ECE 376 - Final Exam: Name ____________________

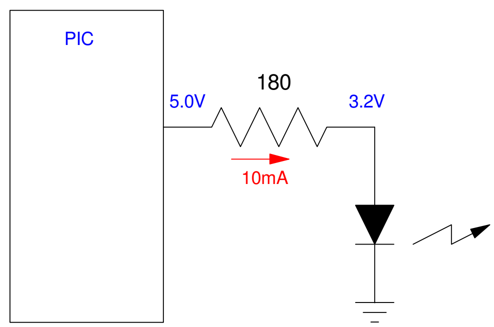
Open-Book, Open Note, Calculators and Matlab permitted. Individual Effort.

1a) Binary Outputs:

1a) Give a circuit which allows a PIC to turn on and off a 32mW LED

- Id = 10mA
- Vd = 3.2V

For outputs less than 5V and less than 25mA, just use a resistor:



1b) Give a circuit which allows a PIC to turn on and off a 30W LED

- Vd = 10.0V
- Id = 3.0A

For other outputs, use a transistor. Assume

- A 20V power supply (arbitrary)
- An NPN transistor with a gain of 200 (arbitrary - 6144 NPN transistor specs)

Rc:

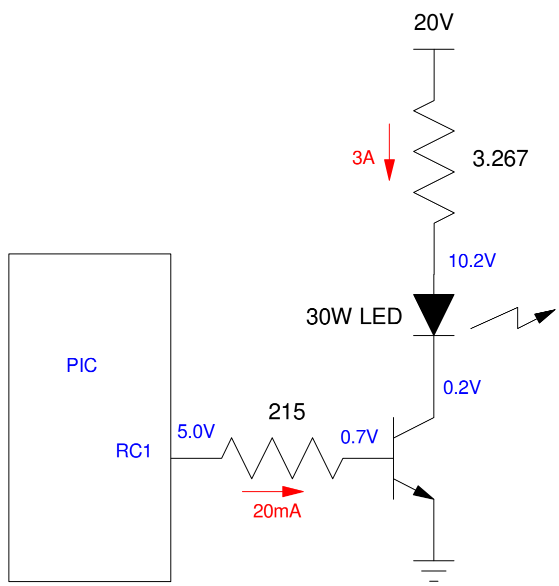
$$R_c = \left(\frac{20V-10V-0.2V}{3A}\right) = 3.267\Omega$$

Rb:

$$I_b > \frac{I_c}{\beta} = \frac{3A}{200} = 15mA$$

Let Rb = 20mA

$$R_b = \left(\frac{5V-0.7V}{20mA}\right) = 215\Omega$$

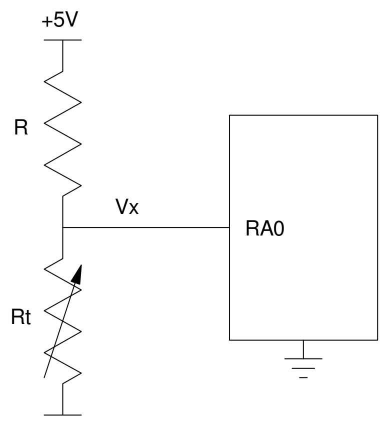


2) Analog Inputs: Determine the voltage, resistance, and temperature if a PIC reads 417 on the A/D input for the following circuit. Assume

- R = 1100 + 100*(your birth month) + (your birth date). For example, May 14th would give R = 1514 Ohms.
- Rt is a thermistor with the temperature - resistance relationship (T = temperature in degrees C)

$$R_t = 2000 \cdot \exp\left(\frac{4400}{T+273} - \frac{4400}{298}\right)\Omega$$

R 1100 + 100*mo + day	raw A/D reading	Vx Volts	Rt Ohms	T Degrees C
1614	**417**	**2.038V**	**1110 Ohms** varies with R	**37.36C** varies with R



