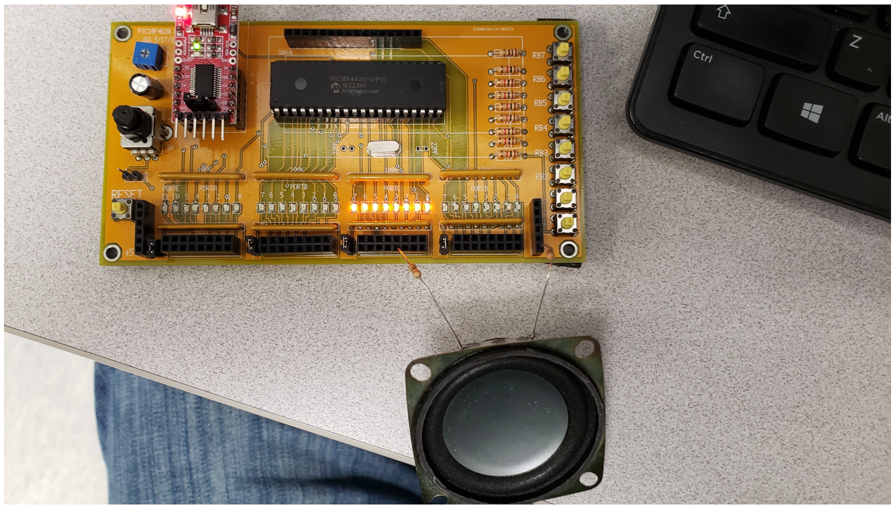
ECE 376 - Homework #4

C Programming & LCD Displays - Due Monday, September 30th

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

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



Speaker connectted to PORTC. Each pin is divide by 2

1a) Counting mod 32

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

f = 1217.8Hz on RC3

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 4105.76$$

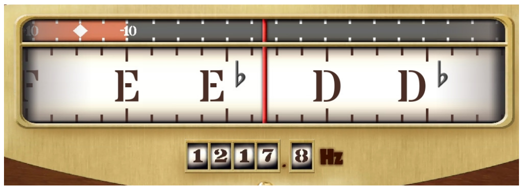
On RC0, the frequency is 8x higher (N = 8x smaller)

$$N/8 = 513.22$$

Divide by 32 to get the time per loop (toggles every 32nd pass)

$$N/8/32 = 16.04$$

It takes 16 clocks to count mod 32



1217.8Hz for counting mod 32 on RC3

1b) Counting mod 35

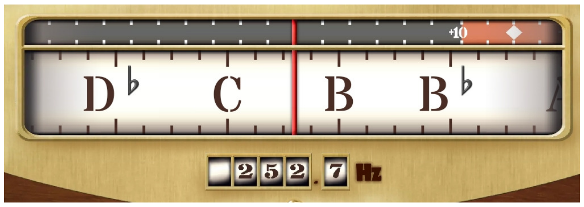
```
unsigned char i
while(1) {
      i = (i + 1)% 35;
      if(i == 0) PORTC += 1;
      }
```

RC0 plays 252.2Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 19{,}786 \text{ clocks}$$

$$N/35 = 565.32$$

It takes 565 clocks to count mod 35



1c) Long Integer Division

```
unsigned long int A, B, C;
A = 123456789;
B = 2731;
while(1) {
    i = (i + 1)% 32;
    if(i == 0) PORTC += 1;
    C = A / B;
    }
```

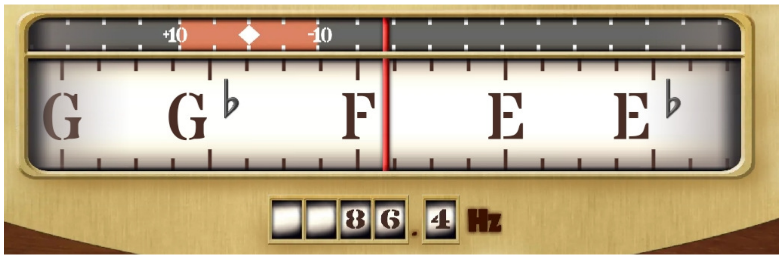
RC0 plays 86.4Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz}\right) = 57{,}870$$

$$N/32 = 1{,}808$$

16 clocks ere for counting mod 32. The remainder are for long integer division

It takes 1792 clocks to do a long integer divide



Long integer division

1d) Floating Point Division

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

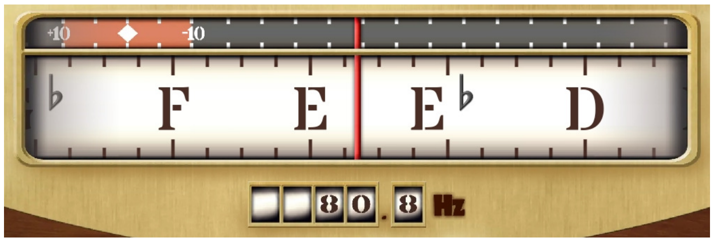
RC0 plays 80.8Hz

$$N = \left(\frac{10{,}000{,}000}{2 \cdot Hz} \right) = 61{,}881$$

$$N/32 = 1934$$

$$N/32 - 16 = 1{,}918$$

It takes 1918 clocks to do a floating point division



floating point division

Note: In C, it often is easiest to find the number of clocks experimentally:

- Toggle a pin within your program as you run it
- Measure the frequency on that pin

Lights-Out Game in C

2) Write a C program which allows you to play the lights-out game from HW #3

- On power up, PORTC = 0xFF and PORTD = 0x00
- When you press and release a button, the corresponding pin on PORTC and its neighbors are toggeled
  - RB0: Toggle pins RC0, RC1
  - RB1: Toggle pins RC0, RC1, RC2
  - etc.
- Each time you press and release a button, PORTD increments by one

Code:

```
// Global Variables

// Subroutine Declarations
#include <pic18.h>

// Subroutines

// Main Routine

void main(void)
{
    unsigned char i;

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

// start with a random value in PORTC
    while(!RB0) PORTC += 1;

    while(PORTC) {
       while(PORTB == 0);

       if(RB0) PORTC = PORTC ^ 0x03;
       if(RB1) PORTC = PORTC ^ 0x07;
       if(RB2) PORTC = PORTC ^ 0x0E;
       if(RB3) PORTC = PORTC ^ 0x1C;
       if(RB4) PORTC = PORTC ^ 0x38;
       if(RB5) PORTC = PORTC ^ 0x70;
       if(RB6) PORTC = PORTC ^ 0xE0;
       if(RB7) PORTC = PORTC ^ 0xC0;

       PORTD += 1;
       while(PORTB);
       }
   while(1);

   }
```

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    DAh (   218) of 10000h bytes   (  0.3%)
    Data space           used     1h (     1) of   F80h bytes   (  0.0%)
    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%)
```

Note:

- The assembler version took up 56 lines of assembler
- The C version produces 109 lines of assembler

C is 94% larger than asembler

But, C was a *lot* easier to write.

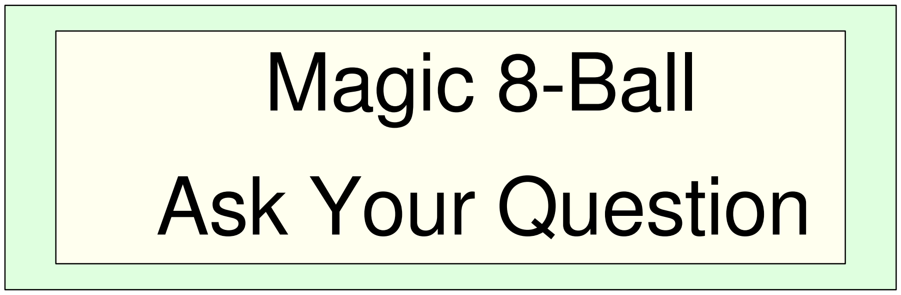
The code works

- On reset, it starts with a random value in PORTC
- When you press and release a button, PORTD counts by one
- When you press and release a button, the corresponding lights on PORTC toggle

LCD Display & Magic 8-Ball!

Problem 4-8) Turn your PIC board into a Magic 8-Ball:

- On power up, the Magic 8-Ball prompts you to ask a question
- You then ask your PIC board a question
- Shake the Magic 8-Ball three times (press RB0 three times)
- The answer to your question is then displayed on the LCD with one of 12 random fortunes:
  - It is certain, It is decidedly so, Without a doubt, Yes definately
  - Reply hazy try again, Ask again later, Better not tell you now, Cannot predict now
  - Dont count on it, My reply is no, Outlook not so good, Very doubtful
- Five seconds after your fortune is revealed, the


Magic 8-Ball
Ask Your Question

Problem 4) Display Routine

Write a subroutine in C which

- Is passed a number from 0..11
- Displays one of twelve messages based upon the number passed

Check your subroutine

Code:

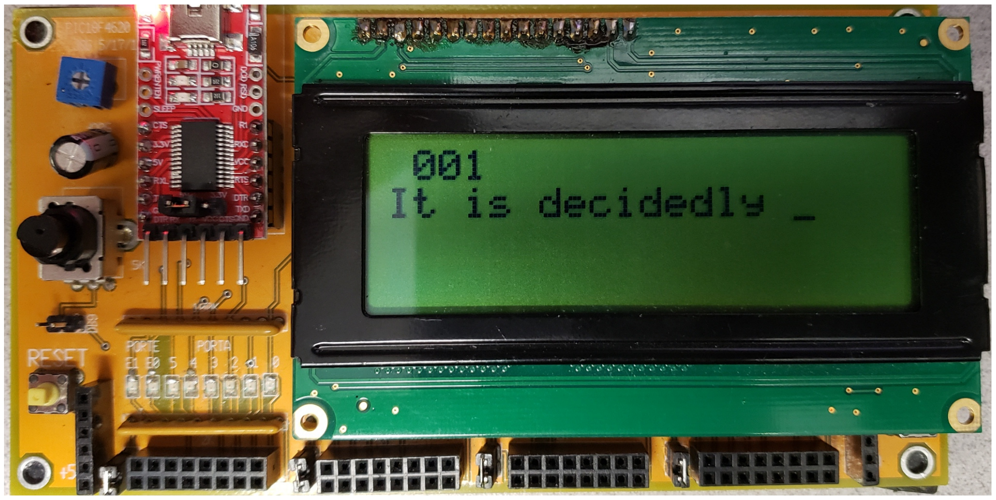
```
// Global Variables
unsigned char msg0[17] = " Magic 8 Ball   ";
unsigned char msg1[17] = "Ask a question  ";
unsigned char f0[17] = "It is certain   ";
unsigned char f1[17] = "It is decidedly ";
unsigned char f2[17] = "Without a doubt ";
unsigned char f3[17] = "Yes definately  ";
unsigned char f4[17] = "hazy try again  ";
unsigned char f5[17] = "Ask again later ";
unsigned char f6[17] = "Cant tell you   ";
unsigned char f7[17] = "Cannot predict  ";
unsigned char f8[17] = "Dont count on it";
unsigned char f9[17] = "My reply is no  ";
unsigned char f10[17] = "Outlook not good";
unsigned char f11[17] = "Very doubtful   ";


