ECE 376 - Homework #4

C Programming and LCD Displays. Due Monday, September 26th

Please make the subject "ECE 376 HW#4" if submitting homework electronically to Jacob_Glower@yahoo.com (or on blackboard)

1) Determine how many clocks the following C code takes to execute

- Compile and download the code (modify working code and replace the main loop)
- Measure the frequency you see on RC0 (toggles every loop).
  - Use an osiclloscope - or -
  - Connect a speaker to RC0 with a 200 Ohm resistor and measure the frequency with a cell phone app like Piano Tuner
  - RC1 is 1/2 the frequency of RC0, RC2 is 1/4th, RC3 = 1/8th, etc
- The number of clocks it takes to execute each loop is

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right)$$

1a) Counting mod 128

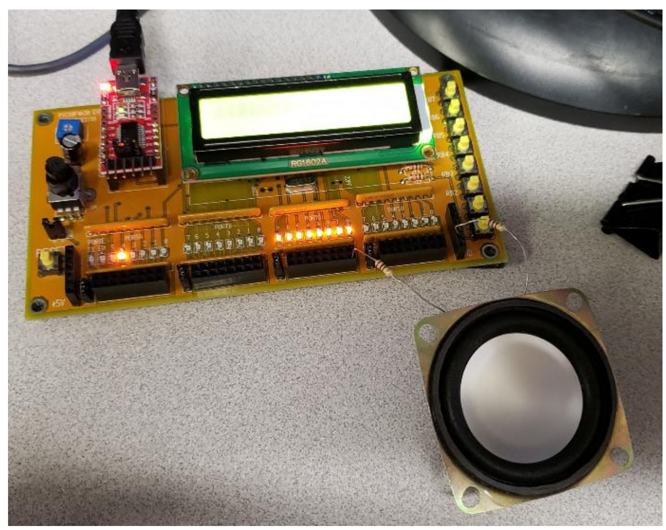
```
unsigned char i
while(1) {
      i = (i + 1) % 128;
      if(i == 0) PORTC += 1;
      }
```

From Pano Tuner, f = 2445.2Hz

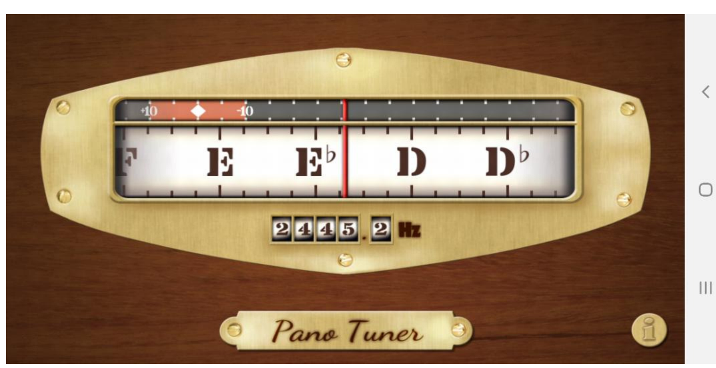
$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 2044.82$$

PORTC counts every 128th count, so each loop takes N/128

$$N/128 = 15.975$$



It takes about 16 locks to count mod 128



1b) Counting mod 127

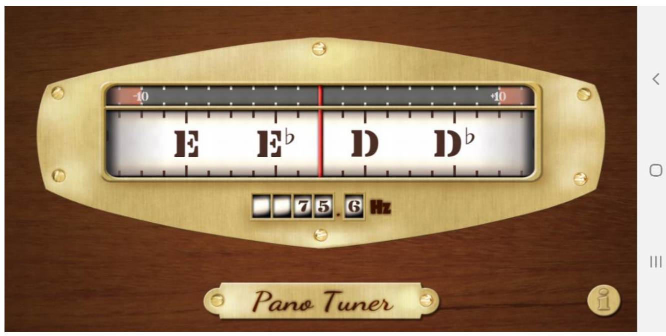
```
unsigned char i
while(1) {
      i = (i + 1)% 127;
      if(i == 0) PORTC += 1;
      }
```

