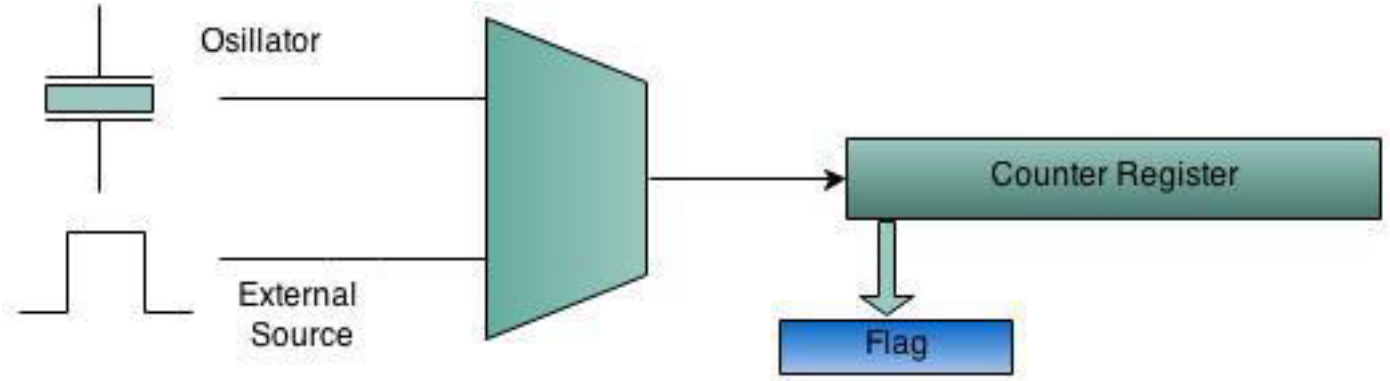
IBERO
PUEBLA ®
IDIT

Elementos Programables II
TIMER/COUNTER

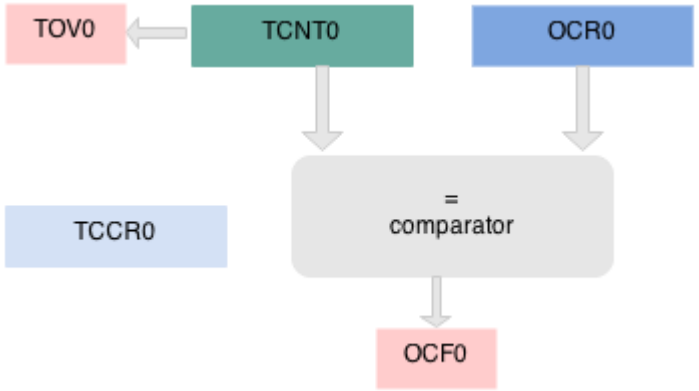
Introduccion

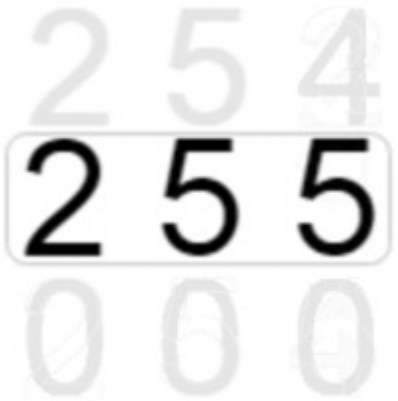


Los temporizadores / contadores son parte esencial de cualquier MCU moderna. Los temporizadores / contadores son una unidad independiente dentro de un microcontrolador. Básicamente se ejecutan independientemente de la tarea que realiza la CPU. Por lo tanto, son muy útiles y se utilizan principalmente para lo siguiente:

- **Temporizador interno**: como temporizador interno, la unidad marca la frecuencia del oscilador. La frecuencia del oscilador se puede alimentar directamente al temporizador o se puede preescalar. En este modo, generaba demoras precisas. O como máquina precisa de conteo de tiempo.
- **Contador externo**: en este modo, la unidad se utiliza para contar eventos en un pin externo específico en una MCU.
- **Generador de modulación de ancho de pulso (PWM):** PWM se utiliza en el control de velocidad de motores y otras aplicaciones.

Timer Funcionamiento





El timer es un temporizador de N bits. Básicamente significa que puede contar de 0 a 2 ^ N. El registro TCNT0 retiene el conteo del timer y se incrementa en cada "tic" del timer. Si el temporizador está activado, marca de 0 al valor maximo y se desborda. Si lo hace, se establece un indicador de overflow del timer (TOV).

