

ECE 376 - Homework #4

C Programming, LCD Displays

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 oscilloscope - 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 16

- 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) % 16;
    if(i == 0) PORTC += 1;
}
```

Data: Measured frequency is 19.54kHz. With 16 loops per toggle, this gives

$$16N = \left(\frac{10,000,000}{2 \cdot Hz} \right)$$

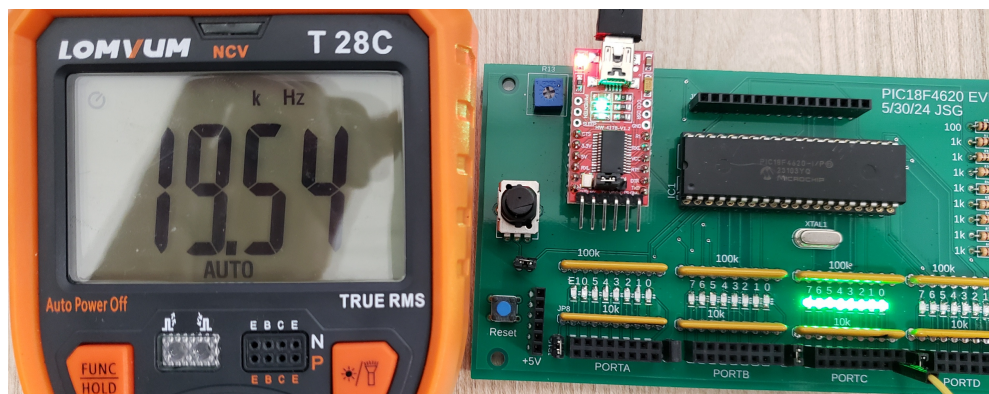
$$N = 15.9928 \approx 16$$

It takes 16 clocks to count mod 16

In terms of compiled code size, this code consists of 90 bytes (45 lines of assembly)

Memory Summary:

Program space	used	5Ah (90) of 10000h bytes (0.1%)
Data space	used	2h (2) of F80h bytes (0.1%)
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%)



1b) Counting mod 17

```
unsigned char i
while(1) {
    i = (i + 1)% 17;
    if(i == 0) PORTC += 1;
}
```

Changing this to counting mod 17 increased the code size from 90 bytes to 394 bytes (187 lines of assembly)

Memory Summary:

Program space	used	18Ah (394)	of 10000h bytes (0.6%)
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%)

The frequency on RC0 drops to 482.6Hz. The time for each pass (17 passes per toggle) is:

$$17N = \left(\frac{10,000,000}{2 \cdot \text{Hz}} \right)$$

$$N = 609.44 \approx 609$$

It takes 609 clocks to count mod 17



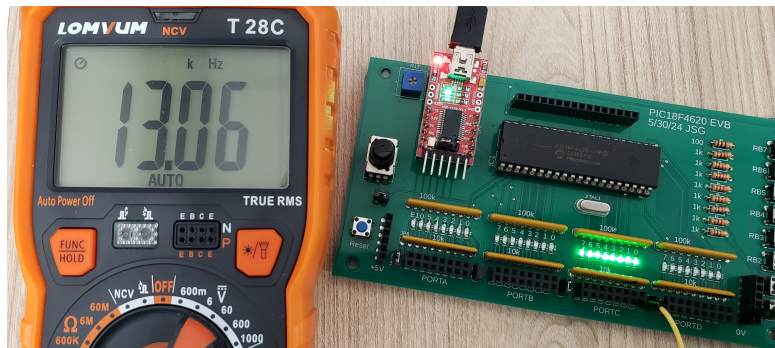
1c) Integer Multiply

```
unsigned int A, B, C;  
unsigned char i;  
A = 0x0432;  
B = 0x2468;  
while(1) {  
    PORTC += 1;  
    C = A*B;  
}
```

RC0 outputs 13.06kHz, meaning each integer multiply is

$$N = \left(\frac{10,000,000}{2 \cdot \text{Hz}} \right) = 382.8 \approx 383$$

It takes 383 clocks to do an integer multiply



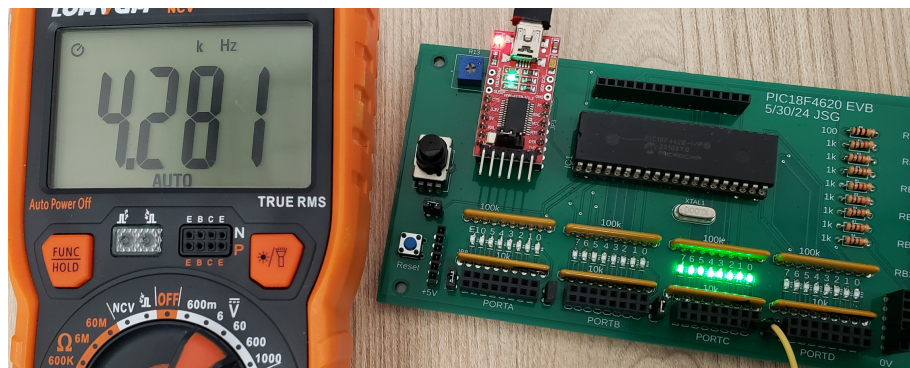
1d) Floating point multiply

