Lec 2: I/O Ports

Clock generator (Oscillator)

- Clock is needed so that microcontroller could execute a program or program instructions.
- There are three methods of generating clock signal:

  1) **An internal oscillator**
  2) **An external RC oscillator**
  3) **An external Crystal oscillator**

Clock generator (Oscillator)

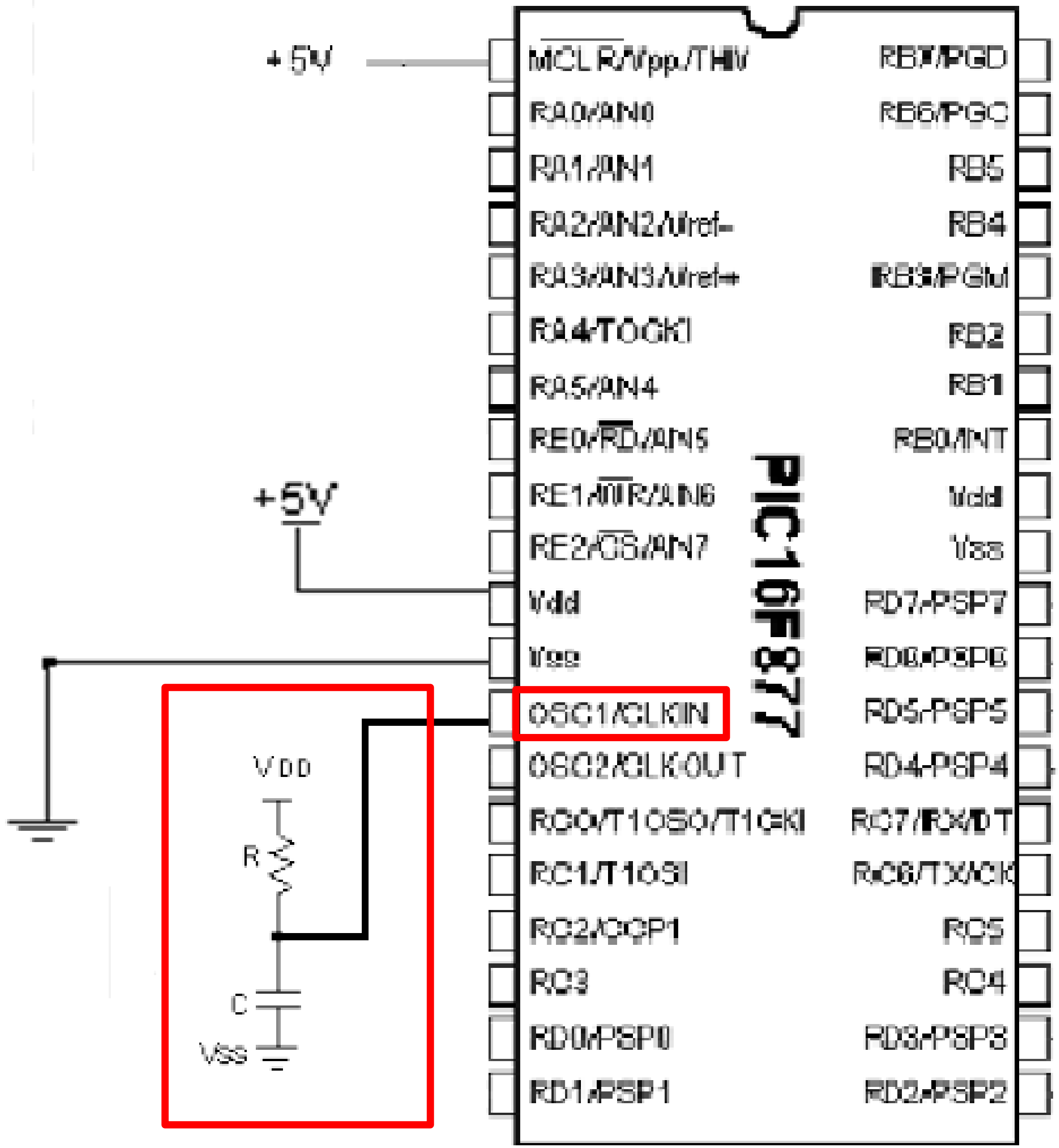
1) An Internal Oscillator

- The **default** clock option is normally the internal oscillator, if available.
- It reduces the number of external components required, and provides a default clock rate of 4 MHZ in standard chips and a maximum clock rate of 32 MHz in more recent 16F1xxx series chips.

