

Practica #1: Puertos de Entrada y Salida

1. Objetivos

- Familiarizarse con el compilador MPLAB X IDE y el simulador de circuitos Proteus.
- Manejar los puertos de un microcontrolador como entrada/salida.

2. Materiales

Computadora con Software MPLAB X IDE y Proteus

3. Introducción

Esta guía de laboratorio muestra cómo utilizar el compilador MPLABX para crear un proyecto en su entorno y simular su funcionamiento en Proteus. El proyecto a realizar será la configuración como entrada y salida de pines de los puertos del microcontrolador 18F4550.

3.1. Herramientas de Software

El MPLAB X es el IDE oficial de la empresa Microchip para poder trabajar sus diversos modelos de microcontroladores, usando diversos compiladores, los cuales contienen operadores estándar de C y librerías de funciones incorporadas, específicas para los registros de los microcontroladores, proporcionado así una herramienta para acceder a operaciones del hardware del microcontrolador.

El IDE lo podemos encontrar desde el sitio oficial de Microchip ([Link](#)) y el compilador que se usara es la versión XC8 ([Link](#)) para microcontroladores de 8 bits. El ambiente del MPLAB X IDE lo podemos ver en la **Fig. 1**

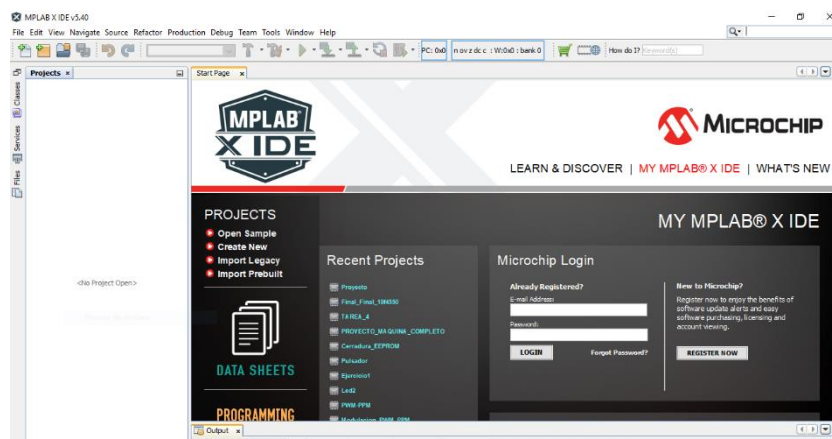


Fig. 1 Entorno de MPLAB X IDE

La amplia gama de la librería de dispositivos, además de otras herramientas virtuales, del simulador de circuitos Proteus hace de uso esencial para monitorear el funcionamiento de un microcontrolador, por lo que estaremos usando este simulador para las prácticas y el modelo de Microcontrolador PIC18F4550. El entorno de trabajo de Proteus se muestra en la Fig. 2.