Timers Registros Basicos

TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0**[7:0]**								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	**WGM02**	**CS02**	**CS01**	**CS00**	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	**WGM02**	**CS02**	**CS01**	**CS00**	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Este registro nos ayuda a inicializar el timer y definir la velocidad con la que un “tic” se realiza.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

$$F_{timer} = \frac{F\_CPU}{Prescaler}$$

P. ej. TCCR0B|=(1<<CS01)//Prescalador de 8

$$F_{timer} = \frac{16000000}{8} = 2000000 = 2MHz$$

$$F = \frac{1}{T_{tic}} \quad s \qquad T_{tics} = \frac{1}{F} = \frac{1}{2MHz} = 0.5\mu S$$

Codigo Basico Timer

```
1.  #include <avr/io.h>
2.void timer0_init()
3.  {
4.  TCCR0B|=(1<<CS01);//Inicializar timer con prescala de 8
5.TCNT0=0;//Reiniciar contador en cero
6.  }
7.  int main(void)
8.  {
9.      DDRB|=(1<<5);
10.     timer0_init();
11.     while (1)
12.     {
13.      if(TCNT0>=250)//Checa el contador  para un tiempo definido
14.       {
15.          PORTB^=1<<5;//Toggle en Led
16.          TCNT0=0;//Reset de contador
17.       }
18.     }
19.}
```

TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

El registro de indicador de interrupción del timer/counter (TIFR) contiene los dos indicadores básicos que necesitamos el TOV0. TOV0 se establece cuando se produce un overflow en Timer / Counter0. TOV0 es borrado por el hardware al ejecutar el correspondiente vector de itnerrupcion. Alternativamente, TOV0 se borra escribiendo uno lógico en la bandera.

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

El registro de mascara de interrupción del timer/counter (TIMSK) contiene la habilitación de las interrupciones por timer. Cuando en el bit TOIE0 se escribe en uno, se habilita la interrupción de desbordamiento del timer/counter.

Codigo Overflow Usando Banderas

```
#include <avr/io.h>
void timer0_init(){
  TCCR0B|=(1<<CS01);//Inicializar timer con prescala de 8
TCNT0=0;//Reiniciar contador en cero
}
int main(){
 uint16_t timerOverflowCount=0;
 DDRB|=1<<5;          //configure PORTD as output
 while(1){
   if(TIFR0 & 0x01){//Revisa que en el bit 0x01(TOV0) haya un 1
     timerOverflowCount++;//AUmenta el contador del overflow
      TCNT0 = 0x00;//Resetea el contador
      TIFR0=0x01;//Resetea La Bandera
    }
   if (timerOverflowCount>=800){//Espera 800
      PORTD ^= (1 << 5);//Toggle Led
      timerOverflowCount=0;//Reset overflow counter
    }
  }
}
```

