Lab 3 – Interfacing to 7-segment displays and buzzer

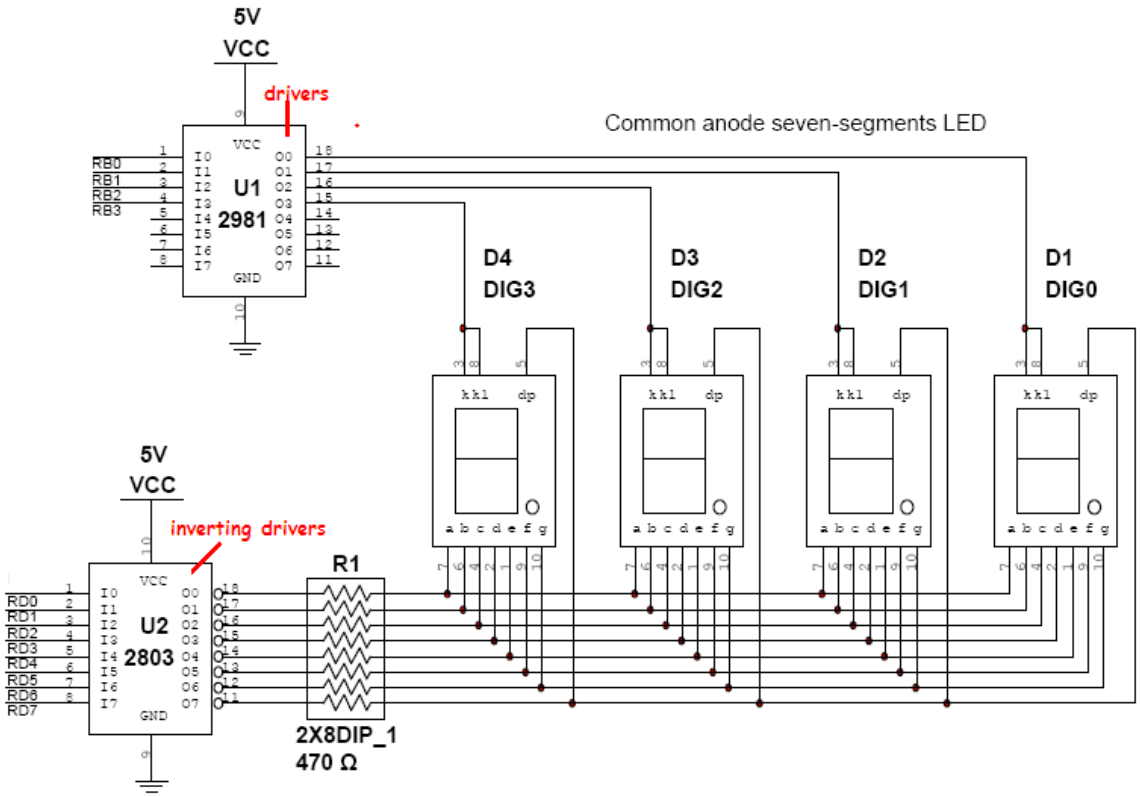
Objectives

- ☐ To learn to display a decimal number on a 7-segment display.
- ☐ To learn to use multiplexing technique to display several digits on several 7-segment displays.
- ☐ To learn to implement a "queue number system".
- ☐ To learn to produce a tone on a buzzer.

