

### Lab 3 - Interfacing to 7-segment displays and buzzer

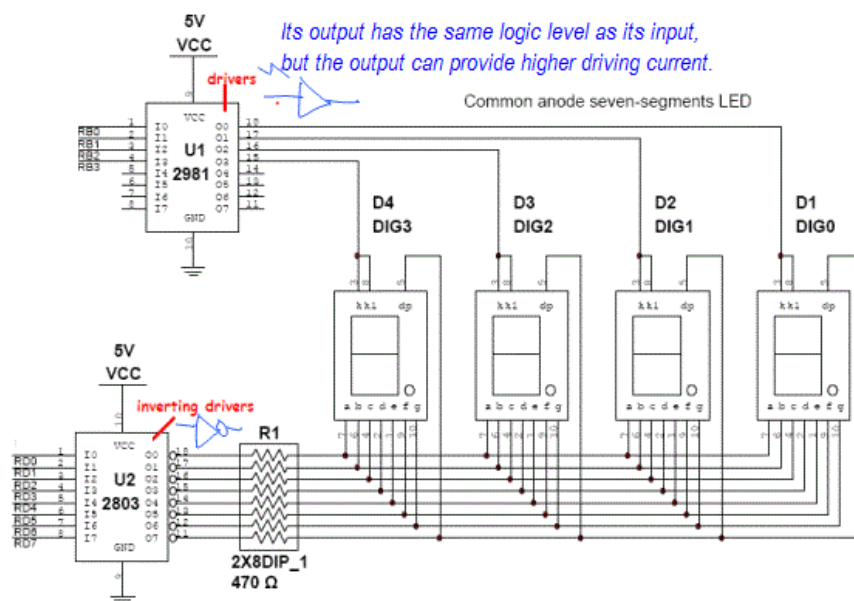
#### Objectives

- ☐ 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.

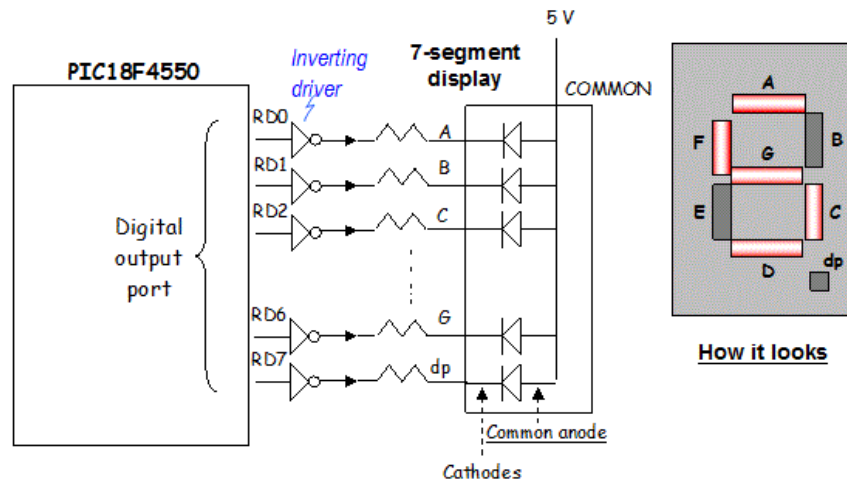
#### Introduction / Briefing

##### 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 "7" on the 7-segment display? PORTD = 0b00001111; CBA Turn on segments A, B, C