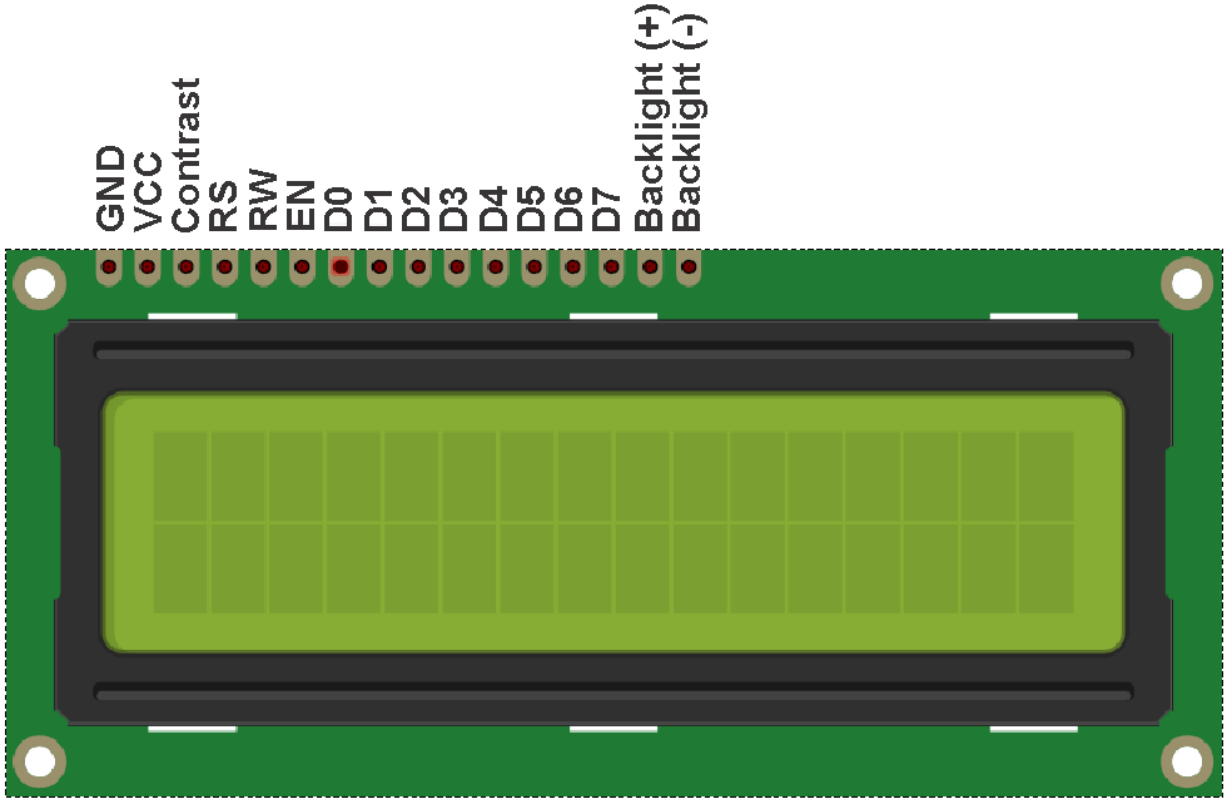
Codigo Overflow Usando Banderas

```
1.  #include <avr/io.h>
2.  void timer0_init(){
3.    TCCR0B|=(1<<CS01);//Inicializar timer con prescala de 8
4.TCNT0=0;//Reiniciar contador en cero
5.}
6.int main(){
6.   uint16_t timerOverflowCount=0;
8.   DDRB|=1<<5;          //configure PORTD as output
9.   timer0_init();
10.  while(1){
11.    if(TIFR0 & 0x01){//Revisa que en el bit 0x01(TOV0) haya un 1
12.      timerOverflowCount++;//AUmenta el contador del overflow
13.       TCNT0 = 0x00;//Resetea el contador
14.       TIFR0=0x01;//Resetea La Bandera
15.     }
16.    if (timerOverflowCount>=800){//Espera 800 overflows ~100ms
17.       PORTD ^= (1 << 5);//Toggle Led
18.       timerOverflowCount=0;//Reset overflow counter
19.     }
20.   }
21.}
```

Codigo Overflow Usando Interrupciones

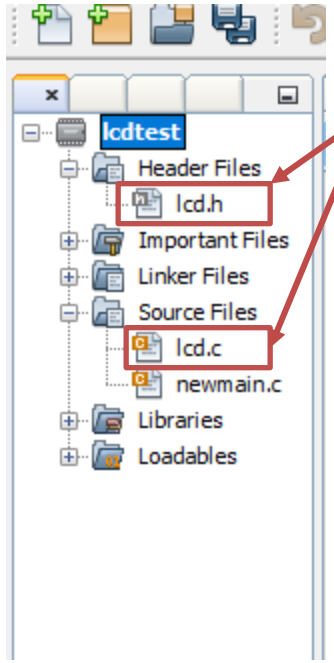
```
1. #include <avr/io.h>
2. #include <avr/interrupt.h>
3. volatile uint16_t timerOverflowCount=0;
4. void timer0_init(){
5.   TCCR0B|=(1<<CS01);//Inicializar timer con prescala de 8
6.   TCNT0=0;//Reiniciar contador en cero
7.   TIMSK0|=(1<<TOIE0); //Habilita interrupciones de timer
8.   sei();//Habilita interrupciones globales
9. }
10. ISR(TIMER0_OVF_vect){//ISR para el vector de timer0
11. timerOverflowCount++;//Aumenta el contador del overflow
12. }
13. int main(){
14.   DDRB|=1<<5;      //configure PORTD as output
15.   timer0_init(); //llamado de funcion de timer
16.   while(1){
17.     if (timerOverflowCount>=800){//Espera 800 ~100ms
18.       PORTB ^= (1 << 5);//Toggle Led
19.       timerOverflowCount=0;//Reset overflow counter
20.     }
21.   }
22. }
```