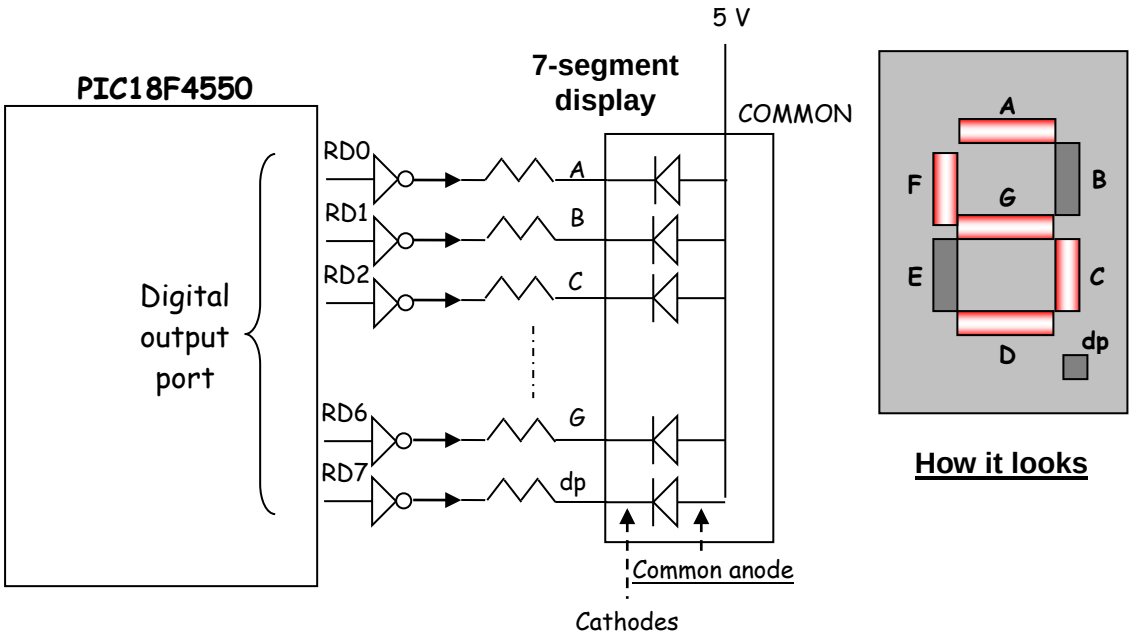
Introduction / Briefing

7-segment display at Ports B & D

- ☐ In this experiment, you will be turning on and off segments in four 7-segment displays connected to Ports B & D, to display some numbers.



☐ Considering only one 7-segment display (shown below) and answer the following questions:



Q1: Are the LED's in the 7-segment display connected in the "common anode" mode or the "common cathode" mode? _____

Q2: What must RD0 produce (logic '0' or logic '1') to turn on segment A?
_____

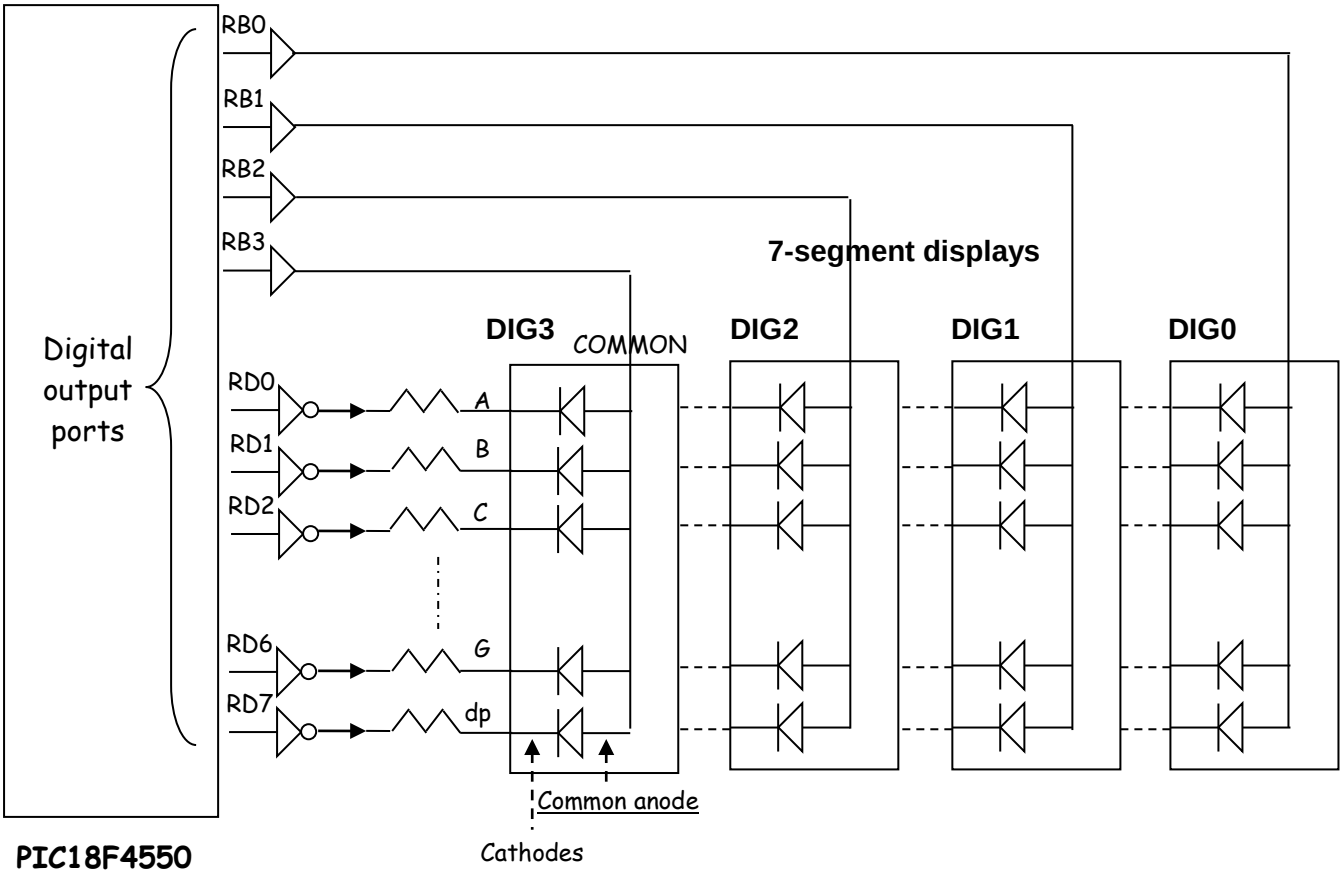
Q3: What must PORT D produce (in binary format) to show the digit "1" on the 7-segment display? PORTD = 0b_____.