$$V_x = \left(\frac{417}{1023}\right)5V = 2.038V$$

$$V_x = 2.038V = \left(\frac{R_t}{R_t+R}\right)5V$$

$$R_t = \left(\frac{2.038V}{5V-2.038V}\right)1614\Omega = 1110\Omega$$

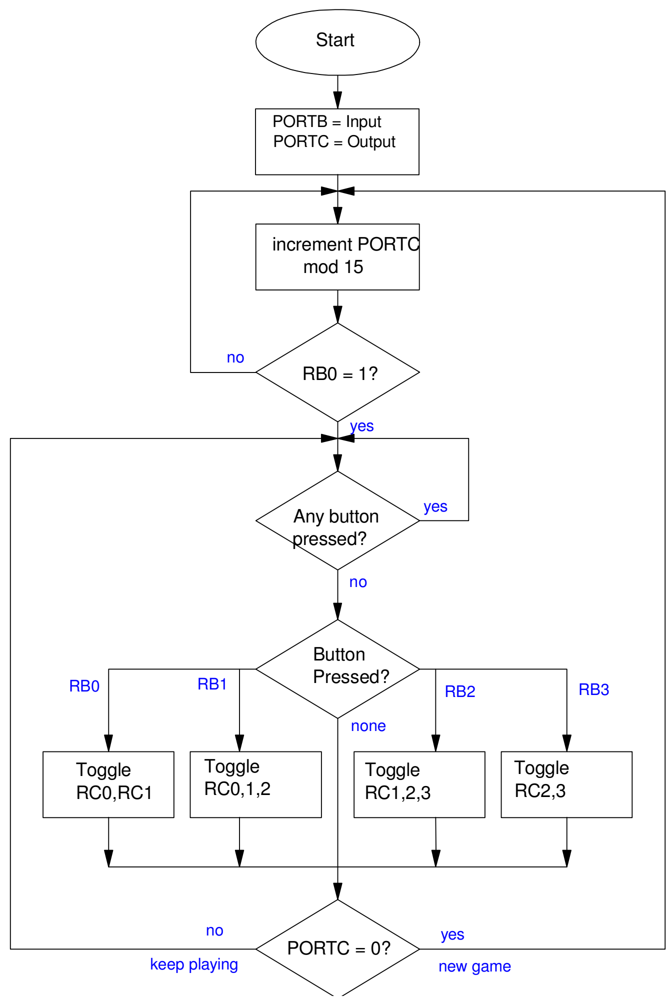
$$T = 37.36^0 C$$

3) C-Coding: Lights Out is a game where

- You start the game by pressing RB0.
- At the start, four random lights are turned on (PORTC = 0..15)
- Once started, you can toggle any light along with its neighbors by pressing buttons RB0/RB1/RB2/RB3
  - For example, if you press RB1, lights RC1 is toggled along with its adjacent lights (toggle RC0/RC1/RC2).
- The goal is to turn all of the light off with the minimum number or button presses.

Write a C program which corresponds with the following flow chart for the game of Lights Out:

```
void main(void) {

ADCON1 = 0x0F;

TRISB = 0xFF;
TRISC = 0x00;


while(1) {

   do {
      PORTC = (PORTC + 1) % 15;
      } while(!RB0);

   do {

      while(PORTB);

//    while(PORTB == 0);

      if(RB0) PORTC = PORTC ^ 0x03;
      if(RB1) PORTC = PORTC ^ 0x07;
      if(RB2) PORTC = PORTC ^ 0x0E;
      if(RB3) PORTC = PORTC ^ 0x0C;

      } while(PORTC);

   }
```



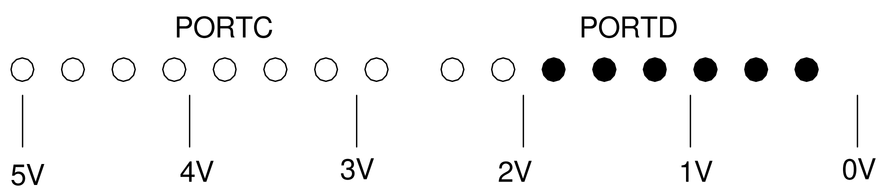
note: game won't quite work as shown. When you press a button, you need to wait for it to be released. Add the section in blue to fix this.

4) C Coding with Analog Inputs: Write a C subroutine which turns your PIC in to a bar-graph for voltage.

When called,

- The subroutine reads the A/D input (0..1023)
- It then turns on LEDs on PORTC:PORTD to display the corresponding voltage as a bar graph:
  - 0V turns off all of the LEDs
  - 1V turns on 1/5th of the LEDs
  - 2V turns on 2/5ths of the LEDs
  - etc.

