```
//Description:
//
//
//This example is using timer 0 and interrupt to make LED (A0) on for 1 second and off for 1
second.
//As we want every 1 second to have 1 interrupt, we need to choose proper starting value of
timer0.
//Choose prescaler: 1/256, to minimize the timer increment frequency.
//Then the timer increment frequency = (Fosc/4)*(1/256)= (48MHz/4)*(1/256)= 46875 Hz.
//
//Because the period of the interupt is 1 second,then amount of the timer register value for 1
second is 46875*1second  = 46875.
//Timer0 starting value = 65535 - 46875 = 18660 = 0x48E4.
//
//When the interrupt is generated, then in the interrupt service routine we just need to toggle
the state of the LED.
//Finally, we can see the LED on for 1 second and off for other 1 second.
//
//

#include <p18f4550.h>
#include <timers.h>    // include the timer library


void timer0_isr(void); //Interrupt service routine prototype
int state=1;           //The state of A0.

//Always include this code, it's necessary when using a bootloader
extern void _startup (void);
#pragma code _RESET_INTERRUPT_VECTOR = 0x000800
void _reset (void)
{
_asm goto _startup _endasm
}
#pragma code
#pragma code  _HIGH_INTERRUPT_VECTOR  = 0x000808
void high_ISR (void)
{

}
#pragma code
#pragma code _LOW_INTERRUPT_VECTOR = 0x000818
void low_ISR (void)
{ //Pre: The interrupt priority is defined as low and enabled.The low priority interrupt service
routine is called
  //Post: Execute the timer0_isr function.

_asm goto timer0_isr _endasm; //when the low priority interrupt is executed, go to timer0 service
routine
}
#pragma code
//End bootloader code


#pragma interrupt timer0_isr //Low priority interrupt service routine
  void timer0_isr(void)
   { //Pre: The low_ISR function is called.One integer for the A0 state is defined.
     //Post: LED (A0) can toggle the state each second.



     INTCONbits.TMR0IF = 0;         //Reset Timer0 interrupt flag
     WriteTimer0(0x48E4);           //give new start value to the timer0
     LATAbits.LATA0 =state;         //give the state to A0.
     state= !state;                 //Toggle the state of A0.
   }
  void main(void)
  {  //Pre: The timer library is included
     //Post: The timer0 overflow interrupt can be generated at every 1 second.

    TRISA = 0b11111110;            //Set channel A0 as output
    LATA=0;                        //Initialize Port A.
    ADCON1 = 0b00001111;           //All ADC disabled
    RCONbits.IPEN = 1;             //Enable priority levels on interrupts
    RCONbits.SBOREN = 0;           //Disable BOR
```

```
  OpenTimer0( TIMER_INT_ON &     //Interrupt enabled
              T0_16BIT &         //Set timer0 as 16 bit mode.
              T0_SOURCE_INT &    //choose Internal clock source (TOSC)
              T0_PS_1_256        //Prescale Value: 1:256
             );

  WriteTimer0(0x48E4);           // set overflow interrupt at every 1 ms.


  INTCON = 0b11000000 ;         // enable high and low priority interrupts
  INTCON2bits.TMR0IP =0;        //Set interrupt priority as low
  INTCONbits.TMR0IE=1;          //Enables the TMR0 overflow interrupt

  while(1)
  {
    //add your codes .................
   }
}
```