```
/*
 *File: LCD.c
 * Se tomo y se adaptaron las librerias de Ligo George
 * de la pagina www.electrosome.com
 * Pagina: https://electrosome.com/lcd-pic-mplab-xc8/
 */


#include "LCD.h"

void Lcd_Port(char a) {
    if (a & 1)
        D0 = 1;
    else
        D0 = 0;

    if (a & 2)
        D1 = 1;
    else
        D1 = 0;

    if (a & 4)
        D2 = 1;
    else
        D2 = 0;

    if (a & 8)
        D3 = 1;
    else
        D3 = 0;

    if (a & 16)
        D4 = 1;
    else
        D4 = 0;

    if (a & 32)
        D5 = 1;
    else
        D5 = 0;

    if (a & 64)
        D6 = 1;
    else
        D6 = 0;

    if (a & 128)
        D7 = 1;
    else
        D7 = 0;
}

void Lcd_Cmd(char a) {
    RS = 0;
    Lcd_Port (a);
    EN = 1;
    __delay_ms(5);

    EN = 0;

}


void Lcd_Clear(void) {

    Lcd_Cmd(0);

    Lcd_Cmd(1);

}


void Lcd_Init(void) {

    Lcd_Port(0x00);

    __delay_ms(30);

    Lcd_Cmd (0x30);

    __delay_ms(6);

    Lcd_Cmd (0x30);

    __delay_ms(15);

    Lcd_Cmd (0x30);


//////////////////////////////////////////////////
////////////////////////

    Lcd_Cmd (0x08);

    Lcd_Cmd (0x01);

    Lcd_Cmd (0x08);

    Lcd_Cmd (0x06);

}

void Lcd_Set_Cursor(char a, char b) {

    char temp, z, y;

    if (a == 1) {

        temp = 0x80 + b - 1;

        z = temp >> 4;

        y = temp & 0x0F;

        Lcd_Cmd(z);

        Lcd_Cmd (y);

    }else if (a == 2) {

        temp = 0xC0 + b - 1;

        z = temp >> 4;

        y = temp & 0x0F;

        Lcd_Cmd(z);

        Lcd_Cmd (y);

    }

}


void Lcd_Write_Char(char a){

    char temp;

    RS = 1;

    Lcd_Port(temp);

    EN = 1;

    __delay_us(40);

    EN = 0;

}


void Lcd_Write_String(char *a){

    int i;

    for (i = 0; a[i] != '\0'; i++)

        Lcd_Write_Char(a[i]);

}
```

```
/*
 *File: LCD.h
 * Se tomo y se adaptaron las librerias de Ligo George
 * de la pagina www.electrosome.com
 * Pagina: https://electrosome.com/lcd-pic-mplab-xc8/


 */
#ifndef LCD_H
#define LCD_H
#endif


#ifndef _XTAL_FREQ
#define _XTAL_FREQ 8000000
#endif

#ifndef RS
#define RS PORTAbits.RA0
#endif

#ifndef RW
#define RW PORTAbits.RA1
#endif

#ifndef EN
#define EN PORTAbits.RA2
#endif

#ifndef D0
#define D0 PORTDbits.RD0
#endif

#ifndef D1
#define D1 PORTDbits.RD1
#endif

#ifndef D2
#define D2 PORTDbits.RD2
#endif

#ifndef D3
#define D3 PORTDbits.RD3
#endif

#ifndef D4
#define D4 PORTDbits.RD4
#endif

#ifndef D5
#define D5 PORTDbits.RD5
#endif

#ifndef D6
#define D6 PORTDbits.RD6
#endif
```

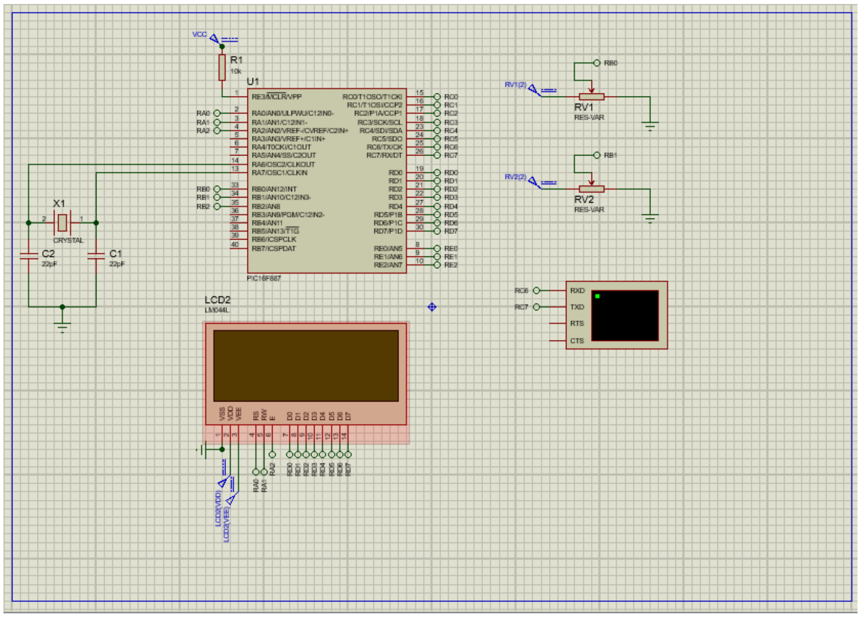
```
#ifndef D7
#define D7 PORTDbits.RD7
#endif



#include <xc.h>


void Lcd_Port (char a);
void Lcd_Cmd (char a);
void Lcd_Clear (void);