```
float A, B;  
A = 1.01;  
B = 0.00123;  
while(1) {  
    PORTC += 1;  
    B = B * A;  
}
```

RC0 outputs 4281Hz, meaning

$$N = \left(\frac{10,000,000}{2 \cdot \text{Hz}} \right) = 1167.95 \approx 1168$$

It takes 1168 clocks to do a floating point multiply



LED Flashlight & LCD Displays

Write a C program which turns your PIC board into an LED flashlight

Input:

- Push buttons on PORTB

Output:

- LCD display
- Lights on PORTC

Relationship: The buttons control the brightness of the LEDs

- RB0: 0% on
- RB1: 1/7 duty cycle (14% on)
- RB2: 2/7 duty cycle (28% on)
- :
- RB7: 7/7 duty cycle (100% on)
- Frequency of lights = 50Hz (20ms cycle time)

The LCD display should show the current duty cycle

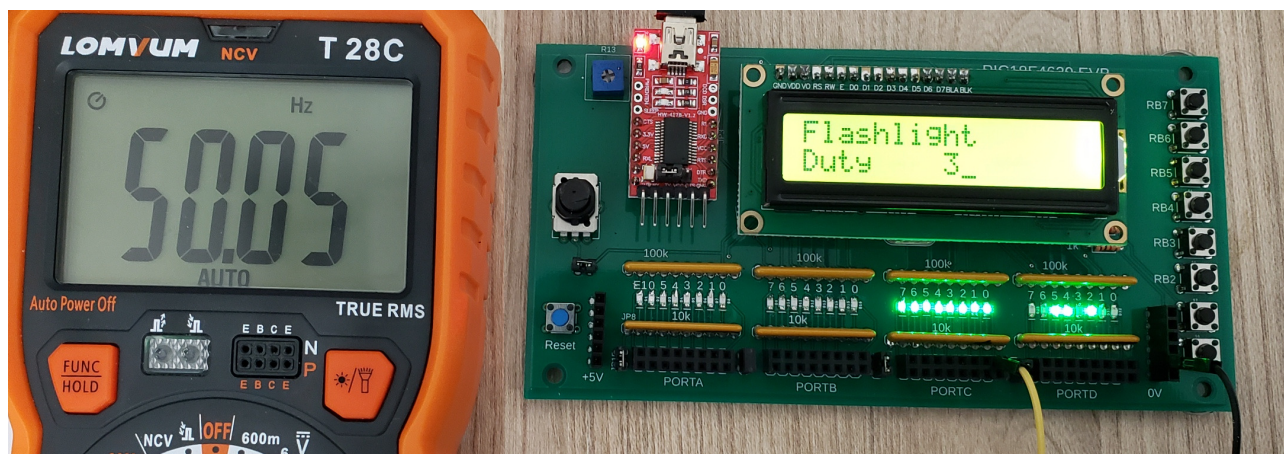
The lights on PORTC are all ON or OFF - for the brightness of the flash light.

Results:

Validation - Frequency:

- Button Pressed: 10.61Hz
- Button Released: 50.05Hz

note: The LCD routines are activated when the button is pressed. These add too much delay in the loop.

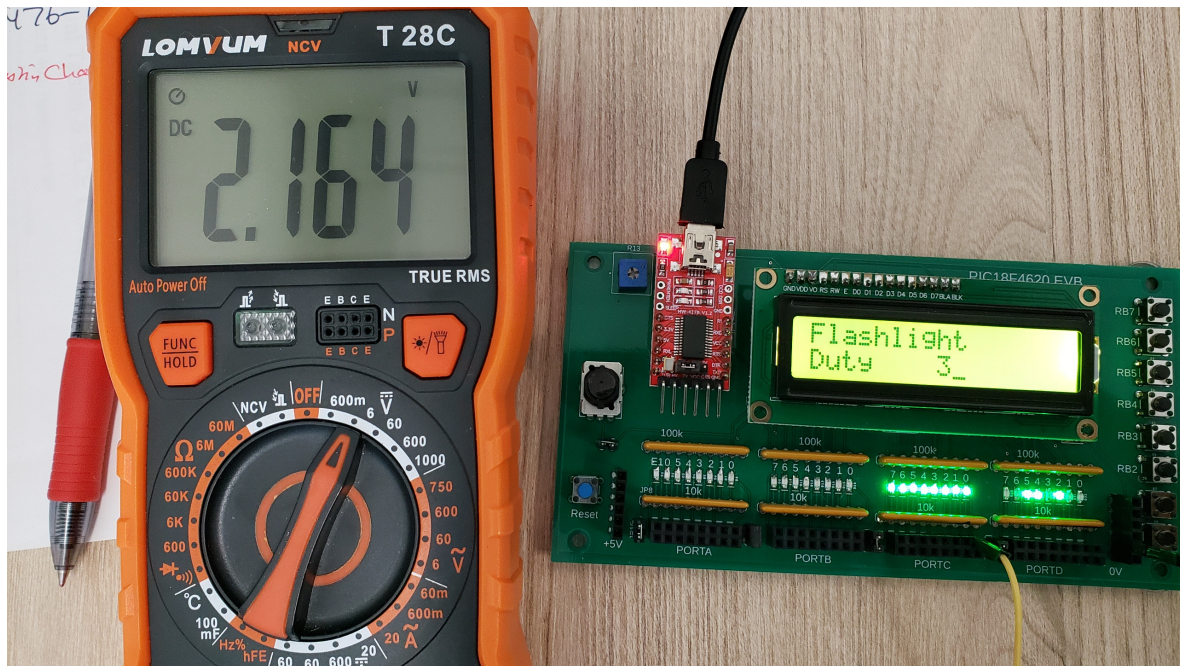


Validation - Duty Cycle (Voltage)

To verify the duty cycle, the DC voltage was measured.

- The DC voltage should be proportional to the duty cycle
 - $DC = \text{average}$
- The AC voltage is a measure of the ripple
 - $AC = \text{ripple (related to } V_{pp})$
 - V_{rms} doesn't tell you much...

Button	0	1	2	3	4	5	6	7
Duty	0/7	1/7	2/7	3/7	4/7	5/7	6/7	7/7
Theory	0V	0.714V	1.429V	2.143V	2.857V	3.571V	4.286V	5.000V
V(DC)	0.002V	0.721V	1.441V	2.164V	2.882V	3.601V	4.321V	5.040V
V(AC)	0.000V	1.742V	2.261V	2.479V	2.479V	2.260V	1.742V	0.000V



Code:

A delay is added to set the frequency to 50.0Hz

- for(i=0; i<1000; i++); results in 85.26Hz
- for(i=0; i<2000; i++); results in 43.62Hz

Adjusting using interpolation and trial and error:

- for(i=0; i<1737; i++); results in 50.05Hz

Code:

