```
//Description:
//
//This example is using the timer2 and interrupt to make A0 high for 100ms and then low for 100ms.
//The timer2 register has 8 bits and we use the internal clock as clock source.
//Suppose we choose the Prescaler as 1:16 and postscaler as 1:15.
//Instead of overflow interrupt, timer2 use comparsion.
//When the value of TMR2 (timer2 register value) matches the value of PR2, then the interrupt
will be generated.
//
//Please notice that the Postscaler can also decrease the timer frequency. In this
example(postscaler: 1:15),
//it means that the match happens 15 times, then interrupt can be generated once.
//
//Timer increment frequency = (Fosc/4)*Prescaler*Postscaler = (48MHz/4)*(1/16)*(1/15) = 50kHz.
//
//Timer2 register starts from 0 to increment till it reaches the value PR2, then generate one
interrupt.After that roll over to 0.
//As we want to have one interrupt every 1 ms, then
//PR2 = 50kHz*1ms = 50.
//
//Then we count the interrupts. When the number of interrupts reaches 100, we know the time is
100 ms.We can switch on AO.
//When the number of interrupts reaches 200, we know it takes 200 ms. We switch off A0.Then we
count the interrupts from 0 again.
//This is how we made the A0 high for 100 ms then low for 100ms.
//
//



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

void timer2_isr(void); //Interrupt service routine prototype
int i=0;               //interrupt counter

//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)
{
  //Pre: The interrupt priority is defined as high and enabled.The high priority interrupt
service routine is called
  //Post: Execute the timer2_isr function.
 _asm goto timer2_isr _endasm
}
#pragma code
#pragma code _LOW_INTERRUPT_VECTOR = 0x000818
void low_ISR (void)
{
;
}
#pragma code
//End bootloader code


#pragma interrupt timer2_isr //High priority interrupt service routine
 void timer2_isr(void)
 {
   //Pre: The high_ISR function is called.One integer for counting interrupts is defined.
   //Post:Swith on A0 at 100ms, then switch off it at 200ms.

   i++;               // counting the interrupts
  PIR1bits.TMR2IF=0;   //reset the interrupt flag.


  if(i==100)           //when interrupt number reaches 100, switch on A0.  at time: 100*1ms =
100ms, switch on A0.
   {
```

```
    LATAbits.LATA0=1;  //switch A0 to high

    }

   if(i==200)          //when interrupt number reaches 200, switch off A0. at time: 200*1ms =
200ms, switch off A0.
    {
    LATAbits.LATA0=0;  //switch AO to low
    i=0;               //reset the count value
    }
  }
 void main(void)
 {
  //Pre: The timer library is included.
  //Post: The timer2 match interrupt can be generated at every 1 ms.

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

 OpenTimer2( TIMER_INT_ON &       //Interrupt enabled
             T2_PS_1_16 &         //Set the Prescale Value: 1:16
             T2_POST_1_15         //Set Postscale Value: 1:15.
           );


 PR2 = 50;                        // Set the period register to 50, which means that when the
timer increase to 50,
                                  // then the interrupt will be generated.
                                  //interrupt at every (50/50kHz) = 1ms

 INTCON = 0b10000000;             //Enable the high priority interrupts


 IPR1bits.TMR2IP=1;              // Set the timer2 interrupt priority to high.
 PIE1bits.TMR2IE=1;              // Enable the timer2 interrupt

  while(1)
  {
    //Add codes here....
  }
 }
```