For example, if 2.00V was input on the A/R reading, the first 6 LEDs on PORTD would turn on.




```
void BarGraph(void) {

   unsigned int A2D;

   A2D = A2D_Read(0);

   if      (A2D <  60) { PORTC = 0x00; PORTD = 0x00; }
   elseif  (A2D < 120) { PORTC = 0x00; PORTD = 0x01; }
   elseif  (A2D < 180) { PORTC = 0x00; PORTD = 0x03; }
   elseif  (A2D < 240) { PORTC = 0x00; PORTD = 0x07; }
   elseif  (A2D < 300) { PORTC = 0x00; PORTD = 0x0F; }
   elseif  (A2D < 360) { PORTC = 0x00; PORTD = 0x1F; }
   elseif  (A2D < 420) { PORTC = 0x00; PORTD = 0x3F; }
   elseif  (A2D < 480) { PORTC = 0x00; PORTD = 0x7F; }
   elseif  (A2D < 540) { PORTC = 0x00; PORTD = 0xFF; }
   elseif  (A2D < 600) { PORTC = 0x01; PORTD = 0xFF; }
   elseif  (A2D < 660) { PORTC = 0x03; PORTD = 0xFF; }
   elseif  (A2D < 720) { PORTC = 0x07; PORTD = 0xFF; }
   elseif  (A2D < 780) { PORTC = 0x0F; PORTD = 0xFF; }
   elseif  (A2D < 840) { PORTC = 0x1F; PORTD = 0xFF; }
   elseif  (A2D < 900) { PORTC = 0x3F; PORTD = 0xFF; }
   elseif  (A2D < 960) { PORTC = 0x7F; PORTD = 0xFF; }
   else                { PORTC = 0xFF; PORTD = 0xFF; }

}
```

not stylish, but it works. You can do almost anything with if() statements.

5) Interrupts: Ohmmeters often times have a short-circuit test option. When you select this mode of operation, a tone will play if the resistance you're measuring is less than 1 Ohm.

Assume a 100 Ohm resistor is used for a voltage divider so that an A/D reading of 10 or less corresponds to R < 1 Ohm

Write a C program using Timer2 and Timer0 interrupts to

- Sample the A/D reading every 3.00ms, and
- Play 372Hz if the A/D reading is 10 or less

a) Interrupt Initialization

Timer0 Initialization N = 13,440	Timer2 Initialization A*B*C = 30,000 (3ms)		
PS	A	B	C
1	**16**	**117**	**16**

Timer0 Interrupt Routine Play 372Hz if A/D reading is 10 or less	Timer2 Interrupt Sample the A/D every 3.00ms

Timer0 Interrupt Routine:

```
if(TMR0IF) {
   TMR0 = -13440;

   if(A2D < 11) RC0 = !RC0;

   TMR0IF = 0;
   }

// assumes A2D is a global variable
```

Timer2 Interrupt:

```
if(TMR2IF) {

   A2D = A2D_Read(0);

   TMR2IF = 0;
   }
```

6) Interrupts: Timer1 Compare. Write the interrupt service routines for a C program which measures how long it takes you to press button connected to RC2 ten times using Timer1 Compare.

- RB0 restarts the game (resets the counter on an INT0 interrupt)
- RB7 goes 0V/5V

6a) Initialization for interrupts

INT0 rising or falling edge?	TIMER1 prescalar = ?	Timer1 Capture 1 rising / falling / 4th rising / 16th rising edge?
rising	**PS = 1**	**Rising**

6b) Write the interrupt service routines

```
// Global variables
unsigend long int START, END, TIME, PERIOD;
unsigned int N;

// time of 10 presses stored in PERIOD
```

INT0 resets the counter (new game)	TIMER1	Timer1 Capture 1 counts presses saves time of 10 presses in global variable TIME10

`if(INT0IF) {`
 	
`N = 0;`
 	
`START = TIME + TMR1;`
 	
`INT0IF = 0;`
 	
`}`
	`if(TMR1IF) {`
	
	`TIME = TIME + 0x10000;`
	
	`TMR1IF = 0;`
	`}`
		`if (CCPR1IF) {`
		
		`N += 1;`
		
		`if(N == 10) {`
		
		`END = TIME + CCPR1;`
		
		`PERIOD = END - START;`
		
		`}`
		
		`CCPR1IF = 0;`
		`}`

note: It would be more accurate to use Timer1 Capture2 interrupts for a rising edge on RC1 to start the game. That would record the start time to 100ns. As written, the START is off by about 50 clocks due to using INT0 interrupts.