ECE 376 - Homework #8

Timer2 Interrupts - Due Monday, November 4th

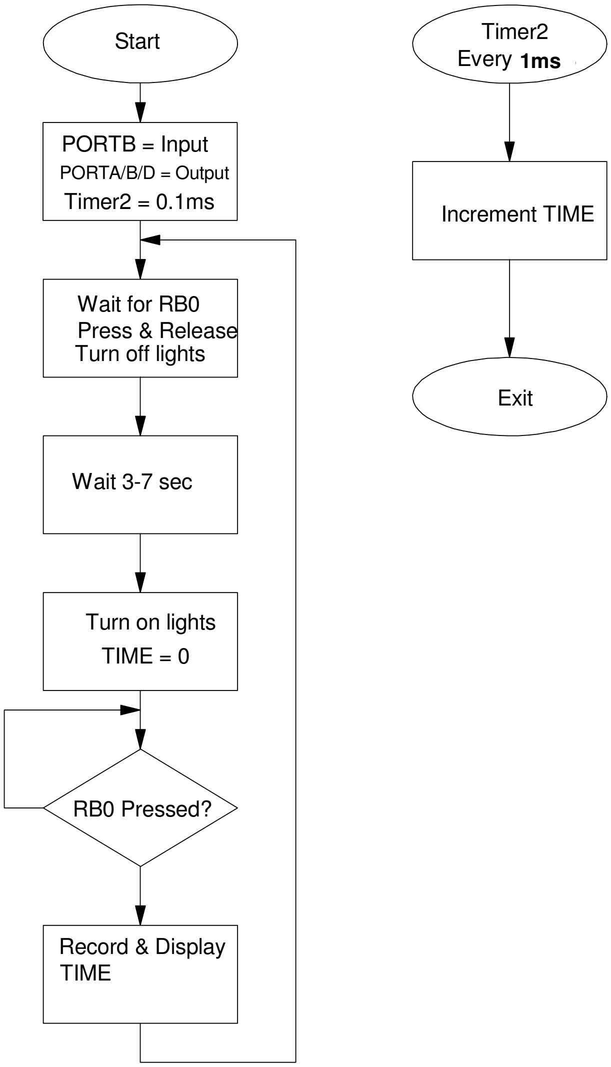
Measuring Time with Timer2

Write a program to measure your reflex time with a resolution of 0.1ms using Timer2 interrupts.

- Press and release RB0 to start the game
- This geneates a random number from 3.0000 to 7.0000 seconds.
- Start decrementing time down to 0.0000 seconds using Timer2 interrupts
- When you get to 0.0000, turn on the lights on PORTA
- As soon as the lights turn on, press RB0 again

The time delay between when the lights turned on and you pressed RB0 is your reflex time.

1) Give a flow chart for this program



2) Write the corresponding C code

Interrupt Service Routine:

```
// Global Variables
unsigned long int TIME;

// High-priority service
void interrupt IntServe(void)
{
   if (TMR2IF) {
      RC0 = !RC0;
      TIME = TIME + 1;
      TMR2IF = 0;
      }
}
```

Initialization

```
// set up Timer2 for 0.11ms
   T2CON = 0x4D;
   PR2 = 24;
   TMR2ON = 1;
   TMR2IE = 1;
   TMR2IP = 1;
   PEIE = 1;

// turn on all interrupts
   GIE = 1;
```

Main Loop

```
   while(1) {


      }
   }
```

```
Memory Summary:
    Program space        used   B02h (  2818) of 10000h bytes   (  4.3%)
    Data space           used    37h (    55) of   F80h bytes   (  1.4%)
    EEPROM space         used     0h (     0) of   400h bytes   (  0.0%)
    ID Location space    used     0h (     0) of     8h nibbles (  0.0%)
    Configuration bits   used     0h (     0) of     7h words   (  0.0%)
```

3) Validation: Collect data to verify your code works

Timer2 is interrupting every 0.1ms

- RC0 measures at 5007Hz
- Timer2 is running at 99.86us (0.14% error)

The delay is random from 3 to 7 seconds

- Time delay for five runs were:
- {3.234s, 4.022s, 5.103s, 6.705s, 3.241s, 6.864s}
- All times were in the range of (3.000, 7.000) seconds