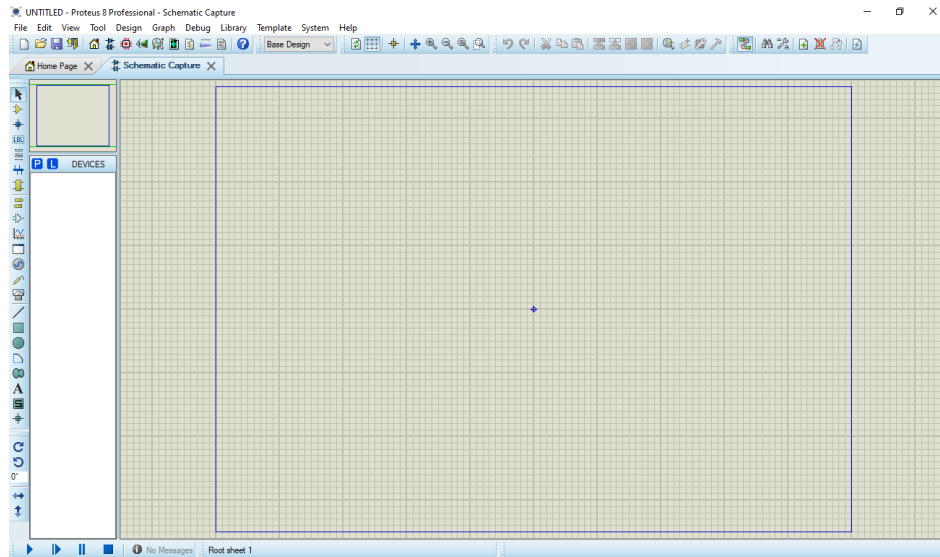


Fig. 2 Entorno de Simulador Proteus

Los modos a usar más frecuentes del Simulador Proteus serán:

- **Modo Componentes:** Selecciona los componentes a utilizar. Ver **Fig. 3**



Fig. 3 Modo Componentes

- **Modo Terminal:** Selecciona Alimentación (Power), Tierra (Ground) y terminales de entrada y salida (Input y Output, respectivamente). Por default, los componentes en Proteus funcionan omitiendo los pines de alimentación y tierra. Ver **Fig. 4**



Fig. 4 Modo Terminal

- **Modo Generador de Señales:** Permite añadir señales DC, Pulso configurable, flanco (Edge) de subida o caída, señal de reloj, patrones de señales digitales, etc. Ver **Fig. 5**



Fig. 5 Modo Generador de Señales

- **Instrumentos:** Dispone de los equipos de medición más utilizados (Osciloscopio, Voltímetro, Amperímetro, etc.) y herramientas propias de Proteus (I2C Debugger, SPI Debugger, etc.). Ver **Fig. 6**



Fig. 6 Modo de Instrumentos

3.2. Microcontroladores

Un Microcontrolador es una computadora a pequeña escala, con recursos limitados empleados para realizar una tarea determinada. Los Microcontroladores a diferencia de los Microprocesadores son dispositivos cerrados, teniendo una arquitectura Harvard (Ver **Fig. 7**), en la cual la memoria de datos y la memoria de programa son completamente independientes, por lo tanto, pueden ser de diferentes tamaños y con su propio bus, lo cual permite al dispositivo el paralelismo de procesamiento de datos y la optimización de tiempo, logrando comunicarse con el mundo real mediante los pines de entrada y/o salida.



Fig. 7 Arquitectura Harvard

Al tener internamente todos los recursos y ser cerrados se podría decir que están limitados, sin embargo, esto se compensa debido a que hay diferentes empresas que se dedican a fabricarlos, cada una de ellas proponiendo diferentes modelos, por lo tanto, cuando se realiza una aplicación puede elegir el que más se ajuste a sus necesidades considerando el precio y los recursos que ofrece.

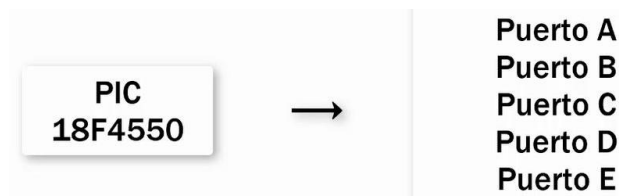
El Microcontrolador que utilizaremos es de la empresa Microchip (PIC18F4550). Los Microcontroladores de esta empresa se llaman “PIC’s” y dependiendo de los recursos que tengan pertenecerá a una familia en específico. El PIC18F4550 es ampliamente utilizado, y entre sus principales módulos y características tiene:

5 puertos con total de 35 pines.	ADC de 10 bits.
3 interrupciones externas.	2 módulos CCP.
4 módulos de timer.	Módulo EUSART.
Módulo MSSP.	Comunicación USB

3.3. Registros del Microcontrolador

Dentro de los Microcontroladores PIC tenemos tres registros asociados a los puertos: el registro TRIS, el registro PORT y el registro LAT.

