

Circle one: JONES section / REESE section

ECE 3724 Test #3 – Fall 2006 Net ID: _____

You may use only the provided reference materials. All figures are on the last pages; questions are on pages 1-8 (four sheets, double sided).

Part I: (80 pts)

- a. (4 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.

```
TRISB = 0xDF;      // since 1 = input, 0 = output,  
                  // the pattern is xx01 xx1x.
```

- 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-else* approach. Assume you have available the *DelayMs()* function for blinking the LED; you do NOT have to include debounce delays for the switch input.

```
while (1) {  
    switch(state) {  
        case LED_OFF :  
            RB5 = 0;  
            while (RB1); //wait for press  
            DelayMs(30);  
            state = LED_BLINK;  
            count = 0;  
            break;  
        case LED_BLINK:  
            //toggle LED  
            if (RB5) RB5 = 0; else RB5 = 1;  
            DelayMs(200);  
            count++;  
            if (count == 10) {  
                state = LED_ON;  
            } else if (RB1) { //check switch  
                //pressed  
                DelayMs(30); //debounce  
                state = LED_OFF;  
            }  
            break;  
        case LED_ON:  
            RB5 = 1;  
            while(!RB1); //wait for release  
            DelayMs(30); //debounce  
            state = LED_OFF;  
            break;  
    }  
}
```

- c. (20 pts) Implement the flowchart and LED/switch configuration of Figure(c) using an interrupt to handle the button on the RB1 input. Use semaphores controlled by the ISR to tell the *while(1)* loop in the main() program to blink, turn on or turn off the LED. Declare all semaphores you use.

1. (14 pts) ISR code

```

volatile char state, led_blink, led_on;
void interrupt pic_isr(void) {
    if (INT1IF) {
        DelayMs(30); //debounce
        INT1IF = 0;
        switch(state){
            case LED_BLINK:
                led_blink = 0;
                led_on = 1;
                state = LED_ON;
                INTEDG1 = 1; //rising edge
                break;
            case LED_ON:
                if (RB4) {
                    led_on = 0;
                    state = LED_OFF;
                }else {
                    led_blink = 1;
                    state = LED_BLINK;
                }
                INTEDG1 = 0; //falling edge
                break;
            case LED_OFF:
                led_on = 1;
                state = LED_ON;
                INTEDG1 = 1; //rising edge
                break;
        } //end switch
    } //end if
} //end interrupt

```

2. (6 pts) *main()* code, be sure to configure and enable your interrupt. Do not assume any default values for variables or control bits.

```

main(){
    TRISB = 0xDF;
    state = LED_BLINK; led_blink = 1; //initially blinking
    RB5 = 0; led_on = 0;
    INTEDG1 = 0; //falling edge
    INT1IF = 0; INT1IE = 1; PEIE = 1; GIE = 1; IPEN = 0;
    while(1){
        if (led_blink) {
            DelayMs(200);
            LB5 = LB5 ^ 1; //toggle led
        }else if (led_on) LB5 = 1;
        else LB5 = 0;
    }
}

```

- d. (6 pts) How many bytes can be sent in 3 seconds assuming a baud rate of 4800, and an asynchronous data format of 1 start bit, 8 data bits, and 2 stop bits assuming the bytes are sent as fast as possible?

**Each byte takes: $11 * 1/4800 = 2.29$ ms, so about 436 bytes per second.
In three seconds, can send about 1309 bytes.**

- e. (6 pts) Write C code that implements the *void putch(char c)* function (write one character to the serial port). No interrupts are enabled.

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

- f. (12 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.(3) From figure F, how many characters are currently *available* in the buffer? (this is not the total number of locations in the buffer) _____

f2. (3) From figure F, what character is returned if the buffer is read?

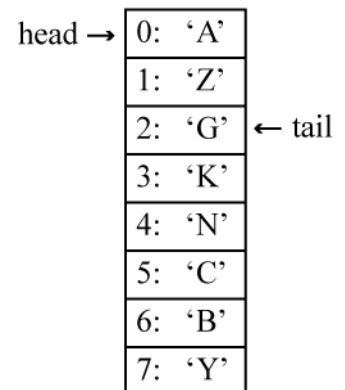
f3. (6) If a softwareFIFO as shown in figure F is used for interrupt driven asynchronous receive on the PIC, circle one (buffer read/buffer write/neither) for each of the following:

f3a. The Buffer Read/Buffer Write/neither is placed in the *putch()* function.

f3b. The Buffer Read/ Buffer Write/neither is placed in the *getch()* function.

f3c. The Buffer Read/ Buffer Write/neither is placed in the interrupt service routine.