Clock generator (Oscillator)

2) An External RC Oscillator

- For applications where the precise timing of the program is not important, and an internal oscillator is not available, an inexpensive RC oscillator can be used.
- It requires only a resistor and capacitor connected to the **CLKIN/OSC1** pin of the chip.
- It is recommended that value of resistor R should be between 3 k and 100 k and capacitor above 20pF,
- Clock Frequency = 1 / (R*C)


+5V
MCLR/Vpp/THV
RA0/AN0
RA1/AN1
RA2/AN2/Vref-
RA3/AN3/Vref+
RA4/TOCKI
RA5/AN4
RE0/RD/AN5
RE1/WR/AN6
RE2/CS/AN7
Vdd
Vss
OSC1/CLKIN
OSC2/CLKOUT
RC0/T1OSO/T1CKI
RC1/T1OSI
RC2/CCP1
RC3
RD0/PSP0
RD1/PSP1
RB7/PGD
RB6/PGC
RB5
RB4
RB3/PGM
RB2
RB1
RB0/INT
Vdd
Vss
RD7/PSP7
RD6/PSP6
RD5/PSP5
RD4/PSP4
RC7/RX/DT
RC6/TX/CK
RC5
RC4
RD3/PSP3
RD2/PSP2
PIC16F877
+5V
VDD
R
C
VSS

Clock generator (Oscillator)

3) An External Crystal Oscillator

- The crystal oscillator is the **most precise** one and it is connected across the **OSC1** and **OSC2** pins, with a capacitor (15-68 pF) to ground from each pin.

- There are three types of external crystal can be used:

  1. Low power (**LP**): from 32 kHz to 200 kHz
  2. Crystal (**XT**): from 200 kHz to 4 MHz
  3. High speed Crystal (**HS**): from 4 MHz to 20 MHz

Ports Direction Configuration

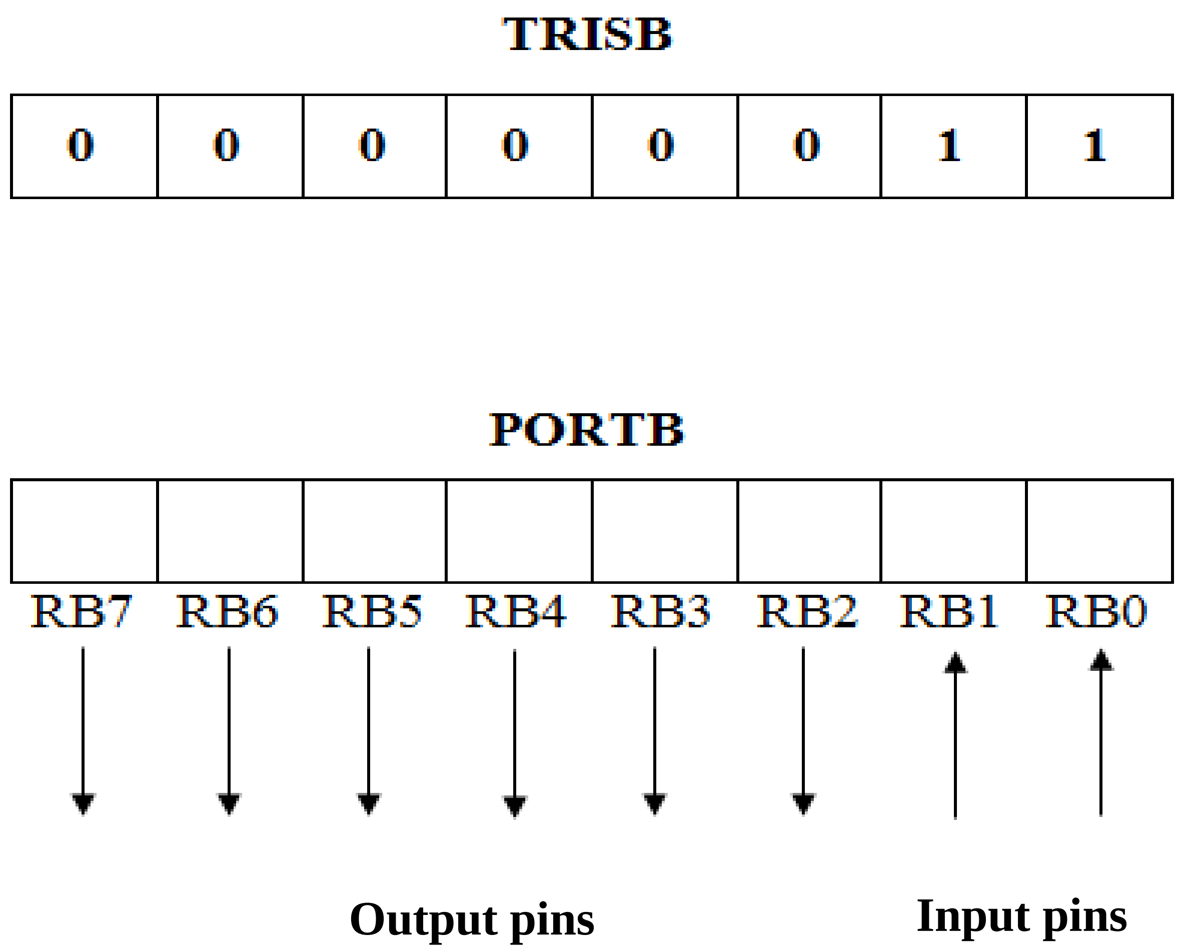
- Ports in a PIC microcontroller are **bi-directional**. Thus, each pin of a port can be used as **an input or an output pin**.
- **Port direction control register** configures the port pins as either inputs or outputs.
- This **register** is called the **TRIS** register and every port has a TRIS register named after its port name. For example, TRISA is the **direction control register for PORTA**. Similarly, TRISB is the **direction control register for PORTB** and so on.