The time from when the lights turn on and you press RB0 is recorded correctly

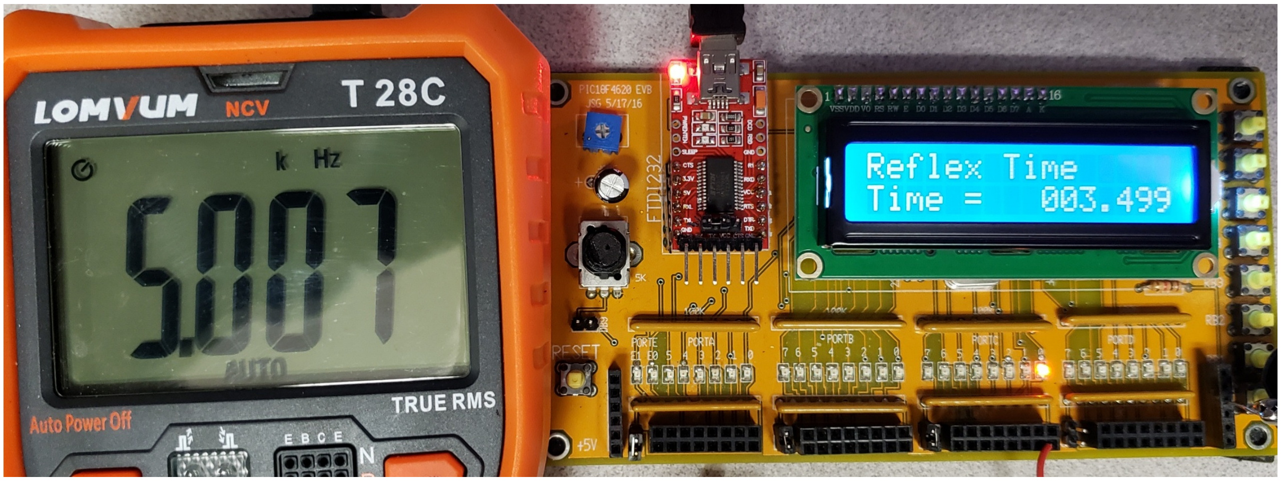
Wait five seconds

- time displayed was 4.6546 seconds

Wait nine seconds

- time displayed was 8.4687 seconds

Timer appears to be correct



Reflex Time
Time =   003.499

4) Student-t Test: Once your program works, collect 2+ measurements of your reflex time.

- From your data, compute the 90% confidence interval for your reflex time.

Measure my reflex times:

{0.1749, 0.1688, 0.2415, 0.2143, 0.1793, 0.1858, 0.1880}

From Matlab, the mean and standard deviation are:

```
>> Data = [0.1749,  0.1688, 0.2415,  0.2143,  0.1793, 0.1858, 0.1880];
>> x = mean(Data)
x =    0.1932

>> s = std(Data)
s =    0.0258
```

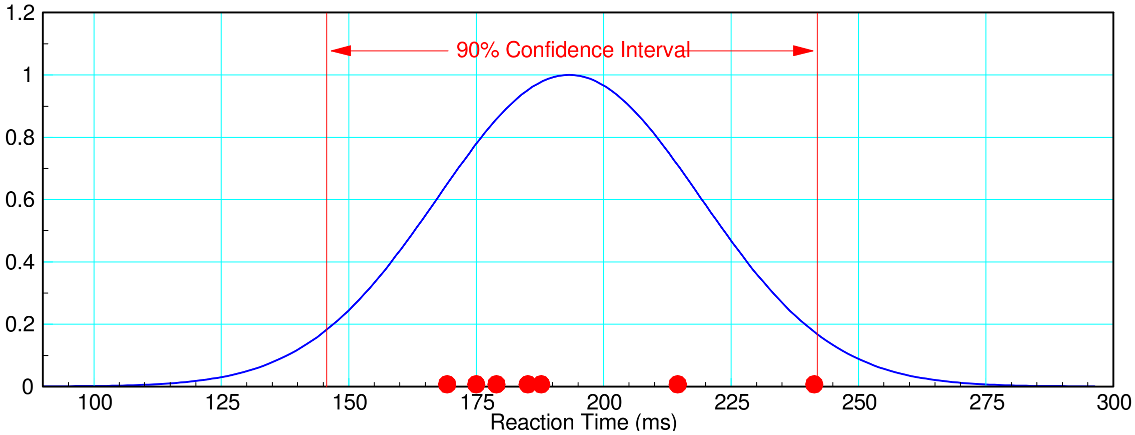
From StatTrek, 5% tails with six degrees of freedom has a t-score of 1.943.