Problem (f)



Answers:

f1: 6 (K, N, C, B, Y)

f2: 'K'

f3a neither (putch() is for transmission to the serial port, the software FIFO is for character reception)

f3b: buffer read (getch() reads received serial characters that are placed into the software FIFO)

f3c: buffer write (the ISR is triggered each time a character is received by the USART, and the ISR must write this character into the software FIFO).

- g. (9 pts) For each of the code segments below, draw the waveform that results on output RB7. If the waveform is repeating, then say so and draw at least two cycles. Also, clearly indicate logic '0' and logic '1' on your waveforms. For all code segments, pin RB7 is connected to RB0 as shown. Be sure to clearly indicate the initial state of the RB7 output.

```

g.1
interrupt isr() {
if (INT0IF) {
    INT0IF = 0; RB7 = 1;
}
}
main() {
    TRISB = 0x7F; RB7 = 1;
    IPEN = 0; INTEDG0 = 0; INT0IF = 0;
    INT0IE = 1; PEIE = 1; GIE = 1;
    while (1){
        RB7 = 0;
        while(!RB0);
    }
}

```

```

g.2
interrupt isr() {
if (INT0IF) {
    INT0IF = 0; RB7 = 1; INT0IE = 0;
}
}
main() {
    TRISB = 0x7F; RB7 = 1;
    IPEN = 0; INTEDG0 = 0; INT0IF = 0;
    INT0IE = 1; PEIE = 1; GIE = 1;
    while (1){
        RB7 = 0;
        while(!RB0);
    }
}

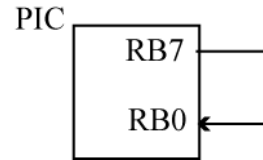
```

```

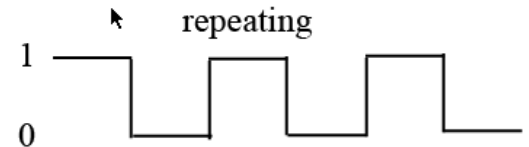
g.3
interrupt isr() {
if (INT0IF) {
    INT0IF = 0; RB7 = 1;
}
}
main() {
    TRISB = 0x7F; RB7 = 1;
    IPEN = 0; INTEDG0 = 1; INT0IF = 0;
    INT0IE = 1; PEIE = 1; GIE = 1;
    while (1){
        RB7 = 0;
        while(!RB0);
    }
}

```

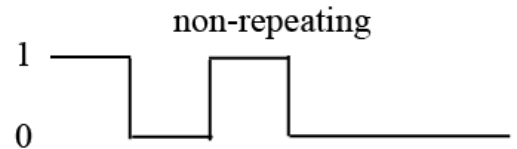
Problem (g)



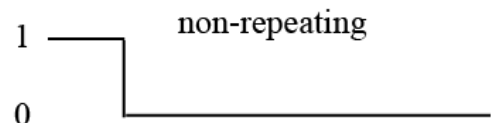
Answer:
RB7 initially high, square wave formed on RB7 output (repeating)



Answer:
RB7 initially high, after the first low-true pulse on the RB7 output, RB7 goes low and remains low because the INT0IF interrupt is disabled (non-repeating)
E.g. high, low, high, low forever



Answer:
RB7 initially high, then goes low and remains low because the interrupt is never triggered since it is programmed for a rising edge interrupt.



- h. (8 pts) Explain what happens in the code below by specifying what appears on the console after the PIC is powered on, and justify your answer by explaining the sequence of events. You must clearly specify if output continually appears on the console or if at some point it stops.

```
main() {
  char c;
  serial_init(95,1); // 19200 in HSPLL mode, crystal = 7.3728 MHz
  SWDTEN = 0;
  if (!POR) {
    POR = 1;
    SWDTEN = 1;
    printf("Hello!");pcrlf();
  }
  if(!SWDTEN) {
    asm("sleep");
  }
  while (1) {
    printf("Looping");pcrlf();
  } //end main()
}
```

Answer:

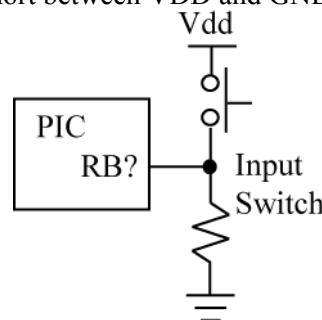
- PIC is powered on.**
- POR bit is '0', so 'Hello' printed and the watchdog timer is enabled.**
- The loop is entered and 'Looping' is printed many times to the console. At some point, the WDT expires, and the PIC is reset.**
- The POR bit is not set this time, so the SWDTEN bit remains a zero; the 'sleep' instruction is executed, and the PIC goes to sleep until it is reawakened by MCLR or power cycle.**

Part II: (20 pts) Answer 5 out of the next 7 questions. Cross out the 2 questions that you do not want graded. Each question is worth 4 pts.

