ECE 376 - Homework #4

C Programming and LCD Displays - Due Monday, February 12th

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

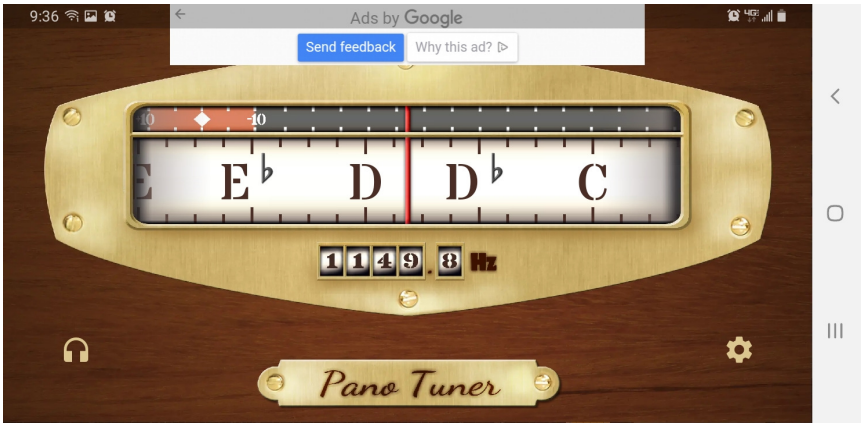
```
unsigned char i
while(1) {
      i = (i + 1) % 128;
      if(i == 0) PORTC += 1;
      }
```

f = 1149.8Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 4348.58 \text{ clocks for 128 loops}$$

$$\frac{N}{128} = 33.97$$

It takes 34 clocks to count mod 128



1b) Counting mod 127

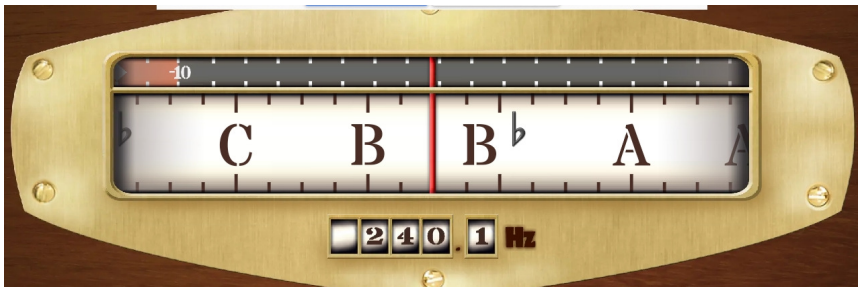
```
unsigned char i
while(1) {
      i = (i + 1)% 127;
      if(i == 0) PORTC += 1;
      }
```

f = 240.1Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 20,824.65 \text{ clocks for 127 loops}$$

$$\frac{N}{127} = 163.97$$

It takes 164 clocks to count mod 127



1c) Floating Point Multiplication

- note: you need to include Math.h #include <math.h>

```
float A, B, C;
A = sqrt(3);
B = sqrt(2);
while(1) {
    i = (i + 1)% 16;
    if(i == 0) PORTC += 1;
    C = A * B;
    }
```

Frequency = 252.7Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 19,786.307 \text{ clocks for 16 loops}$$