Uso de LCD-Cableado



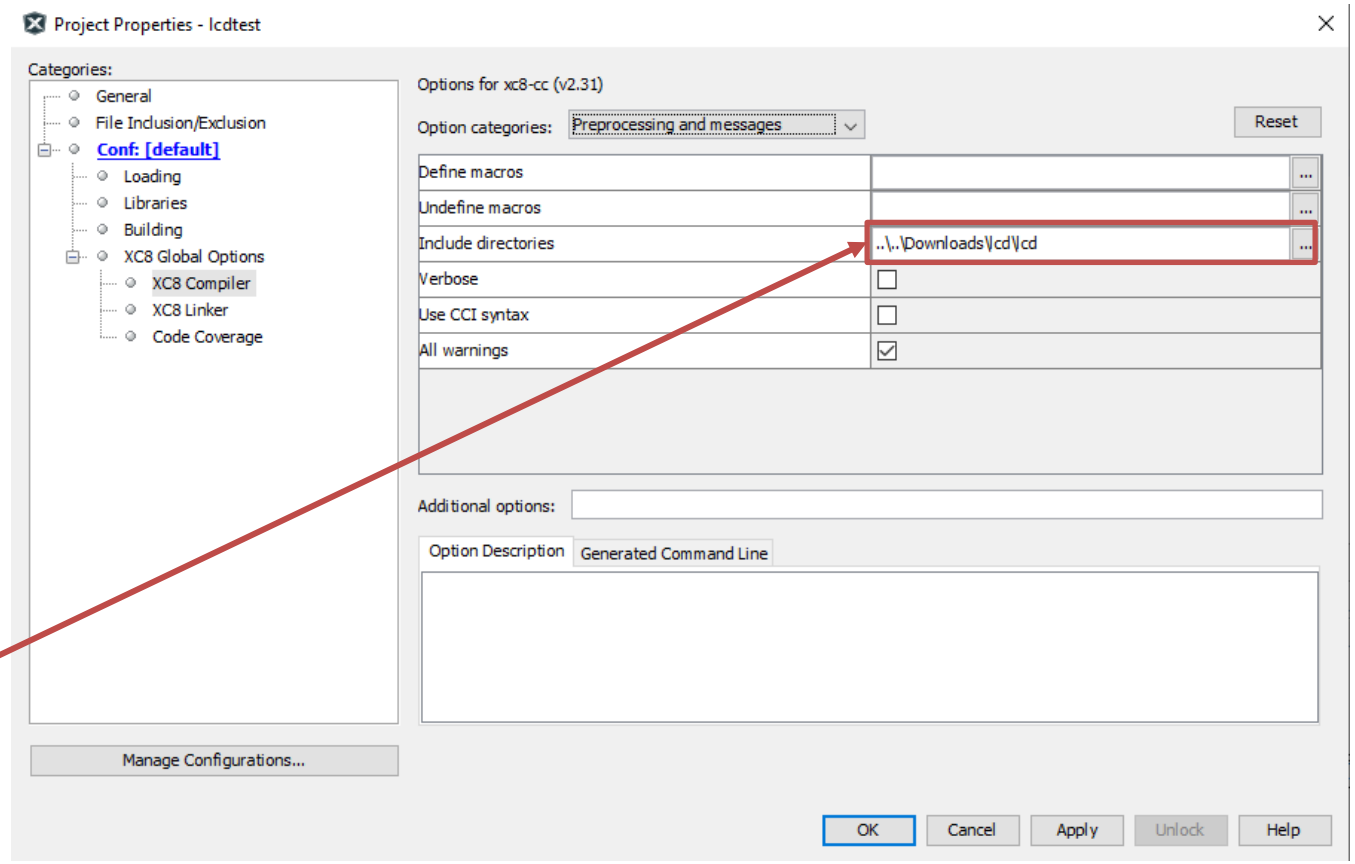

```
#define LCD_IO_MODE 1
#define LCD_PORT PORTA
#define LCD_DATA0_PORT LCD_PORT
#define LCD_DATA1_PORT LCD_PORT
#define LCD_DATA2_PORT LCD_PORT
#define LCD_DATA3_PORT LCD_PORT
#define LCD_DATA0_PIN 0
#define LCD_DATA1_PIN 1
#define LCD_DATA2_PIN 2
#define LCD_DATA3_PIN 3
#define LCD_RS_PORT LCD_PORT
#define LCD_RS_PIN 4
#define LCD_RW_PORT LCD_PORT
#define LCD_RW_PIN 5
#define LCD_E_PORT LCD_PORT
#define LCD_E_PIN 6
```