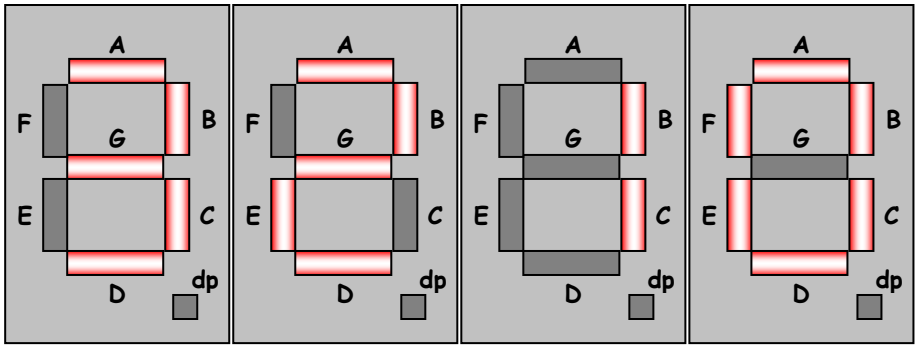
Q4: Of course, PORT D must be configured as an output port. Give the 2-line C command to configure PORT D as a digital output port and to show the digit "5" on the 7-segment display:

TRISD = 0b_____ // configure Port D as digital outp.

PORTD = 0b_____ // display "5"

☐ Considering four 7-segment displays together (shown below) and answer the following questions:





Q5: What will be shown on the 7-segment displays if PORT D outputs 0b01001111 while PORT B outputs 1000 to its lower 4 bits?

DIG3 shows _____
DIG2 shows _____
DIG1 shows _____
DIG0 shows _____

Q6: What must PORT B and PORT D produce to show 2 on DIG2?

PORT D = 0b _____

PORTBbits.RB3 = _____
PORTBbits.RB2 = _____
PORTBbits.RB1 = _____
PORTBbits.RB0 = _____

Q7: What will be shown on the 7-segment displays if following C program is run?

```
TRISB = 0b11110000;        // lower 4 bits are outputs
TRISD = 0b00000000;        // all bits are outputs

while (1)
{
        PORTB = 0b00000001;        // enable DIG0
        PORTD = 0b00111111;        // display 0
        // Some delay

        PORTB = 0b00000010;        // enable DIG1
        PORTD = 0b00000110;        // display 1
        // Some delay

        PORTB = 0b00000100;        // enable DIG2
        PORTD = 0b01011011;        // display 2
        // Some delay

        PORTB = 0b00001000;        // enable DIG3
        PORTD = 0b01001111;        // display 3
        // Some delay
}
```

Your answer:_____

_____

Q8: What do you think will happen if the delay is increased?

Your answer: _____

Q9: What do you think will happen if the delay is decreased?