$$\frac{N}{16} = 1236.64 \text{ clocks per loop}$$

Subtract out 34 clocks per loop for counting and you have 1202.6

It takes 1202 assembler instructions to do a floating point multiply

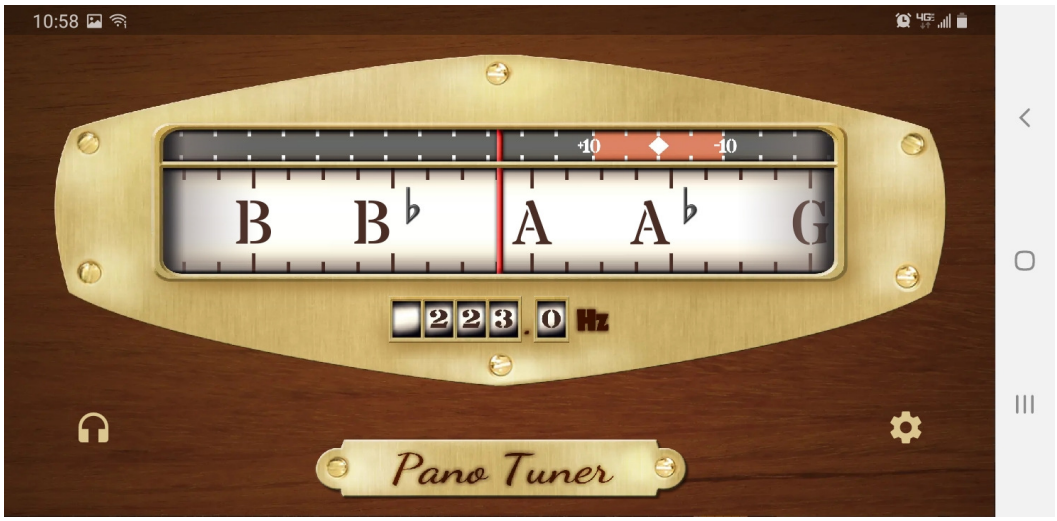
1d) Floating Point Square Root

```
float A, B, C;
A = sqrt(3);
B = sqrt(2);
while(1) {
    C = sqrt(A);
    PORTC += 1;
    }
```

Te frequency is 223.0Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 22{,}421.52$$

It takes 22,421 clocks to find the square root



Stoplight in C

2) Write a C program which turns your PIC into a stoplight:

	7	6	5	4	3	2	1	0
PORTC (E/W)	-	-	R	R	Y	Y	G	G
PORTD (N/S)	-	-	R	R	Y	Y	G	G

The stoplight cycles every 14 seconds

Seconds	E/W	N/S
5 seconds	G	R
2 seconds	Y	R
5 seconds	R	G
2 seconds	R	Y

```
// Subroutine Declarations
#include <pic18.h>

void Wait(unsigned int X)
{
   unsigned int i, j;
   for (i=0; i<X; i++)
      for(j=0; j<617; j++);
   }

// Main Routine

void main(void)
{
   unsigned int TIME;

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

   while(1) {
         PORTA = TIME;
         if(TIME == 0) {
               PORTC = 0x03;
               PORTD = 0x30;
               }
         if(TIME == 5) {
               PORTC = 0x0C;
               PORTD = 0x30;
               }
         if(TIME == 7) {
               PORTC = 0x30;
               PORTD = 0x03;
               }
         if(TIME == 12) {
               PORTC = 0x30;
               PORTD = 0x0C;
               }
         TIME = (TIME + 1) % 14;
         Wait(1000);
        }
   }
```

3) Verify your program runs on your PIC board

- Include the size of the compiled C code
- Check the timing by observation (an oscilloscope would be better...)

```
Memory Summary:
    Program space        used   212h (   530) of 10000h bytes   (  0.8%)
    Data space           used     9h (     9) of   F80h bytes   (  0.2%)
    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%)
```

Resulting code was 265 lines of assembly (530 bytes)

In comparison, the assembler code was 72 lines of code

The C program was 268% larger than the assembler code

but much easier to write

Roulette!

Problem 4-8) Turn your PIC board into a Roulette wheel

Problem 4) Display Routine

Write a C program in C which

- Is passed a number from 0..7
- The routine displays the number on the LCD display, and
- It light up RCx where x is the number (0..7)

Check your subroutine

```
// Global Variables

const unsigned char MSG0[20] = "N:              ";
const unsigned char MSG1[20] = "Bank:           ";

// Subroutine Declarations
#include <pic18.h>

// Subroutines
#include        "lcd_portd.c"

void Display(unsigned int BANK, unsigned int BALL)
{
  LCD_Move(0,8);  LCD_Out(BALL, 1, 0);
  LCD_Move(1,8);  LCD_Out(BANK, 3, 0);
  if(BALL == 0) PORTC = 1;
  if(BALL == 1) PORTC = 2;
  if(BALL == 2) PORTC = 4;
  if(BALL == 3) PORTC = 8;
  if(BALL == 4) PORTC = 0x10;
  if(BALL == 5) PORTC = 0x20;
  if(BALL == 6) PORTC = 0x40;
  if(BALL == 7) PORTC = 0x80;
  }

// Main Routine

void main(void)
{
  unsigned int BANK, BALL;
  unsigned int i, j;

  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]);
  LCD_Move(1,0);  for(i=0; i<16; i++) LCD_Write(MSG1[i]);

   BANK = 10;
   BALL = 3;

   while(1) {
      Display(BANK, BALL);
      Wait_ms(100);
      }
   }
```

Problem 5) Random Number Generator.