- In the table below, put a checkmark in each entry if an output port of the specified type can be driven to the particular state listed by the port itself (and not by any extra or external components)

	High?	Low?	High Impedance (aka, floating)
Normal CMOS output	yes	yes	no
Tri-State output	yes	yes	yes, when output disabled
Open-drain output	no	yes	yes, if output not pulled low, then floats

- Draw a pushbutton connected to a PIC18 port that will provide a logic '1' when the switch is pressed, and a logic '0' when the switch is released. Your connection must be electrically viable, i.e. you cannot produce a short between VDD and GND or any other undesirable condition.

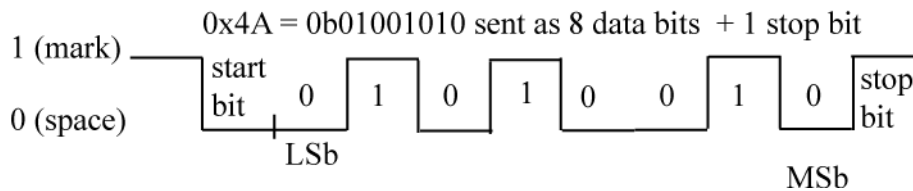


3. If the PIC is drawing 3 mA using the HS oscillator option, and this is changed to the HSPLL option, what would you expect the new current draw to be? (give a number) Why? Justify your answer.

Answer:

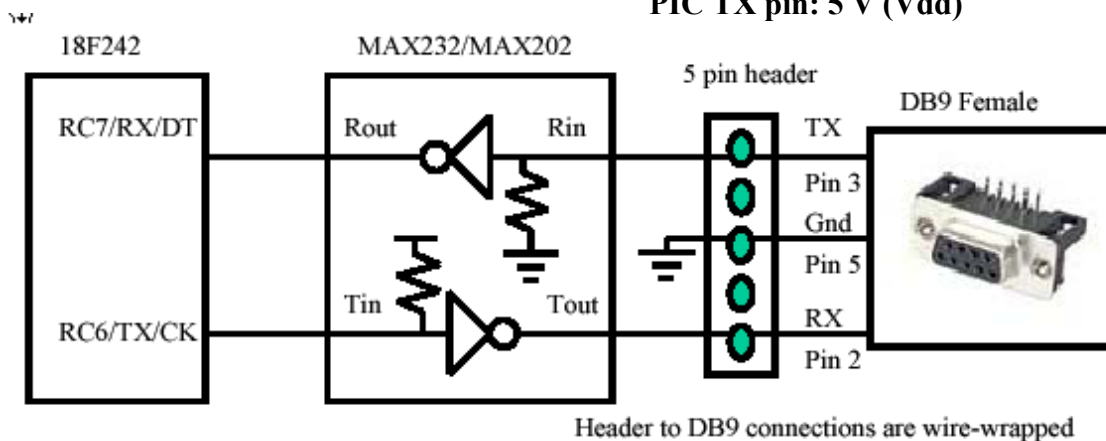
12 ma since the HSPLL option increases the current clock speed by 4, and current varies linearly with clock speed.

4. Draw the waveform for the value 0x4A using 1 start bit, 1 stop bit, and 8-data bits as transmitted by the PIC from its TX pin using asynchronous serial IO.



5. In the diagram below, assume the DB9 is connected to the RS232 port of a PC. If you measured the voltage at the TX pin of the DB9 when the port is idle (no characters being transmitted), about what voltage would you expect to see? If you measured the voltage at the PIC18 TX pin when the port is idle (no characters being transmitted), about what voltage would you expect to see?

DB9 TX pin: ~ -10 V (spec is -3V to -25V)
PIC TX pin: 5 V (Vdd)



6. How does an RETFIE instruction differ functionally from a RETURN instruction (what extra actions are taken by the RETFIE instruction that are not taken by the RETURN instruction)? Thinking about what happens during a hardware interrupt may help you to remember this.