Ports Direction Configuration

TRIS REGISTER		**PORTS**
TRISA	→	PORTA
TRISB	→	PORTB
TRISC	→	PORTC
TRISD	→	PORTD
TRISE	→	PORTE

Ports Direction Configuration

- A bit in the TRIS register containing **1** makes the corresponding port register pin an **input**.
- A bit in the TRIS register containing **0** makes the corresponding port register pin an **output**.
- **For example**, to make pins RB0 and RB1 of PORTB **input** and the other bits **output**, we have to load the **TRISB register** with the bit pattern **0 0 0 0 0 0 1 1**


TRISB
0
0
0
0
0
0
1
1
PORTB
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
Output pins
Input pins

TRIS Register in C++ programming

- For the pervious example we want to load the TRISB register with the bit pattern **0 0 0 0 0 0 1 1**
- We can use an assignment statement as in the following :

TRISB = 0x03; // the assigned value in **hex.** format

TRISB = 3; // the assigned value in **decimal** format

TRISB = 0b00000011; // the assigned value in **binary** format

Port Data Register for Reading Or Writing

- After ports configuration process was done using TRIS register, we can use this configured port to **read data** from its pins or **out data** through its pins.
- The read or write process is done by using the **port data register**:
  - **PORTA**
  - **PORTB**
  - **PORTC**
  - **PORTD**
  - **PORTE**

Port Data Register in C++ programming

- We can use an assignment statement to read from input port or write to output port as in the following :

1) Write Data To PORTB

(assume that PORTB is configured as output port)

```
PORTB= 0x33;          // the assigned value in hex. format
PORTB = 51;           // the assigned value in decimal format
PORTB = 0b00110011;   // the assigned value in binary format
```

Port Data Register in C++ programming

From theses examples we note that:

- A number with a prefix '**0b**' indicates a binary number.
- A number with a prefix '**0x**' indicates a hexadecimal number.
- A number without prefix is a decimal number.