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

// Subroutines

void Fortune(unsigned char n) {
    unsigned char i;
    LCD_Move(1,0);
    for(i=0; i<16; i++) {
      if(n==0) LCD_Write(f0[i]);
      if(n==1) LCD_Write(f1[i]);
      if(n==2) LCD_Write(f2[i]);
      if(n==3) LCD_Write(f3[i]);
      if(n==4) LCD_Write(f4[i]);
      if(n==5) LCD_Write(f5[i]);
      if(n==6) LCD_Write(f6[i]);
      if(n==7) LCD_Write(f7[i]);
      if(n==8) LCD_Write(f8[i]);
      if(n==9) LCD_Write(f9[i]);
      if(n==10) LCD_Write(f10[i]);
      if(n==11) LCD_Write(f11[i]);
      }
    }
```

Testing the code

```
// Main Routine

void main(void) {
  unsigned char i, n;

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

  LCD_Init();

  for(i=0; i<12; i++) {
     LCD_Move(0,0);  LCD_Out(i, 3, 0);
     Fortune(i);
     Wait_ms(1000);
     }
   while(1);
  }
```

This results in the display

- Counting from 0 to 11 (line 1)
- Displaying your fortune (line 2)



```
 001
It is decidedly _
```

Problem 5) Random Number Generator.

Program your PIC board to generate a random number in the range of 0..11 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.

Numbers = 3, 5, 8, 8, 2

Looks good

```
while(1){
  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]);

  Wait_ms(10);
  while(!RB0);
  Wait_ms(10);
  while(RB0) n = (n+1)%12;

  LCD_Move(1,0);  LCD_Out(n, 3, 0);
  Wait_ms(1000);
  }
```

Problem 6) Count to Three

Modify this code so that every third time you press and release RB0

- You generate a random number from 0..11
- A fortune is revealed based upon the random number

```
for(i=0; i<3; i++) {
   Wait_ms(10);
   while(!RB0);
   Wait_ms(10);
   while(RB0) n = (n+1)%12;
   }
```

Problem 7) Five Second Delay

Modify the code so that after you press RB0 three times

- The program pauses for 5.0 seconds, then
- Starts over, prompting you to ask a question

```
while(1){
  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]);

  for(i=0; i<3; i++) {
     Wait_ms(10);
     while(!RB0);
     Wait_ms(10);
     while(RB0) n = (n+1)%12;
     }

  Fortune(n);
  Wait_ms(5000);
  }
```

Problem 8) Demo (20 pt)

Demonstrate your Magic 8-Ball

Test Cases:

Are the Vikings going to win this weekend?

Yes, definately

Will the Vikings make the playoffs?

Cannot predict

Will the Bison score two touchdowns this weekend?

My reply is no