The 90% confidence interval for my reaction time in any given trial is (143.2ms, 243.3ms):

- inividual question

```
>> x + 1.943*s
ans =    0.2433

>> x - 1.943*s
ans =    0.1432
```



The 90% confidence interval for my average reaction time is (174.3ms, 212.2ms):

- population question

```
>> x + 1.943*s/sqrt(7)
ans =    0.2122

>> x - 1.943*s/sqrt(7)
ans =    0.1743
```

Generating Frequencies with Timer2

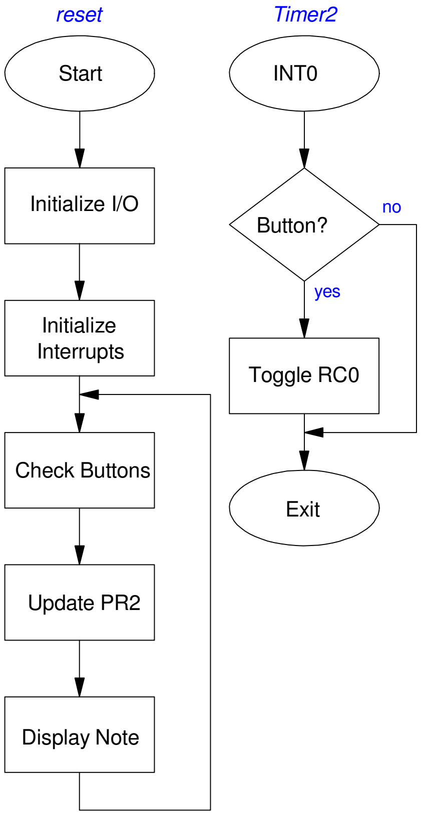
Turn your PIC board into an 8-key piano using Timer 2 interrupts.

- A note plays on a speaker as long as a button is held down.
- The frequency played depends upon the button:

button	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
note	A2	B2	C3	D3	E3	F3	G3	A3
Hz	110	123.47	130.81	146.83	164.81	174.61	196	220
N	45,454.55	40,495.34	38,222.5	34,052.52	30,337.23	28,634.59	25,510.46	22,727.27
A	12	12	12	12	12	12	12	12
B	236.74	210.91	199.08	177.36	158.01	149.14	132.87	118.37
C	16	16	16	16	16	16	16	16

5) Give a flow chart for this program

- One flow chart for the main routine
- One flow chart for each interrupts



6) Write the corresponding C code

```
// Global Variables
unsigned long int TIME;

// High-priority service
void interrupt IntServe(void)
{
   if (TMR2IF) {
      RA1 = !RA1;
      if(PORTB) RA2 = !RA2;
      TMR2IF = 0;
      }
   }

// set up Timer2 for A=12, C=4
   T2CON = 0x5F;
   PR2 = 178;
   TMR2ON = 1;
   TMR2IE = 1;
   TMR2IP = 1;
   PEIE = 1;

// turn on all interrupts
GIE = 1;

   while(1) {






      }
   }
```

```
Memory Summary:
    Program space        used   9E4h (  2532) of 10000h bytes   (  3.9%)
    Data space           used    35h (    53) of   F80h bytes   (  1.3%)
    EEPROM space         used     0h (     0) of   400h bytes   (  0.0%)
    ID Location space    used     0h (     0) of     8h nibbles (  0.0%)
    Configuration bits   used     0h (     0) of     7h words    (  0.0%)
```

7) Validation: Collect data to verify your code works

- Measure the frequency of each note
- Verify a note plays when a button is held down
- Verify the piano is silent when no buttons are pressed

Button	Hz	Hz (actual)	Error (%)
RB7	110	110	0
RB6	123.47	123.5	0.02
RB5	130.81	131	0.15
RB4	146.83	147.3	0.32
RB3	164.81	165	0.12
RB2	174.61	175	0.22
RB1	196	196	0
RB0	220	221	0.45

8) What happens when you press two buttons at once?

Determine by running your program

- RB7 & RB0 = 221Hz
- RB6 & RB1 = 196Hz
- RB5 & RB2 = 175.0Hz

Explain why this makes sense based upon how you wrote your code.

The way the code is written, the last button checked is the one that wins: it over-writes the previous value of PR2.

If instead I had used else if statements, the first button checked would have won.