Anexar Librerías Propias o Externas

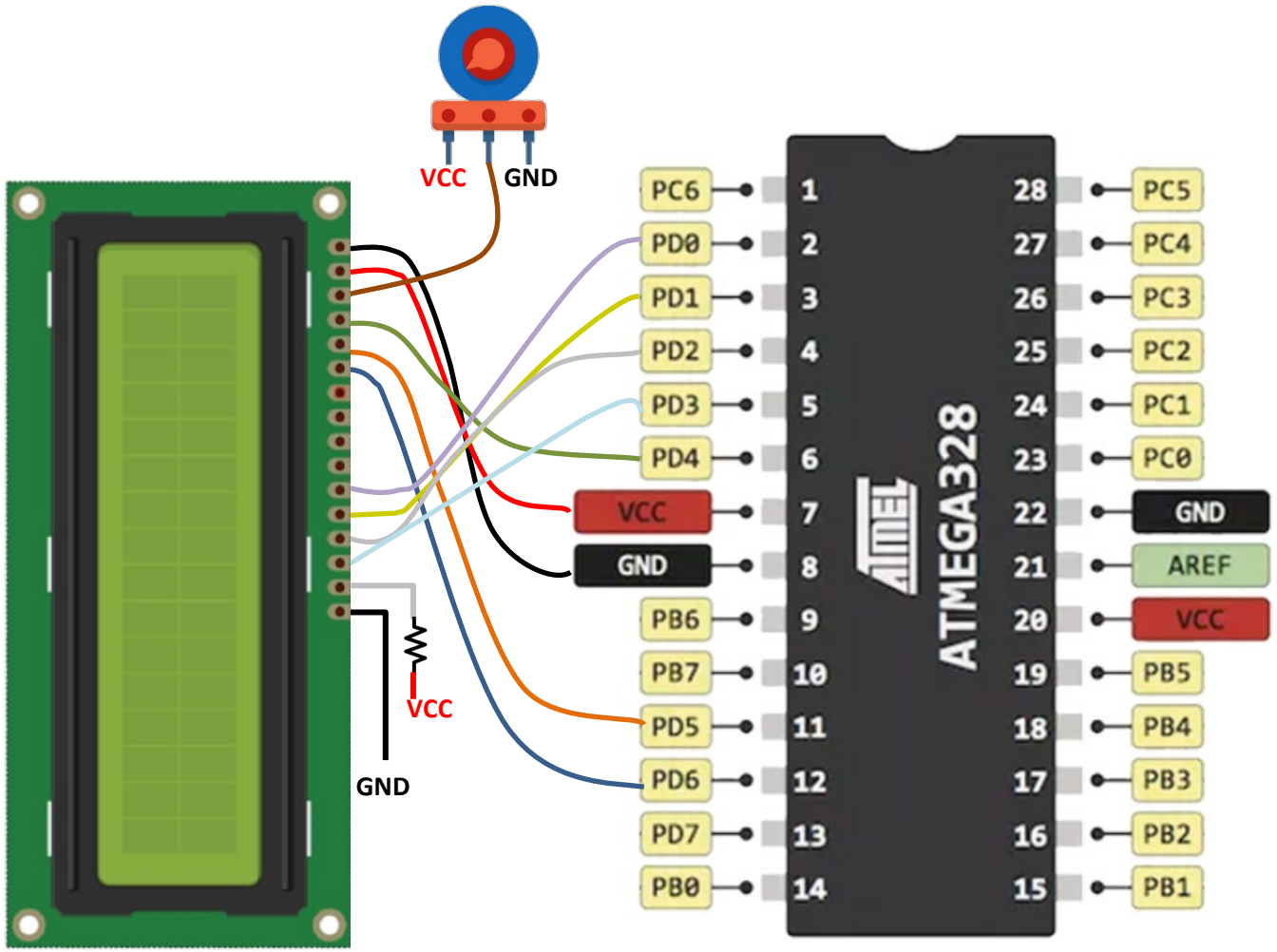


Agregar los archivos al proyecto

Agregar la dirección de la libreria



Uso de LCD-Cableado Ejemplo




```
#define LCD_PORT          PORTD
#define LCD_DATA0_PORT    LCD_PORT
#define LCD_DATA1_PORT    LCD_PORT
#define LCD_DATA2_PORT    LCD_PORT
#define LCD_DATA3_PORT    LCD_PORT
#define LCD_DATA0_PIN     0
#define LCD_DATA1_PIN     1
#define LCD_DATA2_PIN     2
#define LCD_DATA3_PIN     3
#define LCD_RS_PORT       LCD_PORT
#define LCD_RS_PIN        4
#define LCD_RW_PORT       LCD_PORT
#define LCD_RW_PIN        5
#define LCD_E_PORT        LCD_PORT
#define LCD_E_PIN         6
```

DENTRO DE LCD.H

Uso de LCD-Comandos Basicos

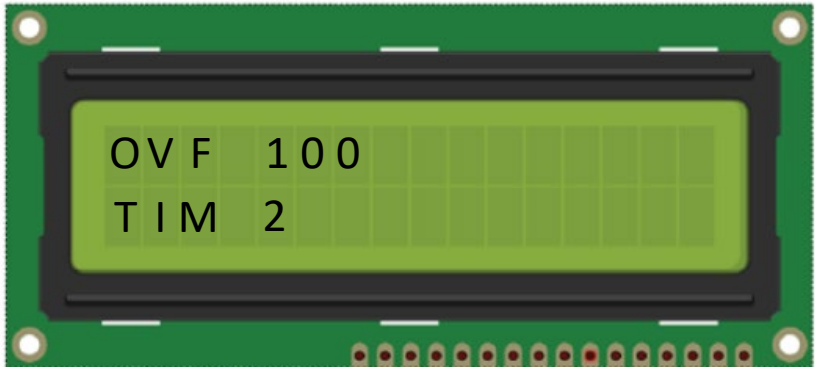
```
1.lcd_init(LCD_DISP_ON);//INICIALIZACION DE LCD CON CURSOR

2.lcd_clrscr();//BORRAR PANTALLA

3.lcd_gotoxy(POSX, POSY);//MOVER CURSOR A UNA POSICION

4.lcd_putc('A');//ESCRIBIR CARÁCTER

5.lcd_puts("STRING");//ESCRIBIR STRIGN