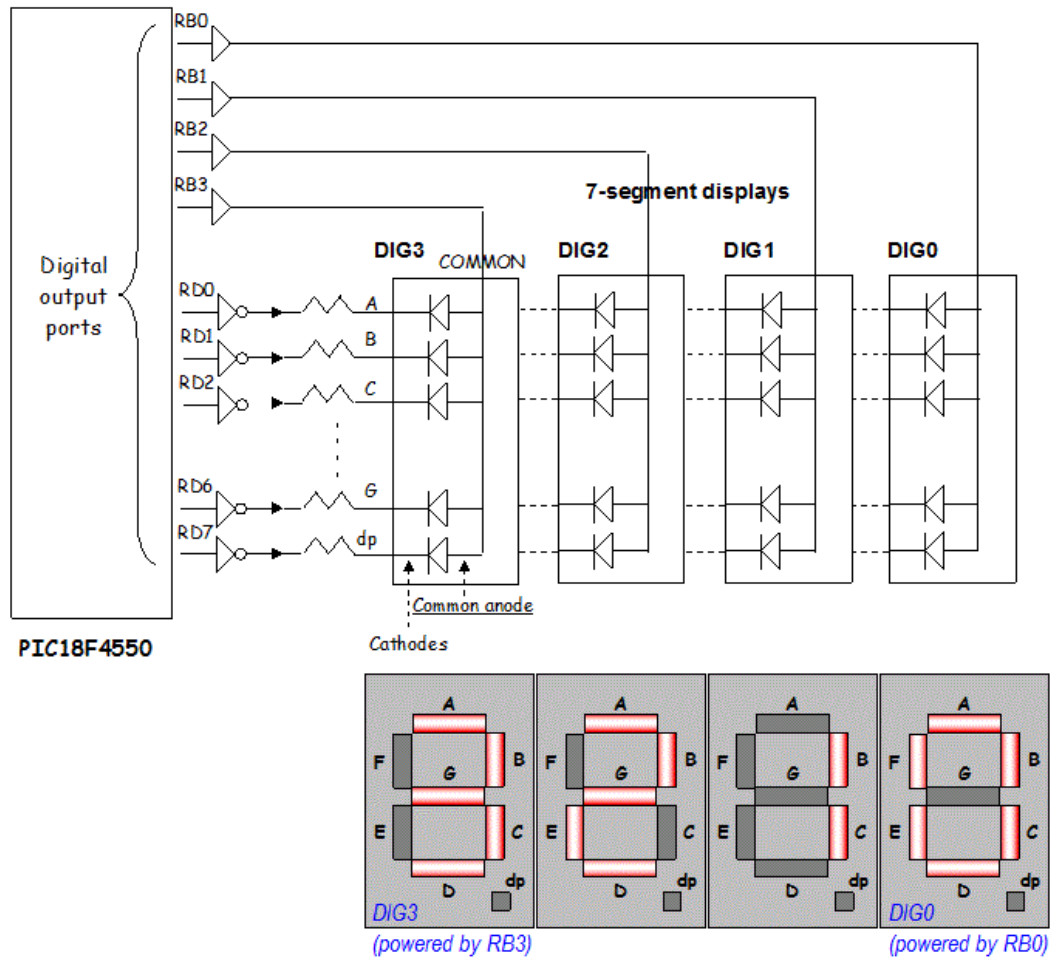
Q4: Of course, PORT D must be configured as an output port. Give the 2-line C command to configure PORT D as a digital output port and to show the digit "5" on the 7-segment display:

Turn on segments A, C, D, F, G

TRISD = 0b00000000; // configure Port D as digital outp.

PORTD = 0b\_\_\_\_\_ // display "5"  
dp,G,F,E,D,C,B,A

- 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: \_\_\_\_\_

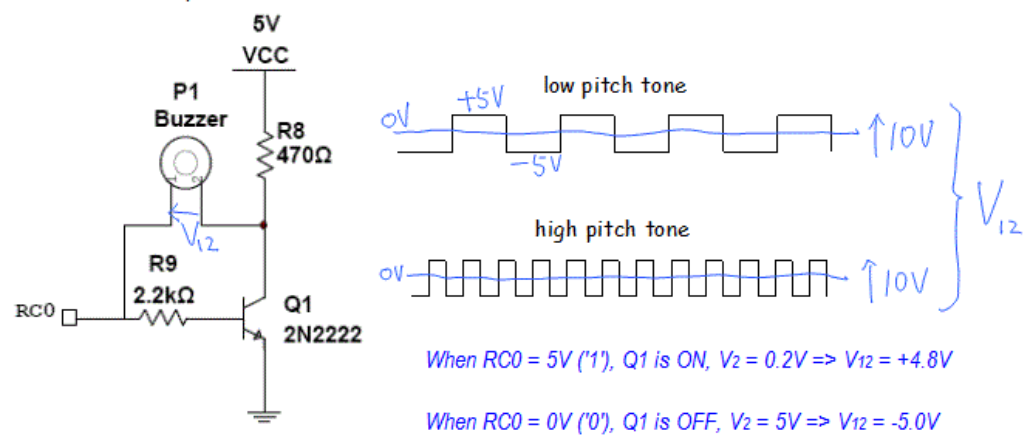
Q9: What do you think will happen if the delay is decreased?

Your answer: \_\_\_\_\_

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

### Buzzer at Ports B & D

- ☐ 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  $\rightarrow$  off  $\rightarrow$  on  $\rightarrow$  off .... ) RC0 continuously, a tone can be produced by the buzzer.
- ☐ If the rate of toggling is high, a high pitch tone is produced.

