CS224
Section No.: 2
Spring 2018
Lab No.: 7
EFE ACER / 21602217

PRELIMINARY DESIGN REPORT

a)

- **Special Function Registers (SFRs)** are the registers inside the microprocessor that monitors and controls various operations of the microprocessor. One can think of a SFR as some kind of a control table.

- **TRISx (tri-state)** type of registers determine a PORT I/O pin to be an input or an output by specifying a corresponding bit. The bit corresponding to a specific pin is 1 when the pin is used as an input and 0 when it is used as an output. All pins are inputs by default (i.e. when reset).

- **PORTx** type of registers provide access to a PORT I/O pin. They basically allow to read data from the pins.

- **LATx** type of registers hold the data that is recently written to a PORT I/O pin.

- **ODCx** type of registers determine a PORT I/O pin to be a normal digital output or an open-drain output. Open-drain here, means that the pin is an open source, that is the current is continuously flowing outside of it. If the bit corresponding to a specific pin is 1, the pin acts as an open-drain output and supplies current to whatever connected to itself. Else if the bit is 0, the pin becomes a normal digital output.

- The x's at the end of each register type denotes one or many possible module instances.

- The I/O devices involved in the second part of the lab are four buttons and eight LEDs. LEDs are used to display the current value of the counter as a visible output. Buttons are used to provide inputs that control the speed and order of the display.

b-c)

```
/*
 * Lab7 - EFE ACER
 * Question 2
 * C code for a counter with adjustable speed and display order.
 */

char count = 0b11111111; //counter variable
```

```
void main() {
   int pace = 500; //pace is half a second as default
   AD1PCFG = 0xFFFF;   // configure AN pins as digital I/O
   JTAGEN_bit = 0;     // disable JTAG
   TRISD = 0;         // initialize PORTD as output
   TRISF = 1;
   LATD = 0b11111111; //LEDs are initially off

   while(1) {
      PORTF = 0b11111111; //reset the buttons as unpressed

      //if block to control the least significant bit using RF0 and RF1 buttons: Part 2.a
      if (PORTFbits.RF0 == 0 && PORTFbits.RF1 == 0) { //do not display if both are
pressed
         LATD = 0b11111111;
         count = 0b11111111;
      } if (PORTFbits.RF0 == 0) { //least significant bit is the rightmost one
         LATD = count;
         count--;
      } else if (PORTFbits.RF1 == 0) { //least significant bit is the leftmost one
         LATD.F7 = (count | 0b11111110) != 0b11111110; //uses bit manipulation to
         LATD.F6 = (count | 0b11111101) != 0b11111101; //reverse the order of the
         LATD.F5 = (count | 0b11111011) != 0b11111011; //LEDs
         LATD.F4 = (count | 0b11110111) != 0b11110111;
         LATD.F3 = (count | 0b11101111) != 0b11101111;
         LATD.F2 = (count | 0b11011111) != 0b11011111;
         LATD.F1 = (count | 0b10111111) != 0b10111111;
         LATD.F0 = (count | 0b01111111) != 0b01111111;
         count--;
      } else { //do not display if both are unpressed
         LATD = 0b11111111;
         count = 0b11111111;
      }

      //if block to control the speed of the display using RF2 and RF3 buttons: Part 2.b
      if (PORTFbits.RF2 == 0 && PORTFbits.RF3 == 0) { //back to default pace if both
are pressed
         pace = 500;
      } if (PORTFbits.RF2 == 0) { //set to slower pace
         pace = 1000;
      } else if (PORTFbits.RF3 == 0) { //set to faster pace
         pace = 200;
      } else { //back to default pace if both are unpressed
         pace = 500;
      }

      VDelay_ms(pace); //set the delay as the pace
   }
}
```