Program your PIC board to generate a random number in the range of 0..7 every time you press and release RB0.

- Display this number on the LCD and on PORTC

Generate 5+ random numbers and check your random number generator works.

```
:
:

unsigned int Spin_Wheel(void)
{
   unsigned int N;
   while(!RB0);
   while(RB0) N = (N + 1) % 8;
   return(N);
   }

// Main Routine

   BANK = 100;
   BALL = 3;

   while(1) {
      BALL = Spin_Wheel();
      Display(BANK, BALL);
      Wait_ms(100);
      }
   }
```

Results: {5, 1, 3, 4, 2, 1, 6, 3, 7, 6, 5, 1, 4}

The numbers look random in the range of 0..7

Problem 6) Spin the Wheel

Modify this code so that each time you press RB0

- You generate a random number from 0..7
- You set a counter to N where N = 32 + the random number

You then start counting down to zero

- Each count is 200ms
- Each count the ball moves one position. (if the ball moves to position #8, it goes back to #0)
- Display the ball position on the LCD and on PORTC

Check you code

```
// Main Loop

   BANK = 100;
   BALL = 3;

   while(1) {
      N = Spin_Wheel();
      for(i=0; i<32+N; i++) {
         BALL = (BALL + 1) % 8;
         Display(BANK, BALL);
         Wait_ms(200);
         }
      Display(BANK, BALL);
      Wait_ms(1000);
      }
```

When you press and release RB0

- N counts mod 8
- A light shows up on PORTC
- After four rotations, the light and number stop
- Each count is 200ms (approx)

Problem 7) Winning Numbers

Modify the code so that after N steps, you check if you won or not.

- If the ball ends up in position #7 (lucky 7), you win and your bank value is increased by $8
- Otherwise, you lose and your bank value is decreased by $1.

Check your code to see that you win on seven and lose otherwise.

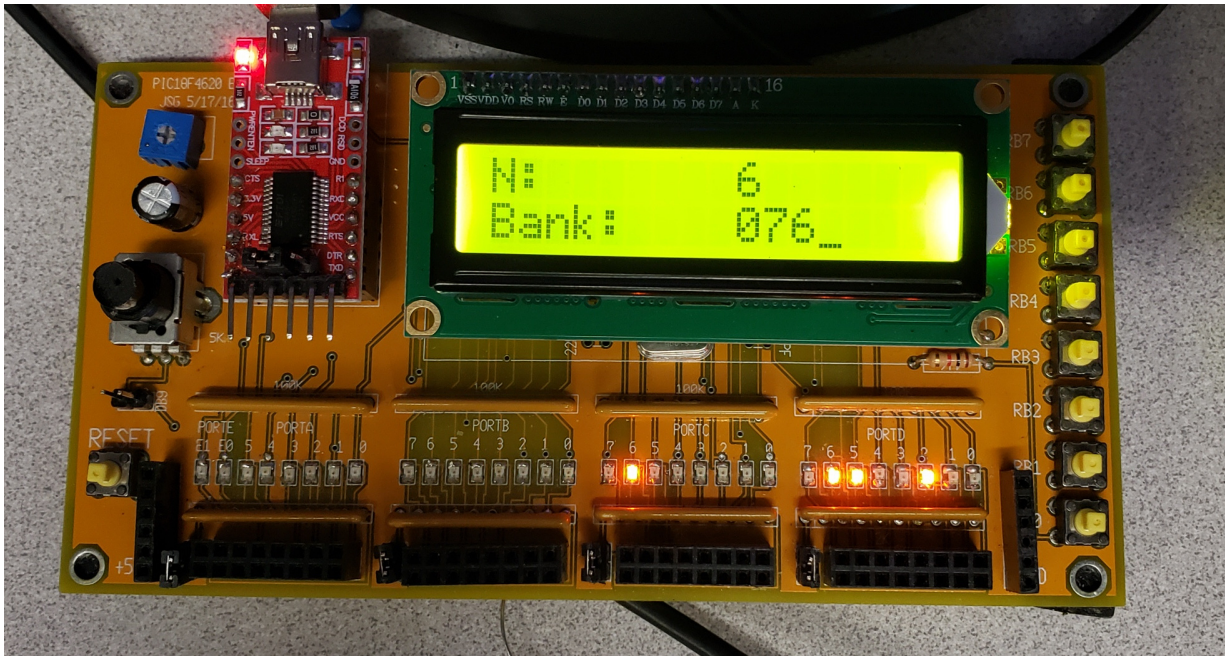
```
while(1) {
    N = Spin_Wheel();
    for(i=0; i<32+N; i++) {
       BALL = (BALL + 1) % 8;
       Display(BANK, BALL);
       Wait_ms(200);
       }

    if(BALL == 7)
        BANK += 8;
    else
        BANK -= 1;

    Display(BANK, BALL);
    Wait_ms(1000);
    }
```