**Activities:** (P.6)

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.

You can refer to a copy of the schematic to see the following connections:

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**

Display  
"0" on  
DIG0.

1. Launch *MPLAB IDE*. Open Lab1 workspace by clicking *Project -> Open...* and selecting *Projeta.mcp* from the *D:\PICProject* folder.
2. Replace *CountLeds.c* with *Single7Seg.c*. 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:

\_\_\_\_\_

Display  
"1" on  
DIG1.

4. Build, download and execute the program. Observe the result and see if it is as expected.
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*.
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* produces the binary pattern required to show a decimal number on a 7-segment. E.g. if decimal number ("*inchar*") = "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 (index = 0; index < 4; index++) // loop from DIG0 to DIG3
{
    PORTB = point; // enable one DIG
    outchar = val [index] // get one decimal number to display from the array
    convert (outchar) // convert to the corresponding binary pattern for the 7-seg.
    PORTD = leddata; // send the binary pattern to the enabled DIG

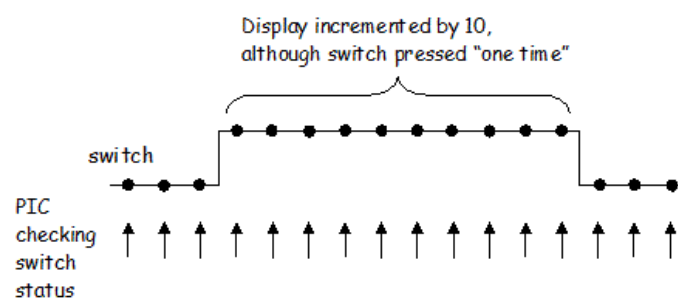
    point = point << 1; // shift left by 1 bit, to enable the next DIG,
                        // so 0b00000001 becomes 0b00000010, then 0b00000100,
                        // then 0b00001000

    .. // 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
}

```

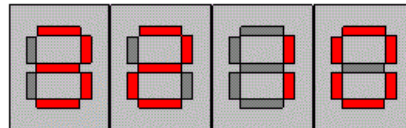
Display is only incremented the first time the switched is found to be pressed.

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



Your Q-number is  
3208

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



b.) UP button

**Producing a tone on a buzzer**

Tone on buzzer

30. Replace *Four7Seg.c* or *Count7SegSw.c* with *Buzzzone.c*.

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

32. Note that in *Buzzzone.c* (under the function *onetone*), two variable are used: *i* & *k*.

The inner "for" loop (using *i*) is to introduce a delay before toggling *PORTCbits.RC0* connected to a buzzer. When *RC0* is toggle on and off, on and off, a buzz tone is produced.

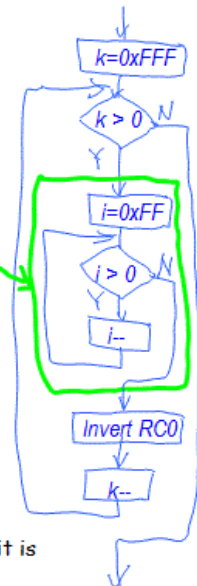
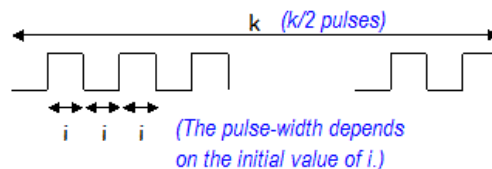
The outer "for" loop i.e. *k* determines the duration of the buzzing, while the inner "for" loop i.e. *i* determines the pitch of the buzzing.

```
void onetone(void) //Function to generate one tone
```

```
{
    unsigned int i,k;

    for(k=0x0FFF; k>0; k--) //Determines duration of tone
    {
        for(i=0xFF; i>0; i--) //Determines pitch of tone
            PORTCbits.RC0 = !PORTCbits.RC0; //Invert logic level at RC0
    }
}
```

Simple delay



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 *i* loop e.g. *for (i=0x2F; i>0; i--)*. Include *twotone* in the *main* program and test out the sound effect, as follows:

---

```
while (1)
{
    Onetone ();
    PORTD = 0b10101010; // pattern on LEDs
    Delay1KTCYx (250);

    Twotone ();
    PORTD = 0b01010101; // another pattern on LEDs
    Delay1KTCYx (250);
}
```

35. Debug until the program can work. When your program is working, show it to your lecturer.

Lecturer's signature \_\_\_\_\_

**// Single7Seg.c**

// Program to test one 7-segment display

#include &lt;delays.h&gt;

// other lines not shown...

void main(void)