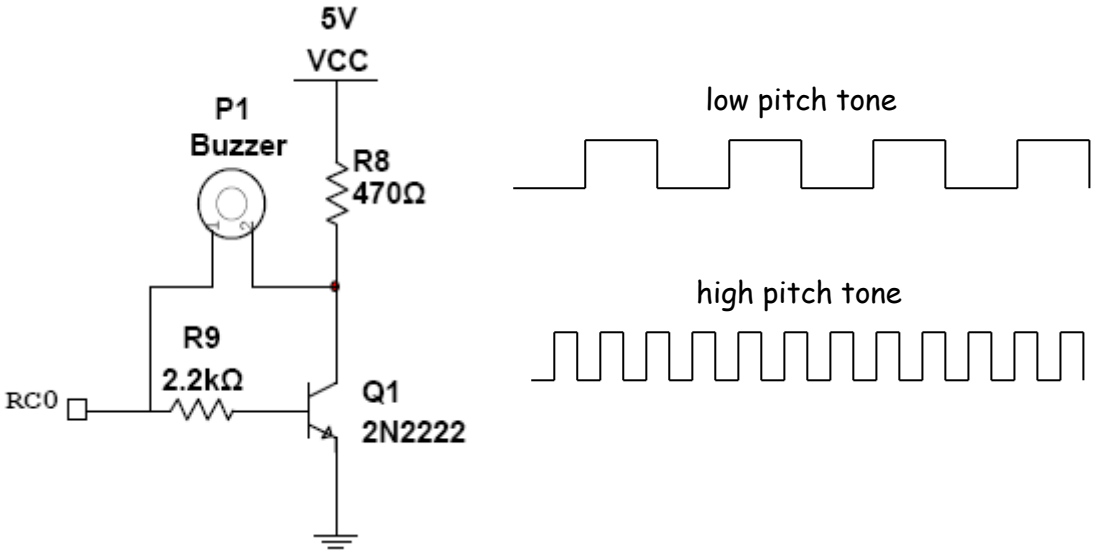
Your answer:

_____

You will find out the answer to the two previous questions in the experiment.

Buzzer at Port C

☐ In this experiment, you will also be turning on and off a buzzer connected to PORT C to produce a "tone".



☐ Study the above diagram and answer the following questions:

Q10: What happens when RC0 outputs logic '1'?

The transistor is turned on and (assuming $V_{CE[sat]}$ = 0.2V,) pin 2 of the Buzzer is at _____ V while pin 1 of the Buzzer is at _____ V. So the Buzzer will be turned _____. If RC0 outputs logic '0', the Buzzer will be turned _____.

☐ By toggling (on -> off -> on -> off ….) RC0 continuously, a tone can be produced by the buzzer.

☐ If the rate of toggling is high, a high pitch tone is produced.

Activites:

Before you begin, ensure that the Micro-controller Board is connected to the General I/O Board. The General I/O Board is further connected to a 7-Segment/Switch Board.

PORTD – 8 segments ('a' to 'g', and decimal point) of all 4 digits, active high ('1' turns on a segment, '0' turns off a segment).

PORTB – RB0 to RB3 – COM pins of all 4 digits (DIG0 to DIG3), active high ('1' enables digit, '0' disables digit).

PORTB – RB5 – push button switch, active low (pressed gives '0', released gives '1').

So PORTD controls the number e.g. '8' to be displayed on a digit, while PORTB controls which digit displays the number.

Displaying a decimal number on a 7-segment display

1. Launch the *MPLABX IDE* and create a new project called *Lab3*.

Display "0" on DIG0. ⇨

2. Add the file *Single7Seg.c* to the *Lab3* project *Source File* folder. Make sure *Copy* is ticked before you click *Select*. If you have forgotten the steps, you will need to refer to the previous lab sheet.

3. Study the code and describe what this program will do:

   _____

4. Build, download and execute the program. Observe the result and see if it is as expected.

Display "1" on DIG1. ⇨

5. Modify the code to display the digit "*1*" on the next 7-segment i.e. DIG1. Build, download and execute the program to verify your coding.

6. Describe what will happen when *PORTB = 0b00001111*. Why?

   Answer: _____

Display 4 decimal numbers on four 7-seg.'s

Displaying 4 different decimal numbers on four 7-segment displays

7. Replace *Single7Seg.c* with *Four7Seg.c*. Note that the program uses the delay function *delay_ms()* and contains *#include "delays.h"*. The files *delays.h* and *delays_utilities.c* need to be added to the Project.