Port Data Register in C++ programming

2) Read Data From PORTB

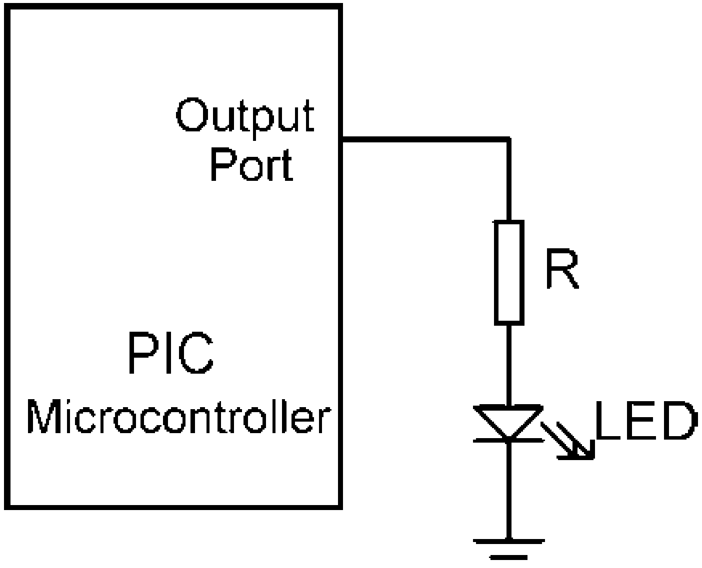
(assume that PORTB is configured as **input** port and PORTC is configured as **output** port)

```
D = PORTB            // D is a predeclared variable

PORTC = PORTB        // read from PORTB and out to PORTC

PORTC.B2 = PORTB.B1 & PORTB.B2

// perform ANDING operation between the reading data from pins  RB1 and RB2
//and out the result to RC2 pin. These statement can be used in another form:

PORTC.F2 = PORTB.F1 & PORTB.F2
```

I/O Interfaces

LED Interface:

- LEDs consume about 10 mA of current for normal brightness.
- If the output voltage of the port is 5 V and the voltage drop across the LED is 2 V, we need to drop 3 V across the resistor.
- If we assume that the current through the LED is 10 mA, we can calculate the value of the required resistor as



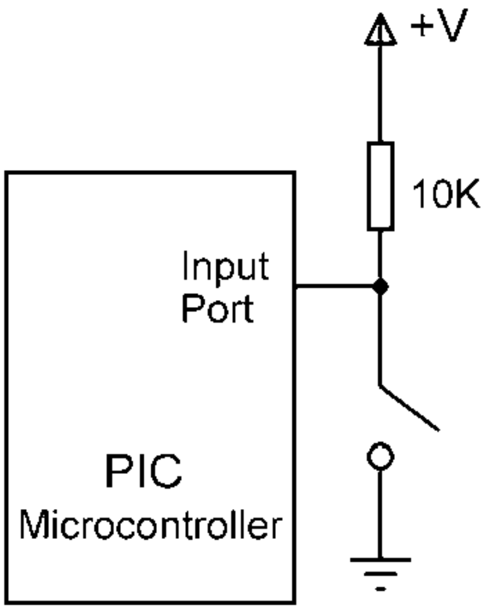
$$R = \frac{5 - 2\ \mathrm{V}}{10\ \mathrm{mA}} = \frac{3\ \mathrm{V}}{10\ \mathrm{mA}} = 0.3\ \mathrm{K}$$

The nearest physical resistor we can use is 330 Ohm.

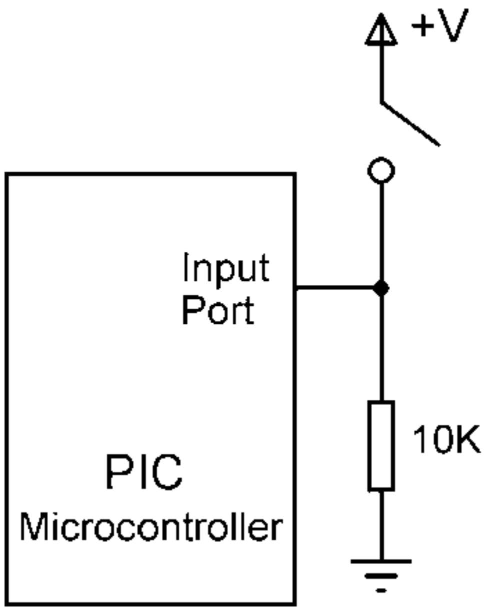
I/O Interfaces

Button Input:

- **In active low-button input**, **normally** the microcontroller input is pulled to logic 1 by the resistor (this is also called a **pull-up resistor**).
- **In active high-button input**, **normally** the microcontroller input is pulled to logic 0 by the resistor (**pull-down resistor**).



