**Name :**

**Roll no :**

**Group B Lab Assignment:** 7

**Subject :**PSDL

**Title :** ISR based buzzer on/off using Timer

**Assignment No: 7**

**Title :** Embedded C program for Timer programming ISR based buzzer on/off.

**Aim :** To write a C program to interface PIC18F4550 to Buzzer and switch it ON/OFF using Timer ISR.

**Experimental Setup:** MicroPIC18F board, USB cable, Power supply adaptor, MPLABx IDE, PICLoader software.

**Objective:**

* To Perform interfacing of real-world input and output devices to PIC18FXXX microcontroller.
* To learn embedded C programming for ISR based buzzer on/off using Timer

**Theory:**

**1). LED interfacing with PIC Microcontroller**

There are several devices which have proven themselves to be irreplaceable as they have been found quite convenient for different applications. A light-emitting diode is such a device and this chapter deals with various aspects of interfacing an LED with a PIC microcontroller.

An LED is basically a p-n junction diode which emits light energy when adequate voltage is given. Light-emitting diodes (LED) are basic display units in the electronics world. It has found its way to the low power display systems recently. It is the most commonly used indicator lights in all the applications. Ever since it’s widespread use as a reliable electronic component, LEDs have been ruling the electronic arena as no other similar technology has ever replaced them.

LED bulbs are available in different colours. Earlier LEDs made use of the infra-red wavelength and such LEDs are used even these days in remote-control circuits. LED bulbs making use of the ultraviolet wavelength is also available in the market