8. Study the code and describe what this program will do:

_____

9. Build, download and execute the program. Observe the result and see if it is as expected.

Increase delay

10. Increase the delay between digits. What do you observe?

_____

Decrease delay

11. Decrease the delay between digits. What do you observe?

_____

12. As can be seen, multiplexing technique here involves turning on only one digit of display at a time, and after a short delay, move on to the next digit etc:

    Show '0' on digit DIG0.
    Delay
    Show '1' on digit DIG1.
    Delay
    Show '2' on digit DIG2.
    Delay
    Show '3' on digit DIG3.
    Delay
    Repeat above

    The delay is to give time for the LED's to light up and the number to be seen. Too long a delay will cause the numbers to flicker and too short and the display will become blur, as the LED's do not have time to turn on properly and be seen.

13. You may try to display today's date as DDMM and show it to your classmates.

Extra Exercise - Implementing a "queue number system"

(Do this only if you still have time. Otherwise, skip to the next section to try out the "buzzer".)

Q-no. system

14. Replace *Four7Seg.c* with *Count7SegSw.c*.

15. Study the code and describe what this program will do:

_____

16. Read the following explanation if you are stuck.

17. The decimal numbers to display on the four 7-segments are stored in an array of 4 unsigned chars

    unsigned char val[4]; // i.e. val [0], val [1], val [2], val [3]

    These are initialised to *val [3] = 9; val [2] = 8; val [1] = 7; val [0] = 6;* in the *main* program. So, the initial display should be "9 8 7 6".

18. 9876 are what you want to see. However, what the 7-segments want to be told are the binary patterns *0b01101111 [9], 0b01111111 [8], 0b00000111 [7], 0b01111101 [6].*

19. The function *convert* (in the *seg7_utilities*) produces the binary pattern required to show a decimal number on a 7-segment. E.g. if decimal number ("*digit*") = "0", binary pattern ("*leddata*") = 0b00111111.

20. As you know by now, multiplexing technique is used to enable each of the 4 digits in turn, so that the number of PIC pins required to display 4 digits (including the decimal points) is fewer than 4 x 8.

21. Here an unsigned char variable *point* is used to control which digit is lighting up.

22. It is initialised to *0b00000001* i.e. *DIG0* will light up first.

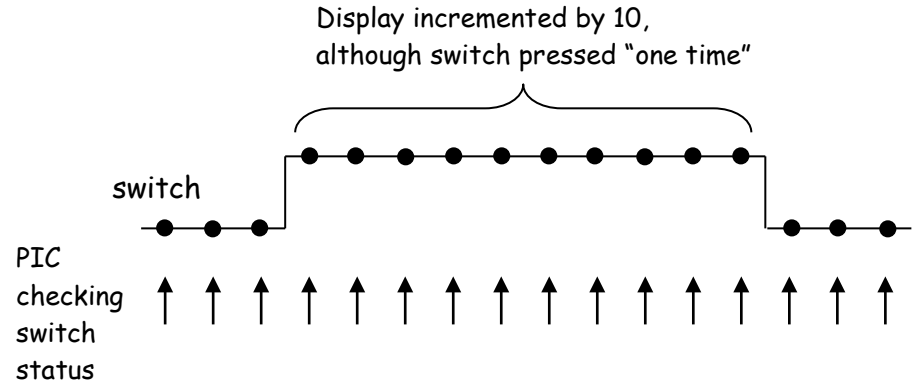
23. Putting all these ideas together, you get the following chunk of codes:

```
point = 0b00000001;  // enable DIG0 first
for (i = 0; i < 4; i++)  // loop from DIG0 to DIG3
{
 PORTB = point;    // enable one DIG
 outchar = val [i];  // get one decimal number to display from the array
                     // convert to corresponding binary pattern for the 7-seg,
 PORTD = convert (outchar);  // and send the binary pattern to the enabled DIG

 point = point << 1; // shift left by 1 bit, to enable the next DIG,
        // so 0b00000001 becomes 0b00000010, then 0b00000100,
        // then 0b00001000

 … // some delay
}
```

