 = 0; else LB5 = 1;
        DelayMs(200);
      }
      // turn on
      RB5 = 1;
      while(!RB2); //wait for release
    } // end else
  } //end while
}
  
```

- c. ( 10 pts) For the switch configuration below, assume both inputs have been configured to generate rising edge triggered inputs, high priority. 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'.



```

interrupt_isr() {
    if (INT1IF) {
        INT1IF = 0; INT1IE = 0;
        edge_count += 2;
        switch_pressed = 1;
    }
    if (INT2IF) {
        INT2IF = 0; INT2IE = 0;
        edge_count += 1;
        switch_pressed = 1;
    }
}

```

- d. (10 pts) Assuming the ISR of (c), write a *main()* function that enables INT1, INT2 interrupts to be rising edge triggered, with priorities disabled. Configure both pins as inputs. Then enter an infinite *while(1){}* loop that waits for the *switch\_pressed* variable to become a '1' value; after *switch\_pressed* is set by the ISR print a message that says 'switch pressed', clear the *switch\_pressed* variable to a '0', and re-enable both interrupts via their interrupt enable flags.

```

main() {
    switch_pressed = 0;
    RBPU = 0;
    TRISB = 0xFF; // not needed, included for completeness
    IPEN = 0; // disable priorities
    INTEDG1 = 1; INTEDG2 = 1; //rising edge triggered
    INT1IF = 0; INT2IF = 0;
    PEIE = 1; //not really needed as INT1/INT2 are not peripheral interrupts
    GIE = 1; //enable all unmasked interrupts
    while (1){
        INT1IE = 1; INT2IE = 1; //unmask INT1, INT2 interrupts
        switch_pressed = 0;
        while (!switch_pressed); // wait for ISR
        printf("Switch pressed!!!");
    }
}

```

- e. (5 pts) Write C code that implements the *putch(char c)* function (sends one character to the serial port). No interrupts are enabled.

```
putch(char c) {
    while(!TXIF);
    TXREG = c;
}
```

- f. (5 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.

```
if (!RI) {
    RI = 1;
    printf("Software reset
detected!");
}
```

- g. (10 pts) Write a C code function *char getdata()* that waits for data to become available in a circular buffer name *buf*, takes data out of the buffer and returns it. Assume the buffer has a maximum of 16 characters, and pointers named *tail* and *head*. The *head* pointer is used for placing data into the buffer. Assume an ISR is placing data into the buffer.

```
char getdata(){
    while (tail == head);
    tail++;
    if (tail == 16) tail = 0;
    return(buf[tail]);
}
```

Part II: (40 pts) Answer 13 out of the next 15 questions. Cross out the 2 questions that you do not want graded. Each question is worth 3 pts.

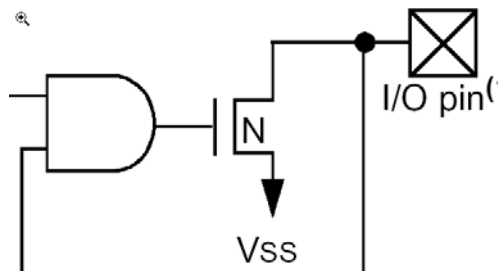
1. Give the value of one bit time in microseconds for a baud rate of 4800.

$$1/4800 = 2.08 \text{ e } -4 = 208 \text{ us}$$

2. Given a voltage of 5 V, a clock freq of 40 MHz, and a current consumption of 20 mA, what is the new current consumption predicted by theory if the voltage is reduced to 3 V and the clock frequency to 10 MHz?

$$20 \text{ mA} (9 * 10) / (25 * 40) = 1.8 \text{ mA}$$

3. Draw an open-drain output..



4. Given a data format of 7-bit + odd parity, what is the value of the parity bit if the data is 0x2A?

0x2A = 010 1010, already has an odd number of '1', so parity bit is 0.

5. What is the SBBRG value for a baud rate of 38,400 assuming an FOSC of 20 MHz and high speed mode?

$$[20\text{MHz} / (38400 * 16)] - 1 = 32$$

6. Define the term half-duplex in reference to a communication channel.

Can communicate in either direction, but only in one direction at a time.

7. How does an RETFIE differ functionally from an RETURN statement?

Sets GIE = 1 to re-enable unmasked interrupts

8. In asynchronous serial communication, why can't we send large numbers of bits at a time? (RS232 uses a maximum of 10 bits).

Sender and receiver do not share common clocks so any clock mismatch error accumulates with each bit that is sent; eventually will sample input bit in wrong place.

9. Why did we need a MAX 202/232 chip in our protoboard? Be specific.

To convert CMOS logic levels (0 V = logic 0, 3-5 V = logic 1) to RS232 levels (-3V to -25 V = logic 1, +3 to +25V = logic 0) and vice-versa.

10. How is a the WDT going off after a *sleep* instruction different from the WDT going off during normal operation?

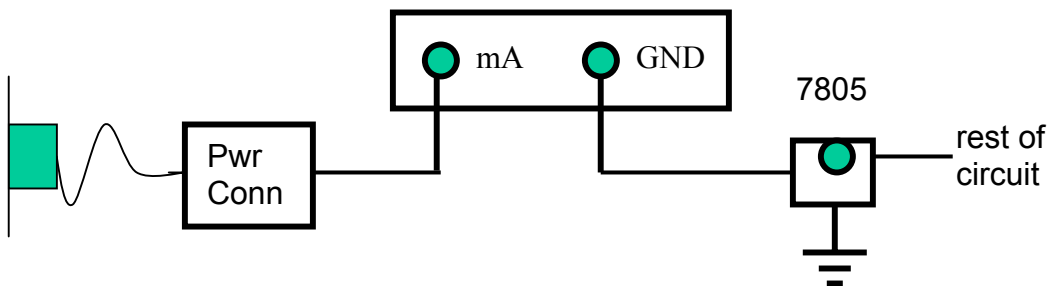
When the WDT expires during normal operation, it forces a processor reset (next instruction is fetched from location 0).

When the WDT expires during sleep, it causes a wake up which means the next instruction is fetched from the location following the *sleep* instruction.

11. The *reset.c* code used in a lab had variable that was declared to be 'persistent' – this caused the compiler to do something special for this variable. What was it?

**The variable is left untouched by the runtime code that initializes variables before main() is executed.**

12. Draw a diagram that shows how to use a multimeter to measure the current flowing through the your ENTIRE protoboard, including the current through the 7805 voltage regulator.



13. When the SLEEP instruction was executed in the *reset.c* program, the current consumption dropped considerably. Why did this occur. Be specific.

**The SLEEP instruction stops the internal clocks of the PIC; no switching activity, much less power consumption.**

14. What mechanism was used in the square root program of experiment 8 that allowed it to deal with bursts of input characters. Be specific.

The UART was configured to generate an interrupt when a character was received. The ISR placed this character into a circular buffer.

15. Given an advantage of serial IO. Give a disadvantage of serial IO. Your advantage cannot be equal to the NOT(disadvantage).

Advantage: fewer external pins required, so connection is cheaper  
Disadvantage: it is slower than parallel IO