```

{
    ADCON1 = 0x0F;
    TRISB=0b11110000;    // RB3-0 are connected DIG3-0, active high
                          // RB5 is connected to a switch
    TRISD=0b00000000;    // RD7-0 are connected to segment LEDs, active // high

    while(1) // repeat
    {
        PORTB = 0b00000001; // enable DIG 0
        PORTD = 0b00111111; // display 0
        Delay1KTCYx(10);
    }
}

```

**// Four7Seg.c**

// Program to multiplex four 7-segment display

#include &lt;delays.h&gt;

// other lines not shown...

void main(void)

```

{
    ADCON1 = 0x0F;
    TRISB=0b11110000;    // RB3-0 are connected DIG3-0, active high
                          // RB5 is connected to a switch
    TRISD=0b00000000;    // RD7-0 are connected to segment LEDs, active // high

    while(1) // repeat
    {
        PORTB = 0b00000001; // enable DIG 0
        PORTD = 0b00111111; // display 0

        Delay10KTCYx(250); // LEDs on for a while
        Delay10KTCYx(250);
        Delay10KTCYx(250);
        Delay10KTCYx(250);

        PORTB = 0b00000010; // enable DIG 1
        PORTD = 0b00000110; // display 1

        Delay10KTCYx(250); // LEDs on for a while
        Delay10KTCYx(250);
        Delay10KTCYx(250);
        Delay10KTCYx(250);

        PORTB = 0b00000100; // enable DIG 2
        PORTD = 0b01011011; // display 2

        Delay10KTCYx(250); // LEDs on for a while
        Delay10KTCYx(250);
    }
}

```

```

        Delay 10KTCYx(250);
        Delay 10KTCYx(250);

        PORTB = 0b00000001; // enable DIG 3
        PORTD = 0b01001111; // display 3

        Delay 10KTCYx(250); // LEDs on for a while
        Delay 10KTCYx(250);
        Delay 10KTCYx(250);
        Delay 10KTCYx(250);
    }
}

```

### // Count7SegSw.c

// Counting on 4 7-segment display by a switch on 7-seg Board

```
#include <delays.h>
```

```
// other lines not shown...
```

```
unsigned char val[4];
unsigned char index, point, outchar, leddata, inchar, press;
```

```
void convert(unsigned char inchar) // Function to change value to LED code
{
    if(inchar==0)leddata=0b00111111;
    if(inchar==1)leddata=0b00000110;
    if(inchar==2)leddata=0b01011011;
    if(inchar==3)leddata=0b01001111;
    if(inchar==4)leddata=0b01100110;
    if(inchar==5)leddata=0b01101101;
    if(inchar==6)leddata=0b01111101;
    if(inchar==7)leddata=0b00000111;
    if(inchar==8)leddata=0b01111111;
    if(inchar==9)leddata=0b01101111;
}

```

```
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;
    }
}

void main(void)
{
    ADCON1 = 0x0F;
    TRISB=0b11110000;    // RB3-0 are connected DIG3-0, active high
                        // RB5 is connected to a switch
    TRISD=0b00000000;    // RD7-0 are connected to segment LEDs, active high

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

            point = point<<1;    // point to the next DIG
            Delay1KTCYx(10);
        }

        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
    }
}

```

### // BuzzOne.c

// Program to activate buzzer with one tone  
 // For project using USB interface with Bootloader

```

#include <delays.h>

// other lines not shown...

void onetone(void);    // -- Function prototype

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

```

---

```

        for(k=0x0FFF; k>0; k--)           // Determines duration of tone
        {
            for(i=0xFF; i>0; i--);        // Determines pitch of tone
            PORTCbits.RC0 = !PORTCbits.RC0; // Invert logic level at RC0
        }
    }

void main(void)
{
    ADCON1 = 0x0F;
    TRISCbits.TRISC0=0; // - Set RC0 as output
    TRISD = 0x00;       // - Set all pins on PortD as output

    while(1)
    {
        onetone();           // sound ON then OFF
        PORTD=0b10101010;    // pattern on LEDs
        Delay1KTCYx(250);
        Delay1KTCYx(250);
        Delay1KTCYx(250);
        Delay1KTCYx(250);

        onetone();           // sound ON then OFF
        PORTD=0b01010101;    // another pattern on LEDs
        Delay1KTCYx(250);
        Delay1KTCYx(250);
        Delay1KTCYx(250);
        Delay1KTCYx(250);
    }
}

```