24. Whenever the switch connected to *RB5* is pressed, the 4-digit display is incremented by 1. This is done by the following lines of code and the function *update*:

```
if (PORTBbits.RB5 == 0) // if switch is pressed
{
 press = 1; // this is explained below
 val [0] = val [0] + 1;  // increment the lowest digit by 1
 update ();  // update the other digits accordingly
}
```

25. A micro-controller can work very fast. When a switch is pressed "one time", a micro-controller could have read it several times, and increment the display several times, as shown below:



26. To solve this problem, a variable *press* is used to control the flow of the program:

```
…
press = 0; // initially… in the "switch not pressed" state
while (1)
{
 … // display the decimal numbers

 if (press == 0) // starting from the "switch not pressed" state
  {
   if (PORTBbits.RB5 == 0) // if switch pressed
   {
    press = 1; // change to the "switch pressed" state
    … // increment the 4-digit number to display
   }
  }

 if (PORTBbits.RB5 == 1) // switch released
  press = 0; // change to the "switch not pressed" state, ready for next round
}
```

Display is only incremented the first time the switched is found to be pressed.

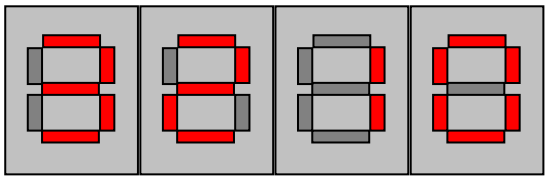
Subsequently, only check for the release of the switch.

27. The best way to check whether you have understood this program is to try to explain it to a classmate.

28. Once you have understood it, build, download and execute the program. Observe the result and see if it is as expected.

29. Describe how you can use this in a Q-number system. What else do you need?

I still need _____

a.) display 

b.) UP button

Your Q-number is
3208

c.) ticket printer + "counter" + user button

Producing a tone on a buzzer

Tone on buzzer

30. Replace *Count7SegSw.c* with *Buzzone.c*.

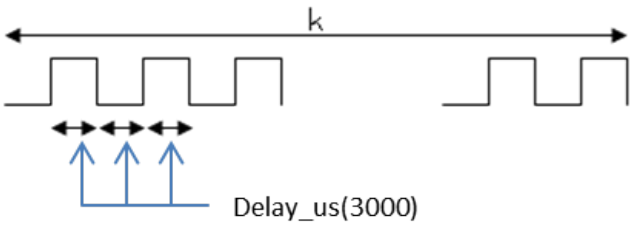
31. Study the code and describe what this program will do:

_____

32. Note that in *Buzzone.c* (under the function *onetone*), the variable is used: *k*.

    The "for" loop i.e. *k* determines the duration of the buzzing, while the delay_us() determines the pitch of the buzzing.

```
void onetone(void) //Function to generate one tone
{
    unsigned int k;

    for (k = 0; k < 100; k++) //Determines duration of tone
    {
        delay_us(3000);  // useable values from 100  to 5000
        PORTCbits.RC0 = !PORTCbits.RC0; //Invert logic level at RC0
    }
}
```



33. Build, download and execute the program. Observe the result and see if it is as expected.

Different tone

34. Modify the program by adding another function named *twotone* with a different *value* in *delay_us(value)*. Include *twotone* in the *main* program and test out the sound effect, as follows:

```
while (1)
{
        Onetone ();
        PORTD = 0b10101010; // pattern on LEDs
        delay_ms(500);
        Twotone ();
        PORTD = 0b01010101; // another pattern on LEDs
        delay_ms(500);

        while (1); // loop forever to stop music!
}
```

