ECE 376 - Homework #4

C - LCD Displays - Keypads

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 256

- 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)% 256;
      if(i == 0) PORTC += 1;
      }
```

f = 1302.8Hz

- N = 3837.89 clocks
- N/ 256 = 14.992 (15)

It takes 15 clocks to count mod 256

1b) Counting mod 255

```
unsigned char i
while(1) {
      i = (i + 1)% 255;
      if(i == 0) PORTC += 1;
      }
```

f = 41.1 Hz

- N = 121,654 clocks
- N / 255 = 477.07

It takes 477 clocks to count mod 255

1c) Integer Multiply

```
unsigned int A, B, C;
unsigned char i;
A = 0x1234;
B = 0x5678;
while(1) {
     i = (i + 1)% 256;
     if(i == 0) PORTC += 1;
     C = A*B;
     }
```

f = 42.3Hz

- N = 118,203
- N / 256 = 461.7
  - 15 clocks to count mod 256
  - plus 447 clocks to do an integer multiply

It taekes 467 (ish) clocks to do an integer multiply

1d) Floating point multiply

```
float A, B;
A = 1.0002;
B = 0.02;
while(1) {
    i = (i + 1)% 256;
    if(i == 0) PORTC += 1;
    B = B * A;
    }
```

f = 85.3Hz

- N = 58,616.6
- N / 256 = 228.97 (229 clocks)
  - 15 clocks to count mod 256
  - plus 214 clocks to do a floating point multiply

Master-Mind

In the game of Master-Mind, you try to guess a secret code

At the start of the game, a random 4 digit number is selected (each digit can be 0..6)

Each round, you guess what the 4-digit code is

- The computer then looks at the code, digit by digit.
- If the number in the code is also in the guess but wrong spot, the player scores 1 point.
- If the number in the code is also in the guess but right spot, the player scores 10 points.

The game continues until the player gets all 4 digits correct (and scores 40 points)

2) Write a C program which starts the game

- Start the game by pressing RB0
- This generates a random 4-digit number from 0000 to 5555 (all digits 0..5)

Verify your code.

Code

```
while(1) {

// generate code
while(!RB1);
while(RB1) {
   X = (X + 1) % 1296;
}

c0 = X % 6;
X = X / 6;
c1 = X % 6;
X = X / 6;
c2 = X % 6;
X = X / 6;
c3 = X % 6;
CODE = c3*1000 + c2*100 + c1*10 + c0;

LCD_Move(0,6);
LCD_Out(CODE, 4, 0);

}
```

Codes

```
0 4 5 3
0 1 5 2
5 4 4 1
2 3 4 0
```

Comments:

- Each digit is in the range of 0-5
- Results appear to be random

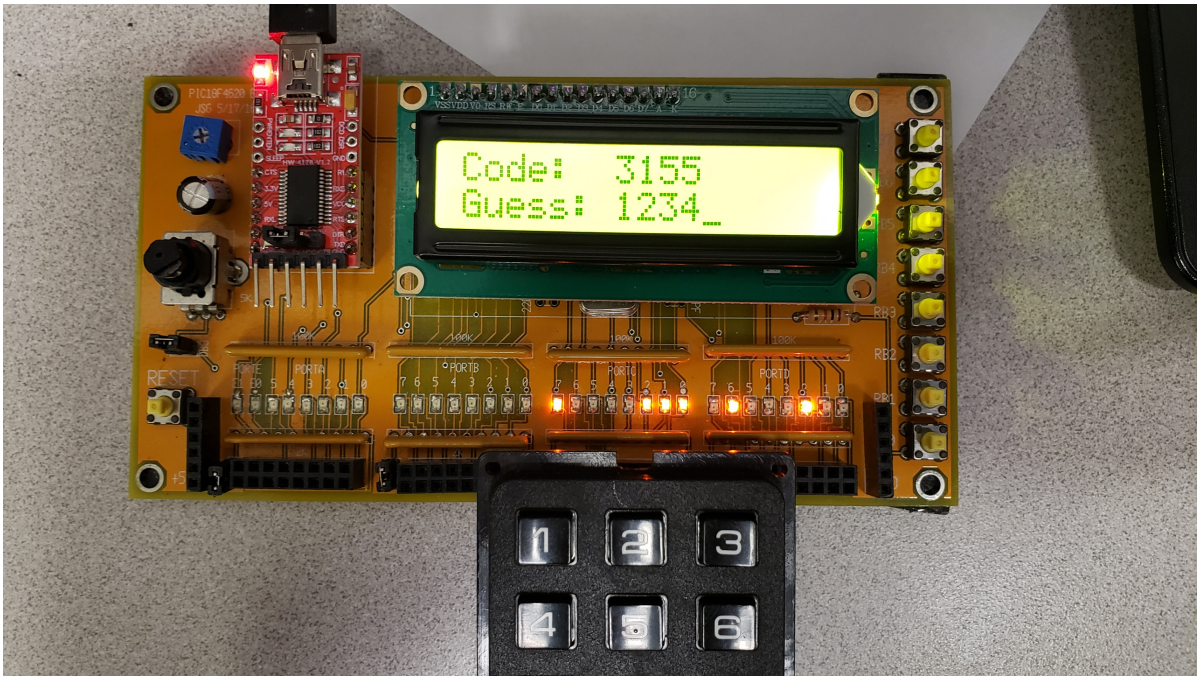
3) Add to this code the user inputting a 4-digit guess using the numeric keypad (0000 to 5555)

- Display the guess on the LCD display
- Verify your code