Check:

- When 7 comes up, the bank increases by 8
- For other numbers, the bank drops by 1



Problem 8) Beep

Finally, modify your code so that a speaker beeps every count

- Frequency = 200Hz
- Duration = 50ms (20 toggles)

Beep Routine:

```
void Beep(void) {
   unsigned int i, j;
   for(i=0; i<20; i++) {
      RC0 = !RC0;
      for(j=0; j<1558; j++);
      }
   }
```

Test Code:

```
while(1) {
    Beep();
 }
```

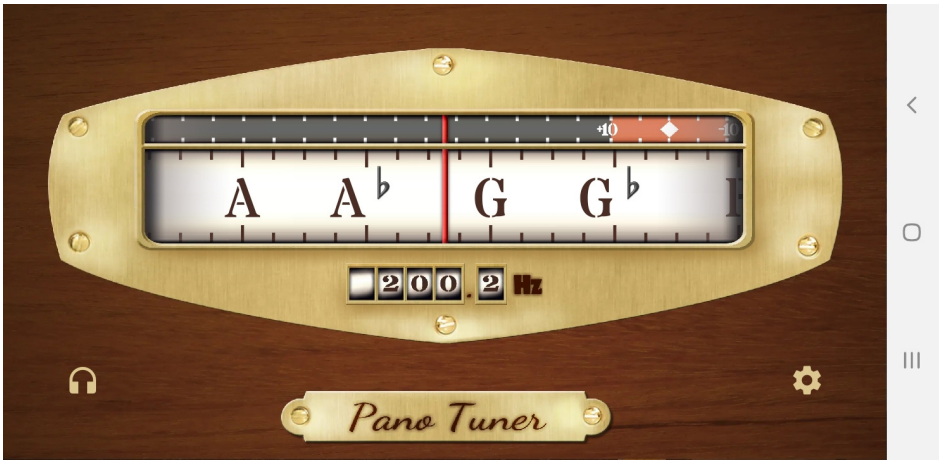
Result: 200.2Hz

Final Main Routine:

```
while(1) {
   N = Spin_Wheel();
   for(i=0; i<32+N; i++) {
      BALL = (BALL + 1) % 8;
      Display(BANK, BALL);
      Beep();
      Wait_ms(100);
      }

   if(BALL == 7)
       BANK += 8;
   else
       BANK -= 1;

   Display(BANK, BALL);
   Wait_ms(1000);
   }
```