void Lcd_Set_Cursor (char a, char b);
void Lcd_Init (void);
void Lcd_Write_Char (char a);
void Lcd_Write_String (char *a);
void Lcd_Shift_Right (void);
void Lcd_Shift_Left (void);



***************************************
***************************************
*
 * File:   mainTemplate.c
 * Author: Mariandree Rivera
 * Carnet: 18178
 * Archivo template
 *
 * Created on February 08, 2021,
```

```
/**************************************
***************************************
**
*/
//**************************
// Importacion de librerias
//**************************
#include <xc.h>
#include <stdint.h>
#include "LCD.h"


//**************************
// Palabra de configuracion
//**************************


// CONFIG1
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = OFF      // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)
```

```
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disabled)

#pragma config IESO = OFF       // Internal External Switchover bit (Internal/External Switchover mode is disabled)

#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)

#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)


// CONFIG2

#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)

#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits (Write protection off)




#define _XTAL_FREQ 8000000

#define RS PORTAbits.RA0

#define RW PORTAbits.RA1

#define EN PORTAbits.RA2

#define D0 PORTDbits.RD0

#define D1 PORTDbits.RD1

#define D2 PORTDbits.RD2

#define D3 PORTDbits.RD3

#define D4 PORTDbits.RD4

#define D5 PORTDbits.RD5

#define D6 PORTDbits.RD6

#define D7 PORTDbits.RD7
```

```
//**************************

// Variables

//**************************


char counter = 0;


//**********

// Interrupciones

//**********


void __interrupt() ISR()

{

 //esto se activara si la interrupcion viene del receptor en el UART

    if (PIR1bits.RCIF == 1)

    {

        //esto nos limpiara la interrupcion

        PIR1bits.RCIF = 0;

        TXREG = (RCREG + 1);

        while (TXSTAbits.TRMT == 0);


    }

}

//**************************

// Prototipos de funciones

//**************************

void setup(void);

void UART_Init(void);
void __interrupt() ISR();
void Lcd_Init(void);
//**************************
// Ciclo principal
//**************************

void main(void) {
  unsigned int a;
  setup();
  void Lcd_Init(void);
  UART_Init();
  //**************************
  // Loop principal
  //**************************
  while (1) {

    Lcd_Clear();
    //      if (PORTCbits.RC7 == 0) {
    Lcd_Set_Cursor(1, 1);
    Lcd_Write_String("Hola Mundo");
    //      }
    //      if (PORTCbits.RC7 == 0) {
    Lcd_Set_Cursor(2, 1);