With this code, f = 75.6Hz

$$N = \left(\frac{10,000,000}{2 \cdot Hz}\right) = 66,137.566$$

$$N/127 = 520.76$$

It takes about 521 clocks to count mod 127



1c) Long Integer Addition

```
unsigned long int A, B, C;
unsigned char i;
A = 0x12345678;
B = 0;
while(1) {
    i = (i + 1)% 128;
    if (i == 0) PORTC += 1;
    B = B + A;
    }
```

f = 795.2Hz

$$N = \left(\frac{10,000,000}{2 \cdot Hz}\right) = 6287.72$$

$$N/128 = 49.12$$

$$N/128 - 16 = 33.12$$

It takes 16 clocks to count mod 128

It takes an additional 33 clocks to add a long integer

1d) Floating point division

```
float A, B, C;
A = 3.14159265379;
B = 2.718281828;
while(1) {
    i = (i + 1)% 8;
    if(i == 0) PORTC += 1;
    C = A / B;
    }
```

f = 323.6 Hz

$$N = \left(\frac{10{,}000{,}000}{2*Hz}\right) = 15{,}451.17$$

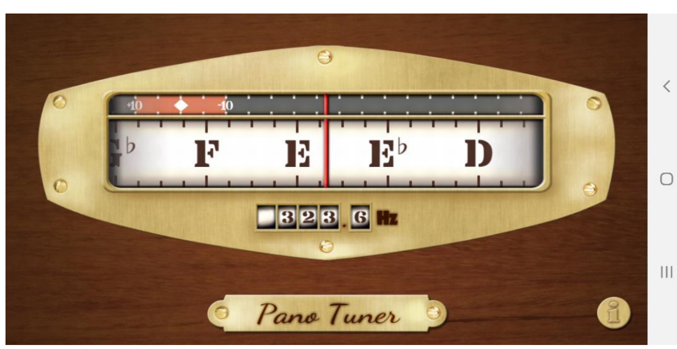
$$N/8 = 1931.39$$

$$N/8 - 16 = 1915.39$$

It takes 1931 clocks per loop

It takes 16 clocks to count mod 8 (same as mod 128)

It takes an additional 1915 clocks to do a single floating point division



$65 Voting Machine

2) Write a C program which turns your PIC into a voting machine capable of counting up to 65,535 votes per candidate (16-bit numbers):

```
// Global Variables

const unsigned char MSG0[20] = "Voting Machine      ";

// Subroutine Declarations
#include <pic18.h>

// Subroutines
#include        "lcd_portd.c"

// Main Routine

void main(void)
{
   unsigned int i;
   unsigned int A, B, C, D;

   TRISA = 0;
   TRISB = 0xFF;
   TRISC = 0;
   TRISD = 0;
   TRISE = 0;
   ADCON1 = 0x0F;

   LCD_Init();                  // initialize the LCD

   A = 0;
   B = 0;
   C = 0;
   D = 0;

   LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MSG0[i]);
   Wait_ms(70);

   while(1) {
   :
   :
   C Code
   :
   :

   }
```

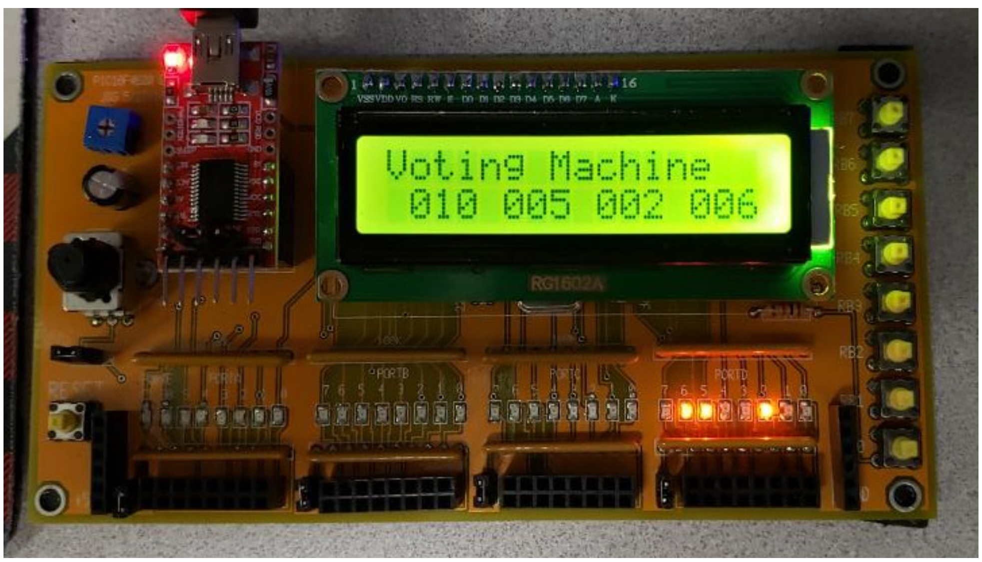
3) How many lines of assembler does your code compile into?