Cada uno de estos registros va a estar asociado para cada puerto del Microcontrolador, por ejemplo, para este caso en el que estamos utilizando el PIC18F4550, vamos a tener 5 puertos de entrada y/o salida: los puertos A, B, C, D y E.



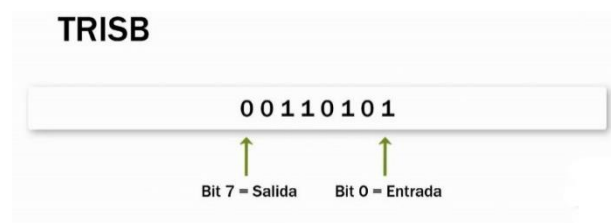
Por lo tanto, existirá cada uno de estos registros para cada puerto.



3.3.1. Registro TRIS.

Es el registro de dirección de datos. En él, vamos a configurar si el puerto se comportará como una entrada o una salida, por ejemplo: si queremos colocar LEDs en algún puerto, entonces ese puerto tendría que ser de salida, ya que el Microcontrolador va a enviar información al mundo exterior. Si quisiéramos ver el estado de un Push Button, ese puerto tendría que ser de entrada, ya que el Microcontrolador va a recibir información del mundo exterior. Entonces, a cada puerto se le asocia un registro TRIS, puede ser: TRISA, TRISB, TRISC, TRISD o TRISE.

Si le colocamos a algún bit del registro TRISB un 0 lógico, es que ese bit del puerto B va a ser una salida; por lo contrario, si colocamos un 1 lógico quiere decir que ese bit del puerto B será una entrada.



3.3.2. Registro PORT.

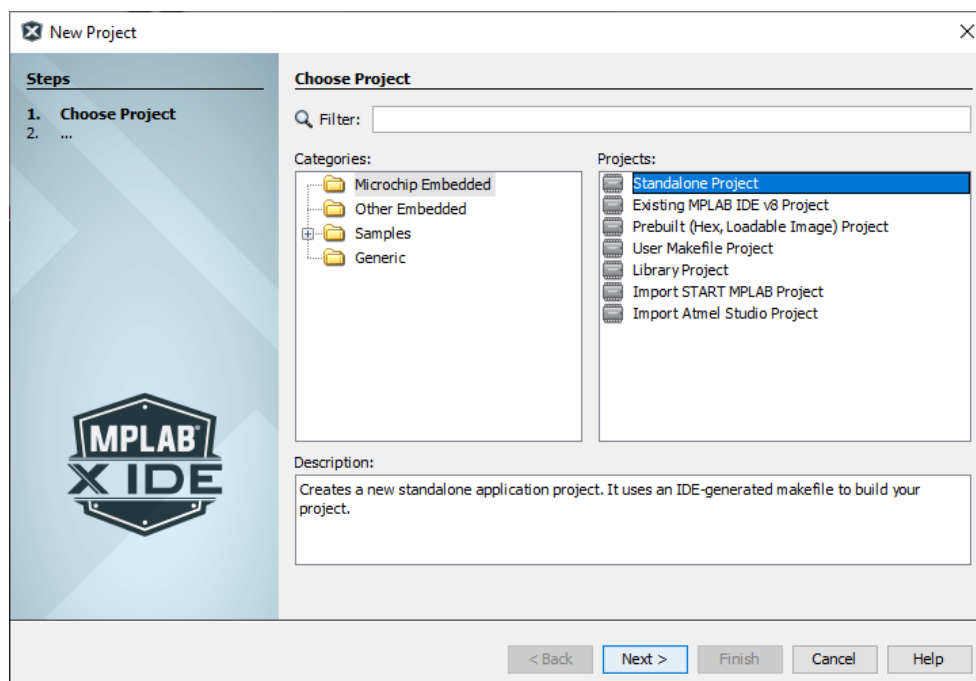
El segundo registro que utilizaremos es el registro PORT. El registro PORT sirve para leer los niveles de los pines en el dispositivo, por lo tanto, cuando queramos leer información del mundo exterior vamos a recurrir a leer la información que tiene el registro PORT. El registro PORT puede ser: PORTA, PORTB, PORTC, PORTD y PORTE.