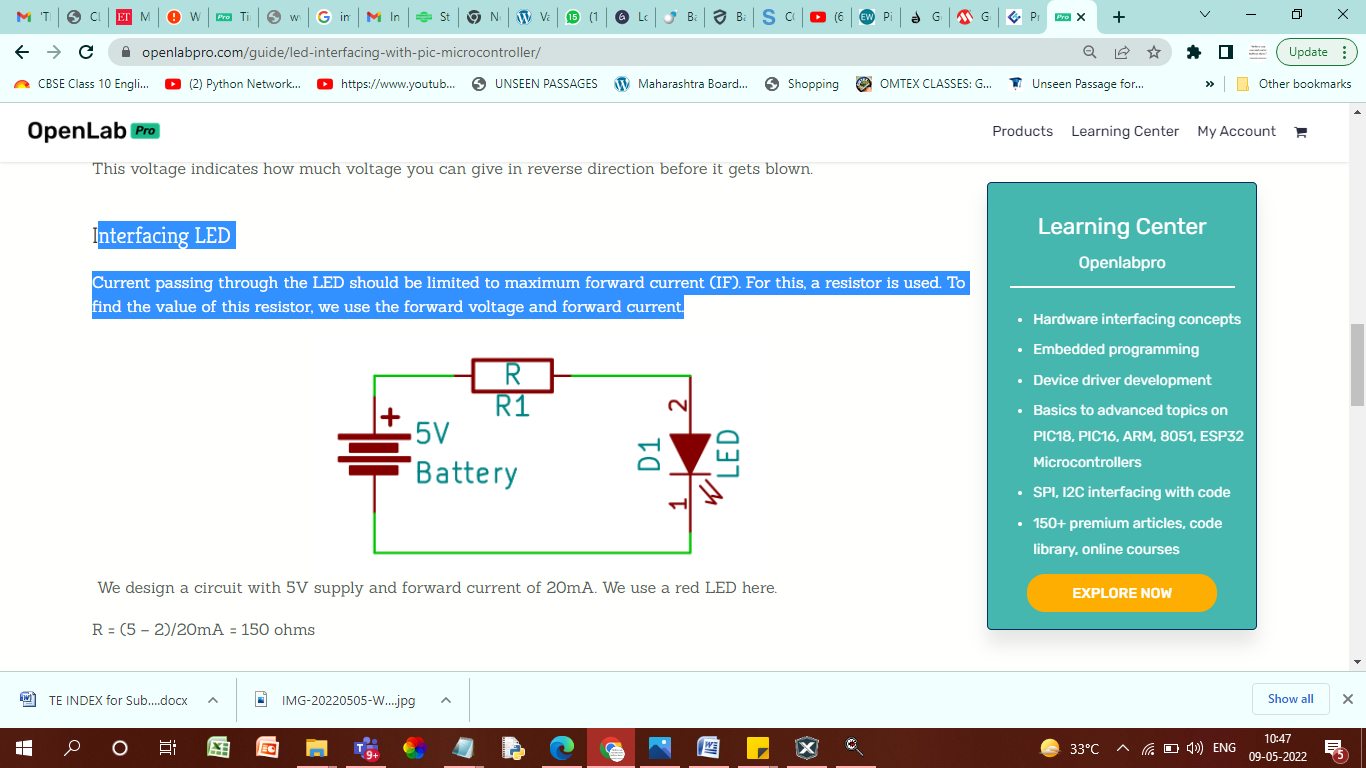
**1.2). LED requirements**

There are two technical specifications that should be taken into account while interfacing an LED. There are other specs like luminous intensity, dominant wavelength, operating temperature etc. Other specifications can be ignored while designing the circuit.