```
while(1) {
      TEMP = ReadKey();

      if (TEMP < 10) X = (X*10) + TEMP;

      if (TEMP == 10) {
         GUESS = X;
         SCORE = Compute_Score(CODE, GUESS);
         }

      if (TEMP == 11) {
         X = X / 10;
         }

      LCD_Move(1,6);  LCD_Out(X, 4, 0);
      LCD_Move(1,13); LCD_Out(SCORE, 2, 0);

      Wait_ms(100);

      }
```

Comments:

- As you type in numbers, they appear on the screen (X)
- You can remove a number by hitting #
- You can submit your guess by pressing *



4) Add to this code calculations of the player's score

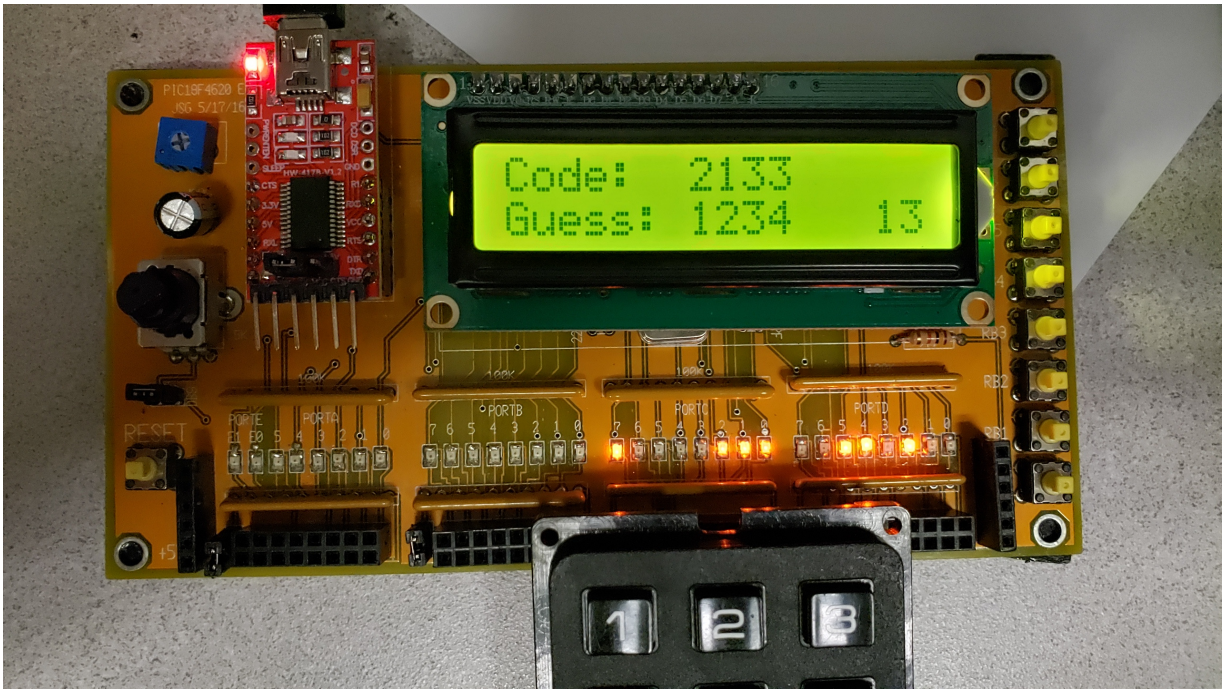
- Check each number one-by-one
- If the number in the code is also in the guess but wrong spot, the player scores 1 point.
- If the number in the code is also in the guess but right spot, the player scores 10 points.

Verify your code

```
unsigned int Compute_Score(unsigned int CODE, unsigned int GUESS) {
   unsigned int SCORE, i, j, p;
   unsigned int C[4], G[4];

