ECE 376 - Homework #4

C Programming and LCD Displays. Due Monday, February 14th

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 32

- note: if using your cell phone to measure the frequency, you might have to try different pins on PORTC until you get one in the audio range. Each pin is 1/2 the frequency of the previous pin

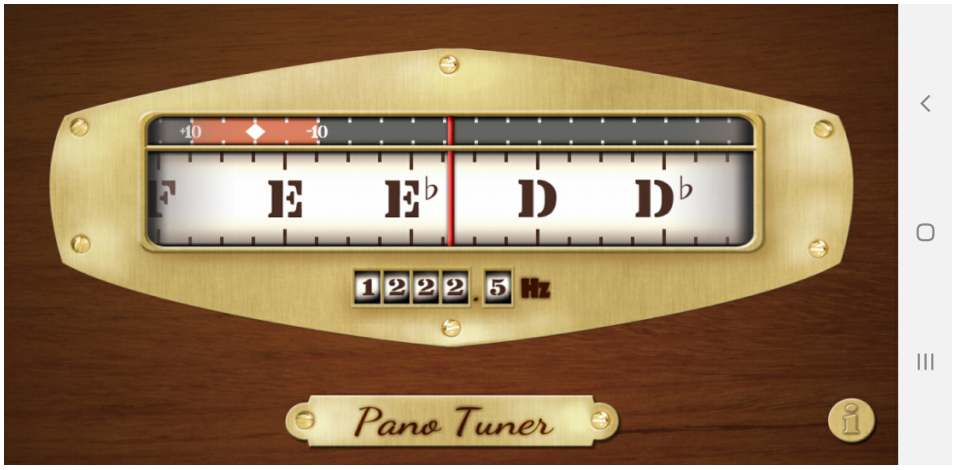
```
unsigned char i
while(1) {
      i = (i + 1) % 32;
      if(i == 0) PORTC += 1;
      }
```

RC3 = 1222.5Hz

- RC0 = 8 x RC3 = 9780Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot 9780Hz}\right) \cdot \left(\frac{1}{32}\right) = 15.97 \text{ clocks}$$

It takes 16 clocks to count mod 32



1b) Counting mod 33

```
unsigned char i
while(1) {
      i = (i + 1)% 33;
      if(i == 0) PORTC += 1;
      }
```

RC0 = 268.4Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot 268.4Hz}\right) \cdot \left(\frac{1}{33}\right) = 564.5 \text{ clocks}$$

It takes 564 clocks to count mod 33

1c) Long Integer Addition

```
unsigned long int A, B, C;
unsigned char i;
A = 0x12345678;
B = 0;
while(1) {
    i = (i + 1)% 32;
    if (i == 0) PORTC += 1;
    B = B + A;
    }
```

RC0 = 3193.6Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot 3193.6Hz}\right) \cdot \left(\frac{1}{32}\right) = 48.92 \approx 49 \text{ clocks}$$

subtract 16 (the time to count mod 32) and you get 33 clocks

A long integer addition takes 49 clocks to execute

1d) Floating point addition

```
float A, B;
A = 3.14159265379;
B = 0;
while(1) {
    i = (i + 1)% 32;
    if(i == 0) PORTC += 1;
    B = B + A;
    }
```

RC0 = 154.3Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot 154.3Hz}\right) \cdot \left(\frac{1}{32}\right) = 1012.6 \text{ clocks}$$

subtract 16 clocks (the time to count mod 32) and you get 996.6 clocks

It takes 996.6 clocks to add a floating point number

$65 Egg Timer

2) Write a C program which turns your PIC into an egg timer with a resolution of 100ms

- TIME is displayed on the LCD display as XXX.X seconds
- On reset, TIME = 0000.0
- When RB0 is pressed, TIME is set to 5.0 seconds
- When RB1 is pressed, TIME is set to 10.0 seconds
- When TIME > 0, PORTC = 0xFF. When TIME == 0, PORTC = 0x00.
- Every 100ms, TIME is decremented by 0.1 second and displayed, stopping at zero