```
// Global Variables

const unsigned char msg0[20] = "Flashlight    ";
const unsigned char msg1[20] = "Duty      ";

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

// Subroutines

// Main Routine

void main(void)
{
    unsigned char Duty = 0;
    unsigned char N = 0;
    unsigned int i;

    TRISA = TRISC = TRISD = TRISE = 0;
    TRISB = 0xFF;
    PORTA = PORTC = PORTD = PORTE = 0;
    ADCON1 = 0x0F;

    LCD_Init();
    Wait_ms(100);
    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]);
    N = 0;

    while(1) {
        if(RB0) { Duty = 0; LCD_Move(1,8); LCD_Write('0'); }
        if(RB1) { Duty = 1; LCD_Move(1,8); LCD_Write('1'); }
        if(RB2) { Duty = 2; LCD_Move(1,8); LCD_Write('2'); }
        if(RB3) { Duty = 3; LCD_Move(1,8); LCD_Write('3'); }
        if(RB4) { Duty = 4; LCD_Move(1,8); LCD_Write('4'); }
        if(RB5) { Duty = 5; LCD_Move(1,8); LCD_Write('5'); }
        if(RB6) { Duty = 6; LCD_Move(1,8); LCD_Write('6'); }
        if(RB7) { Duty = 7; LCD_Move(1,8); LCD_Write('7'); }

        N = (N + 1) % 7;

        if(N >= Duty) PORTC = 0;
        else PORTC = 0xFF;

        for(i=0; i<1737; i++);
    }
}
```

Compilation Results

- The resulting code size is 2114 bytes
- = 1057 lines of assembly

Memory Summary:

Program space	used	842h (2114)	of 10000h bytes	(3.2%)
Data space	used	10h (16)	of F80h bytes	(0.4%)
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%)