1. Peak forward current
2. Continuous forward current
3. Forward voltage
4. Reverse voltage

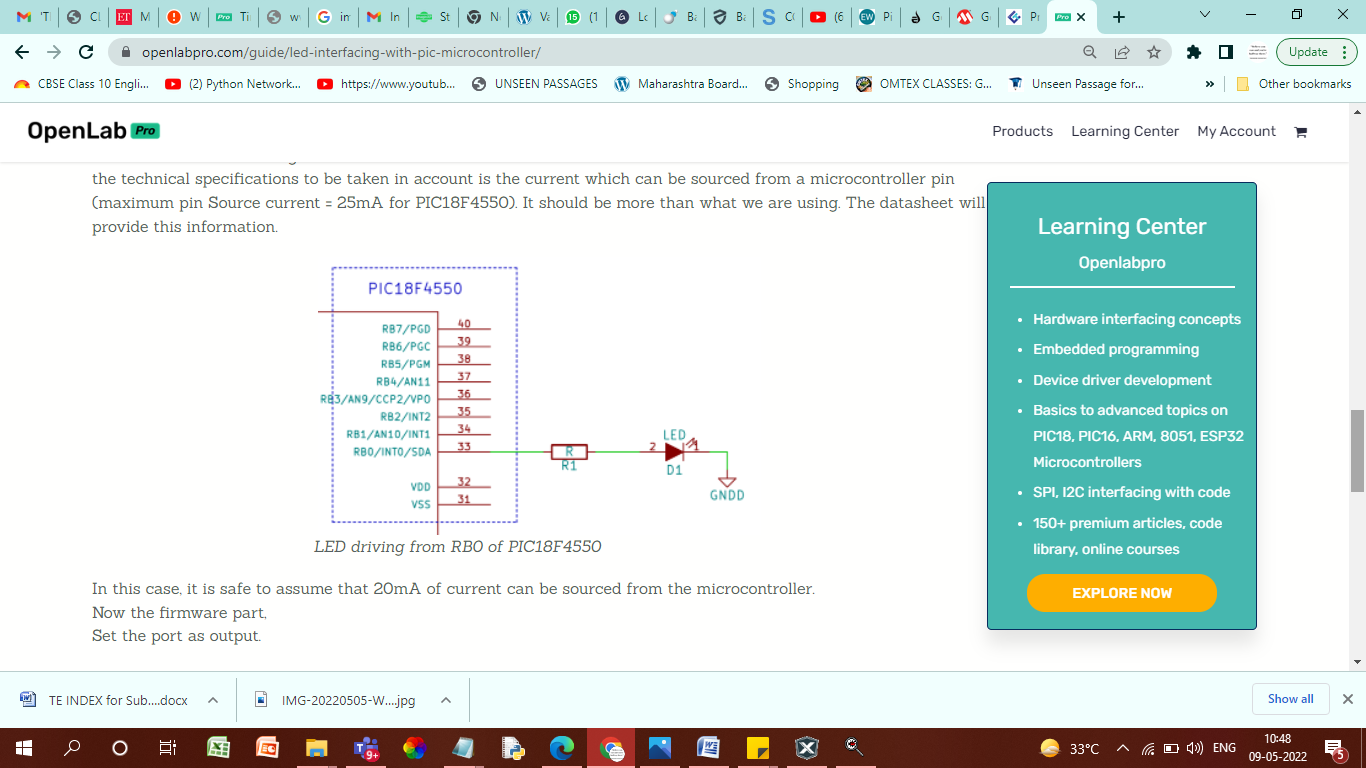
### Interfacing LED

Current passing through the LED should be limited to maximum forward current (IF). For this, a resistor is used. To find the value of this resistor, we use the forward voltage and forward current.



**1.3). LED interfacing with PIC**

Now we know how to design a circuit for LED. This circuit can be used to interface with PIC microcontroller. One of the technical specifications to be taken in account is the current which can be sourced from a microcontroller pin (maximum pin Source current = 25mA for PIC18F4550). It should be more than what we are using. The datasheet will provide this information



**2). Timer Delay Implementation in PIC18F4550**

In PIC18F4550 delay is implemented with general counting loops or with timers. Looping delays have their downsides such as inaccurate delay duration and blocking precious processor time. Thus delays of the order of milliseconds are preferred to be implemented with timers. PIC18F4550 has 4 timers, each of which could be used to provide delays of a few seconds. The delay required is configured with the help of timer control registers and preload value, in accordance with the Oscillator frequency.

**Procedure:**

**Step1:** Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step2:** Write the program in C language for interfacing Buzzer to PIC18F4550, using Timer ISR. (in program properties make sure to add the 0x800 offset).

**Step3:** Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step4:** Prepare the experimental setup by connecting the MicroPIC18F board to the PC using USB cable. Power ON the Board. Check for the USBtoSerial COMx allocated by the PC**.**

**Step5:** Using the PICLoader Software flash the hex file in the PIC18F4550.

**Step6:** Press reset button and execute the program.

**.**

**Source code :**

#include <pic18f4550.h> /\* Contains PIC18F4550 specifications \*/

#define Buzzer LATAbits.LATA5 /\* Define buzzer pin \*/

unsigned int count = 0;

void interrupt Timer1\_ISR()

{

if(TMR1IF==1)

{

//1 ms delay time in timer

TMR1L = 0x20;

TMR1H = 0xD1;

count ++;

if (count >= 1000) //measure upto 1000 ms i.e. 1 seconds

{

Buzzer = ~Buzzer; /\* Toggle buzzer pin \*/

count = 0; //reset count

}

TMR1IF = 0; //timer1 overflow flag to 0

}

}

void main()

{

TRISB=0; /\* Set as output port \*/

TRISAbits.TRISA5 = 0; //set buzzer pin RA5 as output

GIE=1; /\* Enable Global Interrupt \*/

PEIE=1; /\* Enable Peripheral Interrupt \*/

TMR1IE=1; /\* Enable Timer1 Overflow Interrupt \*/

TMR1IF=0;

/\* Enable 16-bit TMR1 register,no pre-scale,internal clock, timer OFF \*/

T1CON=0x80; /\*1:8 prescale\*/

TMR1L = 0x20;

TMR1H = 0xD1;

TMR1ON=1; /\* Turn ON Timer1 \*/

while(1);

}

**Result:** Check if the buzzer is sounding ON/OFF and the ISR is getting executed with the specified timer delay. You can change the delay and vary the sounding rate.

**Conclusion:** Thus, we have studied C program to interface PIC18F4550 to Buzzer and switch it ON/OFF using Timer ISR.