Answer:

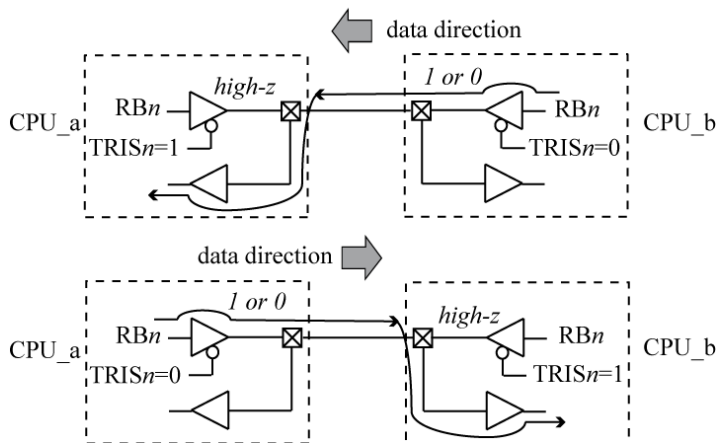
Restores BSR, STATUS, W from the shadow registers, and sets GIE=1, re-enabling interrupts.

7. Draw a diagram that shows how a half-duplex communication channel is built using a single bi-directional wire and tri-state buffers (TSB). Show how communication is accomplished from left to right, and indicate the state of each TSB (enabled or disabled). Repeat the same diagram, and show how communication is accomplished from right to left, and indicate the state of each TSB (enabled or disabled).

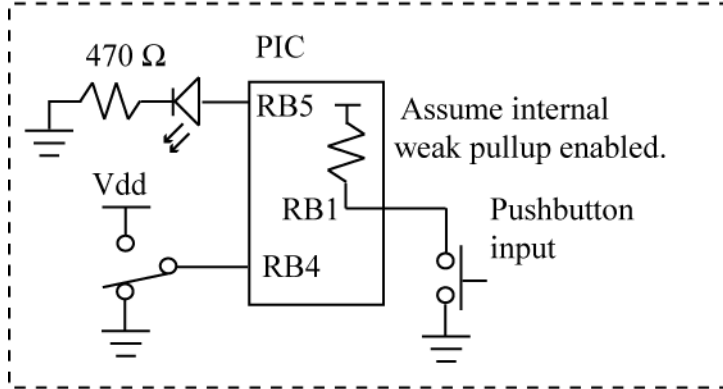
Tristate buffer with low-true enable



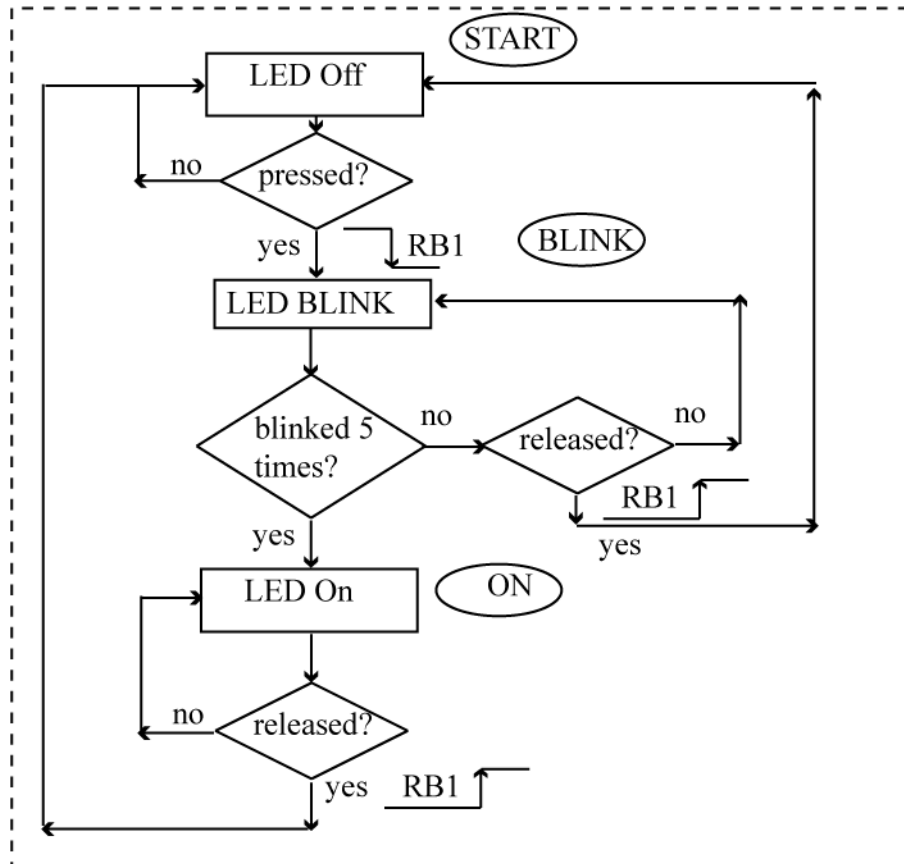
Tristate buffer with high-true enable



Problem (a)



Problem (b)



Problem (c)