 # instructions = 2782/2 = 1391 lines of assembler

```
Memory Summary:
    Program space        used   ADEh (  2782) of 10000h bytes   (  4.2%)
    Data space           used    2Dh (    45) of   F80h bytes   (  1.1%)
    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%)
```

4) Collect data to verify your voting machine works (each press results in one vote for the correct candidate)

- Press D six times (D counts to six)
- Press C two times (C counts to two)
- Press B five times (B counts to five)
- Press A ten times (A counts to ten)



$65 Banjo

5) Requirements: Specify the inputs / outputs / how they relate.

Inputs: Buttone RB0 .. RB3

Outputs: RC0

Relationship

Play the following notes when a button is pressed

- RB0: C4 (261.63Hz)
- RB1: G4 (392.00Hz)
- RB2: B3 (246.94Hz)
- RB3: D4 (293.66Hz)

Tolerance: +/- 1%

6) C code, flow chart, and resulting number of lines of assembler

To generate a note, the following test code was used

```
void main(void)
{
   unsigned int i;

   TRISA = 0;
   TRISB = 0xFF;
   TRISC = 0;
   TRISD = 0;
   TRISE = 0;
   ADCON1 = 0x0F;

   while(1) {
      if(RB0) {
         RC0 = !RC0;
         for(i=0; i<1000; i++);
         }
      }
   }
```

The results was a 312.2Hz square wave.

To output different freuqencies, change the count:

RB0: C4 (261.63Hz)

$$N = \left(\frac{312.2Hz}{261.63Hz}\right)1000 = 1193$$

RB1: G4 (392.00Hz)

$$N = \left(\frac{312.2Hz}{392.00Hz}\right)1000 = 796$$

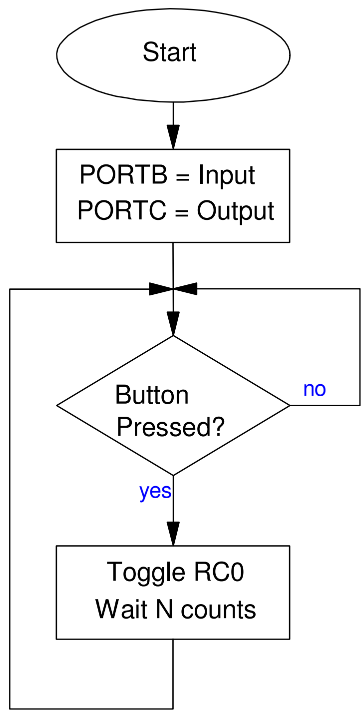
RB2: B3 (246.94Hz)

$$N = \left(\frac{312.2Hz}{246.94Hz}\right)1000 = 1264$$

RB3: D4 (293.66Hz)

$$N = \left(\frac{312.2Hz}{293.66Hz}\right)1000 = 1063$$

Flow Chart



```
// ---  Banjo.C -------------------

// Global Variables
unsigned char MSG0[16] = "Electronic Banjo";
unsigned char MSG1[16] = "C4 (261.63Hz)   ";
unsigned char MSG2[16] = "G4 (392.00Hz)   ";
unsigned char MSG3[16] = "B3 (246.94Hz)   ";
unsigned char MSG4[16] = "D4 (293.66Hz)   ";

// Subroutine Declarations
#include <pic18.h>
#include "LCD_PortD.c"

// Main Routine

void main(void)
{
   unsigned int i;

   TRISA = 0;
   TRISB = 0xFF;
   TRISC = 0;
   TRISD = 0;
   TRISE = 0;
   ADCON1 = 0x0F;

   LCD_Init();
   LCD_Move(0,0);
   for(i=0; i<16; i++) LCD_Write(MSG0[i]);

   while(1) {
         :
         :
         C Code
         :
         :
         }
   }
```

7) Validation: Collect data in lab to verify you met the requirements.

Refer to the requirements

Inputs: Button RB0 .. RB3

- Yes, buttons RB0..RB3 are inputs (LED lights up when pressed)

Outputs: RC0

- Yes - connecting a speaker to RC0 plays a note

Relationship

Play the following notes when a button is pressed

- **RB0: C4 (261.63Hz)**
- **RB1: G4 (392.00Hz)**
- **RB2: B3 (246.94Hz)**
- **RB3: D4 (293.66Hz)**

Tolerance: +/- 1%

Data:

Button	RB0	RB1	RB2	RB3
Hz (desired)	261.63	392	246.94	293.66
Hz (actual)	261.6	391.3	247.0	293.4
% Error	-0.01%	-0.18%	+0.02%	-0.09%
Within tolerance?	yes	yes	yes	yes

8) Demo