   SCORE = 0;

   for(i=0; i<4; i++) {
      C[i] = CODE % 10;
      CODE = CODE / 10;
      G[i] = GUESS % 10;
      GUESS = GUESS / 10;
   }

   for (i=0; i<4; i++) {
      p = 0;
      for(j=0; j<4; j++)
         if(C[i] == G[j])
            p = 1;
      if(C[i] == G[i]) p = 10;
      SCORE = SCORE + p;
      }
   return(SCORE);
   }
```

Comment

- Scoring looks correct (trying several guesses and checking the score)



5) Add to this code a loop where you keep playing until the player scores 40 points

- All 4 digits correct

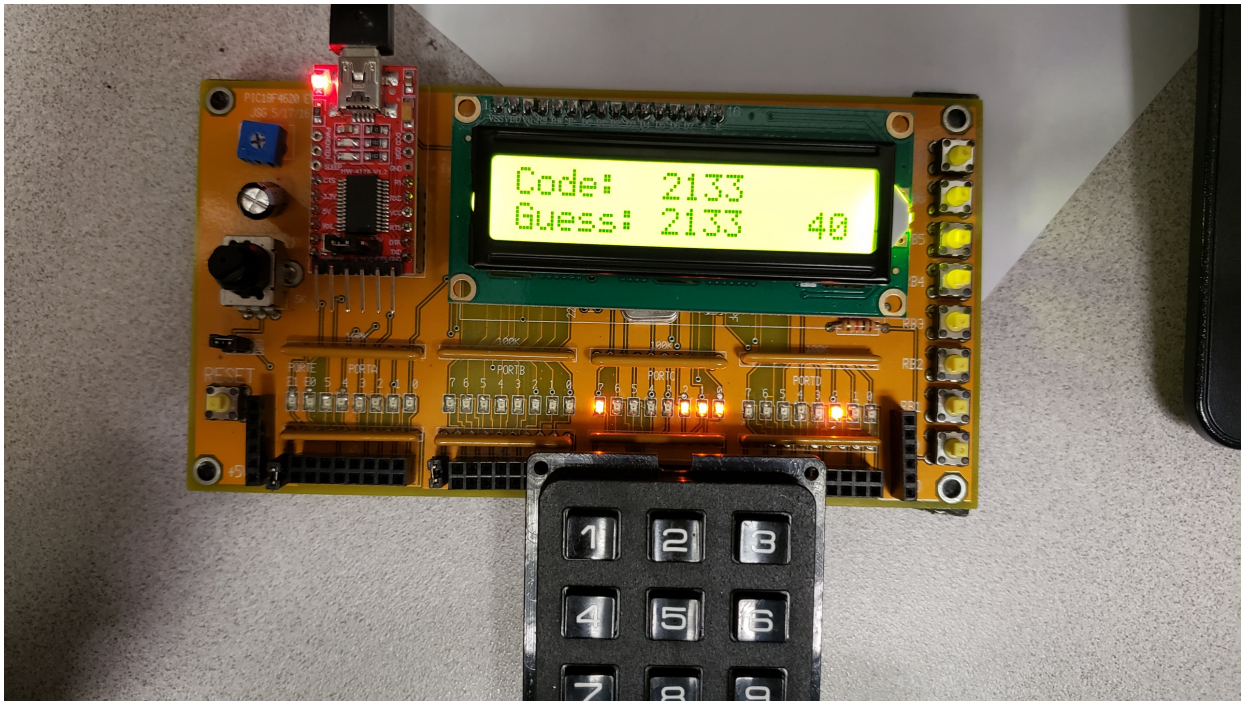
```
while(1) {


 // generate code
    while(!RB1);
    while(RB1) {
       X = (X + 1) % 1296;
    }

    c0 = X % 6;
    X = X / 6;
    c1 = X % 6;
    X = X / 6;
    c2 = X % 6;
    X = X / 6;
    c3 = X % 6;
    CODE = c3*1000 + c2*100 + c1*10 + c0;

    SCORE = 0;
    N = 0;
    while(SCORE < 40) {
       TEMP = ReadKey();

       if (TEMP < 10) X = (X*10) + TEMP;

       if (TEMP == 10) {
          N = N + 1;
          GUESS = X;
          SCORE = Compute_Score(CODE, GUESS);
          }

       if (TEMP == 11) {
          X = X / 10;
          }

       LCD_Move(1,6);  LCD_Out(X, 4, 0);
       LCD_Move(1,13); LCD_Out(SCORE, 2, 0);
       LCD_Move(0,13); LCD_Out(N, 2, 0);

       if(SCORE == 40) {
          LCD_Move(0,6);
          LCD_Out(CODE, 4, 0);
          Wait_ms(2000);
          }
       }