35. Debug until the program can work.

```
// Single7Seg.c
// Program to test 1 7-segment display

#include <xc.h>

void main(void)
{
      ADCON1 = 0x0F;

      TRISB=0b11110000; //RB3 to RB0 are connected DIG3 to DIG0
                        //RB5 is connected to a switch

      TRISD=0b00000000; //RD7 to RD0 are connected to segment LEDs


      while(1)                //repeat
      {

            PORTB = 0b00000001;     //enable DIG0

            PORTD = 0b00111111;     //display 0

      }
}
```

```
/*
 * File:   Four7seg.c
 *
 * Created on 13 January, 2016, 1:52 PM
 */


#include <xc.h>
#include "delays.h"
void main(void)
{

      TRISB=0b11110000; //RB3 to RB0 are connected DIG3 to DIG0
                        //RB5 is connected to a switch

      TRISD=0b00000000; //RD7 to RD0 are connected to segment LEDs


      while(1)               //repeat
      {

            PORTB = 0b00000001;     //enable DIG0
            PORTD = 0b00111111;     //display 0
            delay_ms(1000);         //LEDs on for a while

            PORTB = 0b00000010;     //enable DIG1
            PORTD = 0b00000110;     //display 1
            delay_ms(1000);         //LEDs on for a while

            PORTB = 0b00000100;     //enable DIG2
            PORTD = 0b01011011;     //display 2
            delay_ms(1000);         //LEDs on for a while

            PORTB = 0b00001000;     //enable DIG3
            PORTD = 0b01001111;     //display 3
            delay_ms(1000);         //LEDs on for a while

      }
}
```

```
// Count7SegSw.c
// Counting on 4 7-segment display by a switch on 7-seg Board

#include <xc.h>
#include "delays.h"
#include "seg7.h"

unsigned char point, outchar, press;

void main(void) {
    char i;
    TRISB = 0b11110000; //RB3 to RB0 are connected DIG3 to DIG0
                        //RB5 is connected to a switch

    TRISD = 0b00000000; //RD7 to RD0 are connected to segment LEDs

    val[3] = 9; //contents of DIG3
    val[2] = 8; //contents of DIG2
    val[1] = 7; //contents of DIG1
    val[0] = 6; //contents of DIG0
    press = 0;

    while (1) //repeat
    {

        point = 0b00000001; //enable DIG0

        for (i = 0; i < 4; i++)
        {
            PORTB = point; //enable one DIG
            outchar = val[i]; //get one value for the DIG
            PORTD = convert(outchar); //convert to LED code

            point = point << 1; //point to the next DIG
            delay_ms(1);

        }

        if (press == 0) //switch press first time
        {
            if (PORTBbits.RB5 == 0) //if RB5sw is ON
            {
                press = 1; //switch being pressed
                val[0] = val[0] + 1; //increase DIG0 value
                update(); //adjust the rest of values

            }
        }

        if (PORTBbits.RB5 == 1) press = 0; //switch released
    }
}
```

```
// file : seg7.h

unsigned char val[4]; // variable used

extern void update(void)      ; // update the above variable

extern char convert(char outchar); // converts the outchar to 7
segment                                    //display pattern


/*
 * File:   seg7_utilities.c
 *
 * Created on 14 January, 2016, 7:59 PM
 */

#include <xc.h>

extern unsigned char val[4];

char convert(char digit)
{
    char leddata;

      if(digit==0)leddata=0b00111111;
      if(digit==1)leddata=0b00000110;
      if(digit==2)leddata=0b01011011;
      if(digit==3)leddata=0b01001111;
      if(digit==4)leddata=0b01100110;
      if(digit==5)leddata=0b01101101;
      if(digit==6)leddata=0b01111101;
      if(digit==7)leddata=0b00000111;
      if(digit==8)leddata=0b01111111;
      if(digit==9)leddata=0b01101111;
    return(leddata);
}


void update(void)             //Function to adjust DIG values
{
      if(val[0]>=10)
      {
            val[1]=val[1]+1;
            val[0]=0;
      }

      if(val[1]>=10)
      {
            val[2]=val[2]+1;
            val[1]=0;
      }
      if(val[2]>=10)
      {
            val[3]=val[3]+1;
            val[2]=0;
      }
      if(val[3]>=10)
      {
            val[3]=0;
      }
}
```

```
// BuzzOne.c
// Program to activate buzzer with one tone
// For project using USB interface with Bootloader

#include <xc.h>
#include "delays.h"

void onetone(void) //Function to generate one tone
{
    unsigned int k;

    for (k = 0; k < 100; k++) //Determines duration of tone
    {
        delay_us(3000);  // useable values from 100  to 5000
        PORTCbits.RC0 = !PORTCbits.RC0; //Invert logic level at RC0
    }
}

void main(void) {

    TRISCbits.TRISC0 = 0; //-- Set RC0 as output

    TRISD = 0x00; //-- Set all pins on PortD as output

    {
        onetone(); //sound ON then OFF
        PORTD = 0b10101010; //pattern on LEDs
        delay_ms(500);

        onetone(); //sound ON then OFF
        PORTD = 0b01010101; //another pattern on LEDs
        delay_ms(500);

        while (1); // loop forever to stop music!
    }
}
```