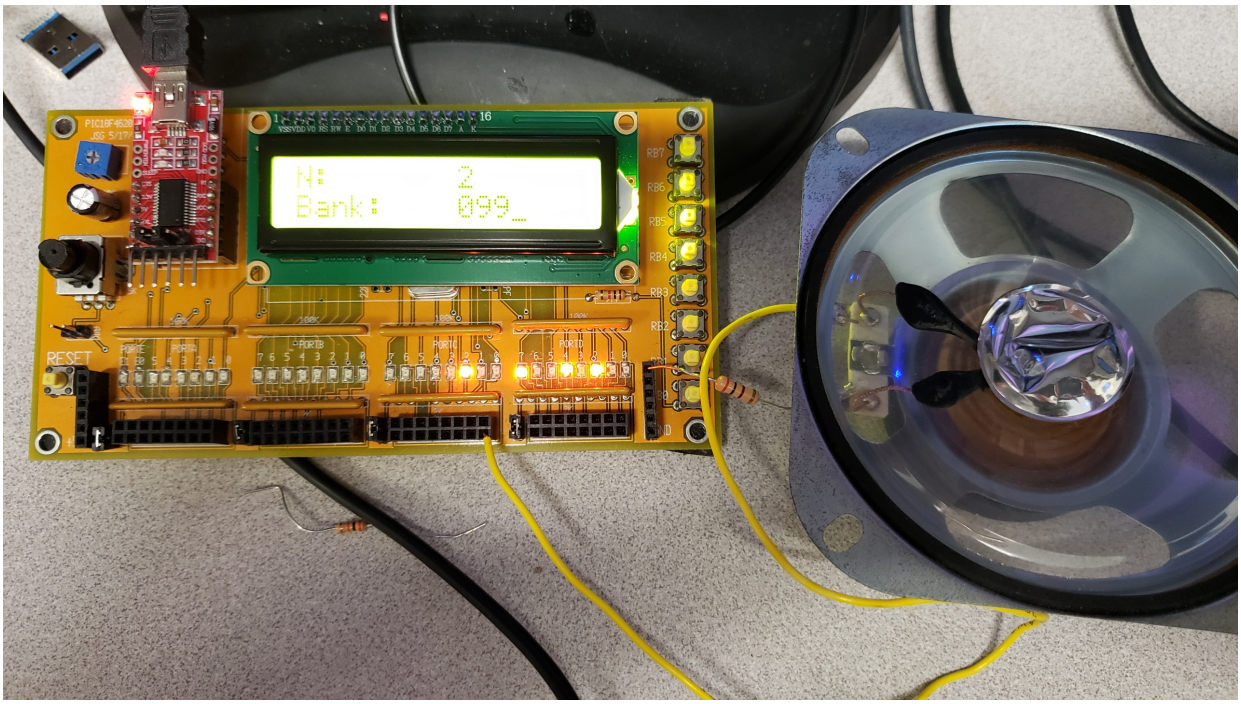
Problem 9) Demo (20 pt)

Demonstrate your Roulette wheel

Final Code:

- Note: a PIC can do a lot more than run a roulette wheel

```
Memory Summary:
    Program space        used   C10h (  3088) of 10000h bytes   (  4.7%)
    Data space           used    31h (    49) of   F80h bytes   (  1.2%)
    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%)
```



Final Code:

```
// Global Variables

const unsigned char MSG0[20] = "N:              ";
const unsigned char MSG1[20] = "Bank:           ";

// Subroutine Declarations
#include <pic18.h>

// Subroutines
#include        "lcd_portd.c"


void Beep(void) {
   unsigned int i, j;
   for(i=0; i<20; i++) {
      RC0 = !RC0;
      for(j=0; j<1558; j++);
      }
   }

void Display(unsigned int BANK, unsigned int BALL)
{
  LCD_Move(0,8);  LCD_Out(BALL, 1, 0);
  LCD_Move(1,8);  LCD_Out(BANK, 3, 0);
  if(BALL == 0) PORTC = 1;
  if(BALL == 1) PORTC = 2;
  if(BALL == 2) PORTC = 4;
  if(BALL == 3) PORTC = 8;
  if(BALL == 4) PORTC = 0x10;
  if(BALL == 5) PORTC = 0x20;
  if(BALL == 6) PORTC = 0x40;
  if(BALL == 7) PORTC = 0x80;
  }

unsigned int Spin_Wheel(void)
{
   unsigned int N;
   while(!RB0);
   while(RB0) N = (N + 1) % 8;
   return(N);
   }

// Main Routine

void main(void)
{
   unsigned int BANK, BALL, N, i, j;

   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]);
   LCD_Move(1,0);  for(i=0; i<16; i++) LCD_Write(MSG1[i]);

   BANK = 100;
   BALL = 3;
```

```
while(1) {
   N = Spin_Wheel();
   for(i=0; i<32+N; i++) {
      BALL = (BALL + 1) % 8;
      Display(BANK, BALL);
      Beep();
      Wait_ms(100);
      }

   if(BALL == 7)
       BANK += 8;
   else
       BANK -= 1;

   Display(BANK, BALL);
   Wait_ms(1000);
   }
}
```