    Lcd_Write_String("Adios Mundo");
    __delay_ms(2000);
    Lcd_Clear();
    //      }
    //      if (PORTCbits.RC7 == 0) {
    Lcd_Set_Cursor(1, 1);
    Lcd_Write_String("Developed By");
    Lcd_Set_Cursor(2, 1);
    Lcd_Write_String("electroSome");
    __delay_ms(2000);
    Lcd_Clear();
    //      }



    //      Lcd_Set_Cursor(1, 1);
    //
Lcd_Write_String("www.electroSome.com");

    for (a = 0; a < 15; a++) {
      __delay_ms(300);
      Lcd_Shift_Left();
    }

    for (a = 0; a < 15; a++) {
      __delay_ms(300);
      Lcd_Shift_Right();
    }

    Lcd_Clear();
    Lcd_Set_Cursor(2, 1);
    Lcd_Write_Char('M');
    Lcd_Write_Char('S');
    __delay_ms(2000);
  }
}
```

```
//**************************
// Configuracion
//**************************

void setup(void) {

   ANSEL = 0;
   ANSELH = 0b00000001;
   TRISA = 0;
   PORTA = 0;
   TRISB = 0b00000111;
   PORTB = 0;
   TRISD = 0;
   PORTD = 0;
   TRISE = 0;
   PORTE = 0;

}

//**************************
// Funciones
//**************************
void UART_Init()
{
   //Seleccionara los 8bits para la transmisicion de los datos.
   TXSTAbits.TX9 = 0;
   //habilitar la transmision
   TXSTAbits.TXEN = 1;
   //habilitacion del modo asincrono
   TXSTAbits.SYNC = 0;
   //operacion en velocidad lenta
   TXSTAbits.BRGH = 0;
   //  habilita el puerto serial
   RCSTAbits.SPEN = 1;
   // habilita que constantemente se reciban datos
   RCSTAbits.CREN = 1;

   //Baudrate 10417
   SPBRG = 11;

   //activacion de las interrupciones
   INTCONbits.GIE =  1;
   // habilitacion de las interrupciones perifericas, ver diagrama.
   INTCONbits.PEIE = 1;

   //habilita las interrupciones del receptor.
   PIE1bits.RCIE = 1;
   // limpia el flag de la interrupcion
   PIR1bits.RCIF = 0;
}
```


VCC
R1
10k
U1
1 RE3/MCLR/VPP
2 RA0 RA0/AN0/ULPWU/C12IN0-
3 RA1 RA1/AN1/C12IN1-
4 RA2 RA2/AN2/VREF-/CVREF/C2IN+
5 RA3/AN3/VREF+/C1IN+
6 RA4/T0CKI/C1OUT
7 RA5/AN4/SS/C2OUT
14 RA6/OSC2/CLKOUT
13 RA7/OSC1/CLKIN
33 RB0 RB0/AN12/INT
34 RB1 RB1/AN10/C12IN3-
35 RB2 RB2/AN8
36 RB3/AN9/PGM/C12IN2-
37 RB4/AN11
38 RB5/AN13/T1G
39 RB6/ICSPCLK
40 RB7/ICSPDAT
RC0/T1OSO/T1CKI 15 RC0
RC1/T1OSI/CCP2 16 RC1
RC2/P1A/CCP1 17 RC2
RC3/SCK/SCL 18 RC3
RC4/SDI/SDA 23 RC4
RC5/SDO 24 RC5
RC6/TX/CK 25 RC6
RC7/RX/DT 26 RC7
RD0 19 RD0
RD1 20 RD1
RD2 21 RD2
RD3 22 RD3
RD4 27 RD4
RD5/P1B 28 RD5
RD6/P1C 29 RD6
RD7/P1D 30 RD7
RE0/AN5 8 RE0
RE1/AN6 9 RE1
RE2/AN7 10 RE2
PIC16F887
X1
2 1
CRYSTAL
C2
22pF
C1
22pF
LCD2
LM044L
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7
1 2 3 4 5 6 7 8 9 10 11 12 13 14
RA0 RA1 RA2
RD0 RD1 RD2 RD3 RD4 RD5 RD6 RD7
LCD2(VDD)
LCD2(VEE)
RB0
RV1(2)
RV1
RES-VAR
RB1
RV2(2)
RV2
RES-VAR
RC6 RXD
RC7 TXD
RTS
CTS