Active low-button input



Active high-button input

Write your first program using C++ Programming Language

Programming language

Structure of c program

```
#define   PI    3.14

 void main( ) {
   Variable  declaration
          Data  direction
           Initial values
      while(1) {
             ...............;
              ...............;
              ...............;
        }

      }
```

EX1 (Flash program):

- Write a PIC16F877A C program to toggle (flash) bit RB0 every 10 sec, use a suitable endless loop.

```
void main ( )  {
   TRISB.F0 = 0;        // Makes PORTB pin RB0 output pin
    while(1)  {                       // Infinite Loop
         PORTB.F0 = 1;            // RB0 ON
         Delay_ms(10000);         // 10 Sec Delay
         PORTB.F0 = 0;            //  RB0 OFF
         Delay_ms(10000);         //  10 Sec Delay
    }
 }
```

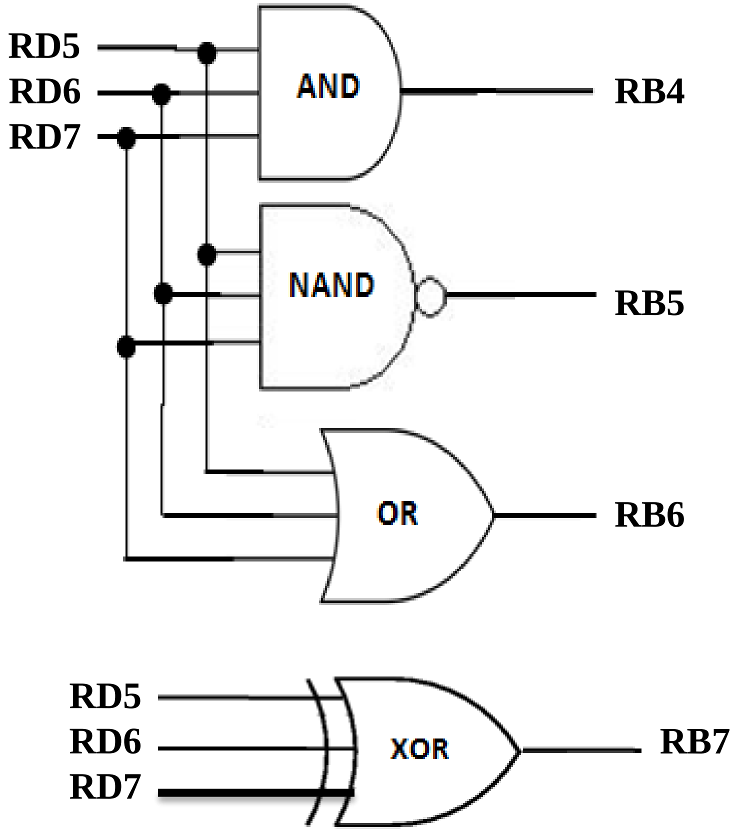
EX2:

- Write a PIC16F877A C program to toggle PORTB every 10 sec, use a suitable endless loop.

```
void main ( )  {
   TRISB  =  0;                          // Makes PORTB output port
   while(1)   {                          // Infinite Loop
       PORTB = 0xFF;                     // PORTB   ON
       Delay_ms(10000);                  // 10 Sec Delay
       PORTB  = 0;                       // PORTB   OFF
       Delay_ms(10000);                  //  10 Sec Delay
   }
}
```

EX3:

- Write a suitable C program for PIC16F877A to perform the following logic gates



Bitwise Operators

Operator	Operation
&	bitwise AND; compares pairs of bits and returns 1 if both bits are 1, otherwise returns 0
|	bitwise OR; compares pairs of bits and returns 1 if either or both bits are 1, otherwise returns 0
^	bitwise XOR; compares pairs of bits and returns 1 if the bits are complementary, otherwise returns 0
~	bitwise NOT; inverts each bit

```
void main( ) {

     bit  X ;   // variable of type  "bit"   ( MikroC is not case-sensitive )

     trisB = 0b00000000 ;   // Makes RB4 & RB5 & RB6 & RB7 output pins

     trisD = 0b11100000 ;   // Makes RD5 &RD6 & RD7 input pins

     portB=0 ;   // initializes portB with 0


    for(  ;  ;  ) {      // Infinite Loop

        x =  portD.f5 & portD.f6 & portD.f7 ;  // stores the AND operation in variable x

        portB.f4 = x ;    // AND output at pin RB4

        portB.f5 = ~x ;   // NAND output at pin RB5

        portB.f6 = portD.f5 | portD.f6 | portD.f7 ;   // OR output at pin RB6

        portB.f7 = portD.f5 ^ portD.f6 ^ portD.f7 ;   // XOR output at pin RB7

    }

}
```

EX4:

- 8 LEDs are connected to PORT B of a PIC16F877A microcontroller. Write a program to repeatedly control the LEDs with delay 1 sec according to the following table:

LED 0	LED 1	LED 2	LED 3	LED 4	LED 5	LED 6	LED 7
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
OFF	**ON**	OFF	OFF	OFF	OFF	OFF	OFF
OFF	OFF	**ON**	OFF	OFF	OFF	OFF	OFF
OFF	OFF	OFF	**ON**	OFF	OFF	OFF	OFF
OFF	OFF	OFF	OFF	**ON**	OFF	OFF	OFF
OFF	OFF	OFF	OFF	OFF	**ON**	OFF	OFF
OFF	OFF	OFF	OFF	OFF	OFF	**ON**	OFF
OFF	OFF	OFF	OFF	OFF	OFF	OFF	**ON**

```
void main() {
    trisb = 0;

    while(1){
        portb = 1;
        delay_ms(1000);
        portb = 2*portb;
        delay_ms(1000);
        portb = 2*portb;
        delay_ms(1000);
        portb = 2*portb;
        delay_ms(1000);
        portb = 2*portb;
        delay_ms(1000);
        portb = 2*portb;
        delay_ms(1000);
        portb = 2*portb;
        delay_ms(1000);
        portb = 2*portb;
        delay_ms(1000);
    }
}
```

Better Solution

```
void main() {
    int i;
    trisb = 0;

    while(1){
        portb = 1;
        delay_ms(1000);

        for(i=0; i<7 ; i++){
            portb = 2*portb;
            delay_ms(1000);
        }
    }
}
```

Ex5:

- 8 LEDs are connected to PORT B of a PIC16F877A microcontroller. Write a program to repeatedly control the LEDs with delay 500 ms according to the following table:

LED 0	LED 1	LED 2	LED 3	LED 4	LED 5	LED 6	LED 7
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
OFF	**ON**	OFF	OFF	OFF	OFF	OFF	OFF
OFF	OFF	**ON**	OFF	OFF	OFF	OFF	OFF
OFF	OFF	OFF	**ON**	OFF	OFF	OFF	OFF
OFF	OFF	OFF	OFF	**ON**	OFF	OFF	OFF
OFF	OFF	OFF	OFF	OFF	**ON**	OFF	OFF
OFF	OFF	OFF	OFF	OFF	OFF	**ON**	OFF
OFF	OFF	OFF	OFF	OFF	OFF	OFF	**ON**
OFF	OFF	OFF	OFF	OFF	OFF	**ON**	OFF
OFF	OFF	OFF	OFF	OFF	**ON**	OFF	OFF
OFF	OFF	OFF	OFF	**ON**	OFF	OFF	OFF
OFF	OFF	OFF	**ON**	OFF	OFF	OFF	OFF
OFF	OFF	**ON**	OFF	OFF	OFF	OFF	OFF
OFF	**ON**	OFF	OFF	OFF	OFF	OFF	OFF
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF

```
void main() {
    int i;
    trisb = 0;

    while(1){
        portb = 1;
        delay_ms(500);
        for(i=0; i<7 ; i++){
            portb = 2*portb;
            delay_ms(500);
        }

        for(i=0; i<7 ; i++){
            portb = portb/2;
            delay_ms(500);
        }
    }
}
```