```

Uso de LCD-Codigo Ejemplo

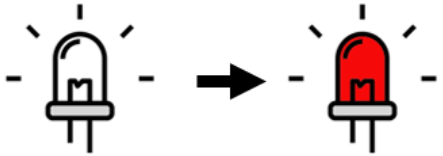
```
#define F_CPU 1000000UL
#include <xc.h>
#include "lcd.h"
#include <util/delay.h>
#include <stdio.h>
int main(void){
    lcd_init(LCD_DISP_ON);//INICIALIZACION DE LCD CON CURSOR
    lcd_clrscr();//BORRAR PANTALLA
    lcd_gotoxy(0, 0);//MOVER CURSOR A UNA POSICION
    lcd_putc('~');//ESCRIBIR CARACTER
    lcd_puts("HOLA");//ESCRIBIR STRING
    _delay_ms(2000);
    char str[5]; //declarar mi variable de texto
    while(1){
        for(int i=0; i<9; i++){
            lcd_clrscr();//LIMPIAR PANTALLA
            sprintf(str, "%d", i);//CONVERTIR i A STRING Y GUARDARLO EN str
            lcd_puts(str);//IMPRIMIR str
            _delay_ms(1000);
        }
    }
    return 0;
}
```

Ejercicios Clase

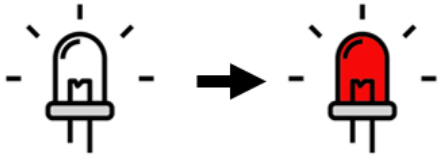
1. Imprimir OVF y Tiempo real



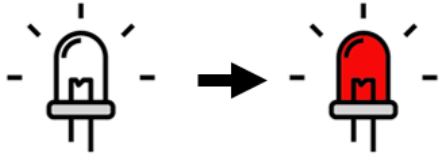
2. Parpadeo en distintas Frecuencias



1.5 Segs



2 Segs



5 Segs