    }
```

Resulting Code: 2937 lines of assembler (5874 bytes)

```
Memory Summary:
    Program space        used  16F2h (  5874) of 10000h bytes   (  9.0%)
    Data space           used    42h (    66) of   F80h bytes   (  1.7%)
    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%)
```

6) Demo your resulting Master-Mind game

- In-person or on a video



notes:

- This is an example of top-down programming
  - Start with the framework of the program
  - Start with the display routine so you can see what's happening
  - Add routines / features one by one
  - Check the routines each step of the way
- C makes this program a *lot* easier to write and debug
- Displaying the code while playing makes debugging easier
- Hiding the code and having you figure it out makes the game more challenging.

Final Code

```
// Master-Mind

// Global Variables

const unsigned char MSG0[21] = "Code:            ";
const unsigned char MSG1[21] = "Guess:           ";


// Subroutine Declarations
#include <pic18.h>

// Subroutines
#include        "lcd_portd.c"

char GetKey(void)
{
   int i;
   unsigned char RESULT;
   TRISC = 0xF8;
   RESULT = 0xFF;
   PORTC = 4;
   for (i=0; i<100; i++);
   if (RC6) RESULT = 1;
   if (RC5) RESULT = 4;
   if (RC4) RESULT = 7;
   if (RC3) RESULT = 10;
   PORTC = 2;
   for (i=0; i<100; i++);
   if (RC6) RESULT = 2;
   if (RC5) RESULT = 5;
   if (RC4) RESULT = 8;
   if (RC3) RESULT = 0;
   PORTC = 1;
   for (i=0; i<100; i++);
   if (RC6) RESULT = 3;
   if (RC5) RESULT = 6;
   if (RC4) RESULT = 9;
   if (RC3) RESULT = 11;
   PORTC = 0;
   return(RESULT);
}

char ReadKey(void)
{
   char X, Y;
   do {
      X = GetKey();
      }   while(X > 20);
   do {
      Y= GetKey();
      }   while(Y < 20);
   Wait_ms(100);  // debounce
   return(X);
   }
```

```
unsigned int Compute_Score(unsigned int CODE, unsigned int GUESS) {
   unsigned int SCORE, i, j, p;
   unsigned int C[4], G[4];

   SCORE = 0;

   for(i=0; i<4; i++) {
      C[i] = CODE % 10;
      CODE = CODE / 10;
      G[i] = GUESS % 10;
      GUESS = GUESS / 10;
   }

   for (i=0; i<4; i++) {
      p = 0;
      for(j=0; j<4; j++)
         if(C[i] == G[j])

            p = 1;
      if(C[i] == G[i]) p = 10;
      SCORE = SCORE + p;
      }
   return(SCORE);
   }


// Main Routine

void main(void)
{
   unsigned int i, j;
   int CODE, GUESS, SCORE;
   int X, TEMP, N;
   int c0, c1, c2, c3;

   TRISA = 0;
   TRISB = 0xFF;
   TRISC = 0xF8;
   TRISD = 0;
   TRISE = 0;
   TRISA = 0;
   ADCON1 = 15;

   PORTA = 0;

   LCD_Init();                  // initialize the LCD

   LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MSG0[i]);
   LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MSG1[i]);

   while(1) {


   // generate code
      while(!RB1);
      while(RB1) {
         X = (X + 1) % 1296;
      }

      c0 = X % 6;
      X = X / 6;
      c1 = X % 6;
      X = X / 6;
      c2 = X % 6;
      X = X / 6;
      c3 = X % 6;
```

```
   CODE = c3*1000 + c2*100 + c1*10 + c0;

   LCD_Move(0,7);
   LCD_Write('x');
   LCD_Write('x');
   LCD_Write('x');
   LCD_Write('x');

   SCORE = 0;
   N = 0;
   while(SCORE < 40) {
      TEMP = ReadKey();

      if (TEMP < 10) X = (X*10) + TEMP;

      if (TEMP == 10) {
         N = N + 1;
         GUESS = X;
         SCORE = Compute_Score(CODE, GUESS);
         }

      if (TEMP == 11) {
         X = X / 10;
         }

      LCD_Move(1,6);  LCD_Out(X, 4, 0);
      LCD_Move(1,13); LCD_Out(SCORE, 2, 0);
      LCD_Move(0,13); LCD_Out(N, 2, 0);

      if(SCORE == 40) {
         LCD_Move(0,6);
         LCD_Out(CODE, 4, 0);
         Wait_ms(2000);
         }
      }

   }

}
```