Partial Code:

```
LCD_Init();                  // initialize the LCD

SEC = 0;

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

TIME = 0;

while(1) {
   if(RB0) TIME = 50;
   if(RB1) TIME = 100;

   LCD_Move(1,0);  LCD_Out(TIME, 3, 1);
   RA1 = 1;
   Wait_ms(85);
   RA1 = 0;
   if(TIME) TIME -= 1;
   }
```

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

The compiled code takes up 2308 bytes of ROM

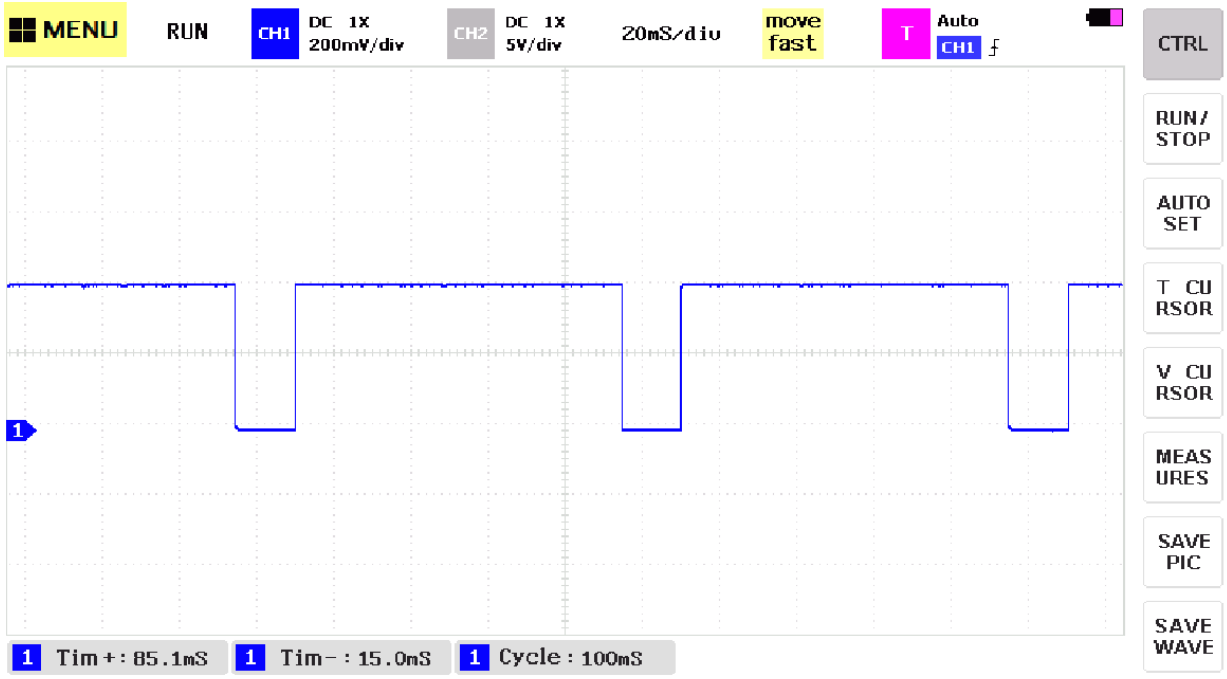
- Each instruction takes 2 bytes
- **1154 lines of assembler**

```
Memory Summary:
    Program space        used   904h (  2308) of 10000h bytes   (  3.5%)
    Data space           used    29h (    41) of   F80h bytes   (  1.0%)
    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 determine how accurate your program is (one count = 100ms ideally)

Measure the signal on RA1 (the reason for those lines of code)

- Period = 100ms
- Wait routine takes 85ms (when RA1 = 1)
- The rest of the code takes 15ms (when RA1 = 0)



PIC 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.