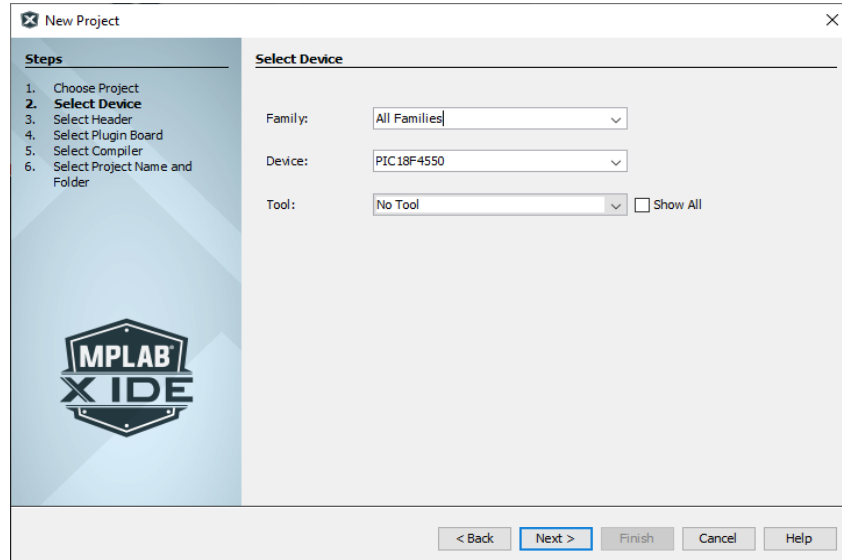
3.3.3. REGISTRO LAT

El registro LAT es un Latch de salida, por lo tanto, si queremos enviar información al mundo exterior, necesitaremos escribir en el registro LAT. El registro LAT puede ser: LATA, LATB, LATC, LATD y LATE.

4. Procedimiento de la Practica.

- a) Abrir el MPLAB X IDE y crear un nuevo proyecto. Los pasos se detallan en las siguientes figuras.





New Project

Steps

1. Choose Project
- 2. Select Device**
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder

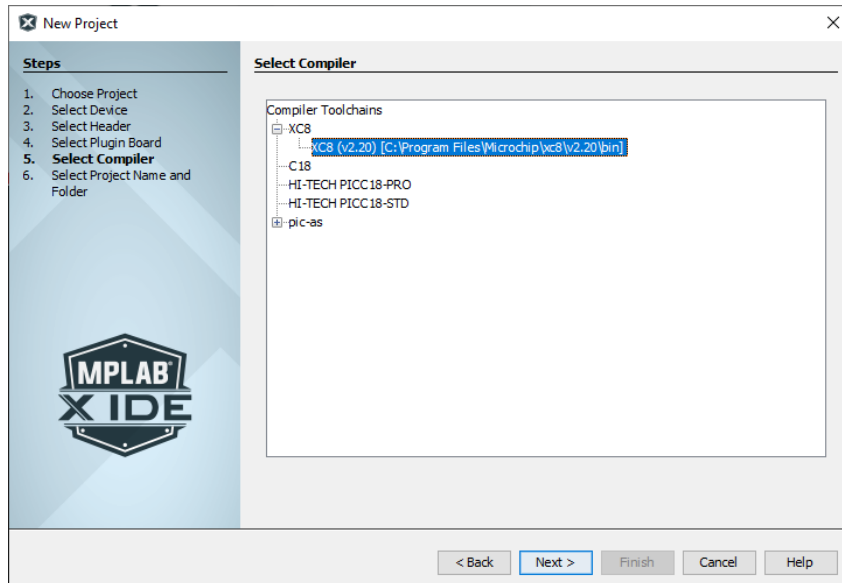
Select Device

Family:

Device:

Tool: ☐ Show All

< Back Next > Finish Cancel Help



New Project

Steps

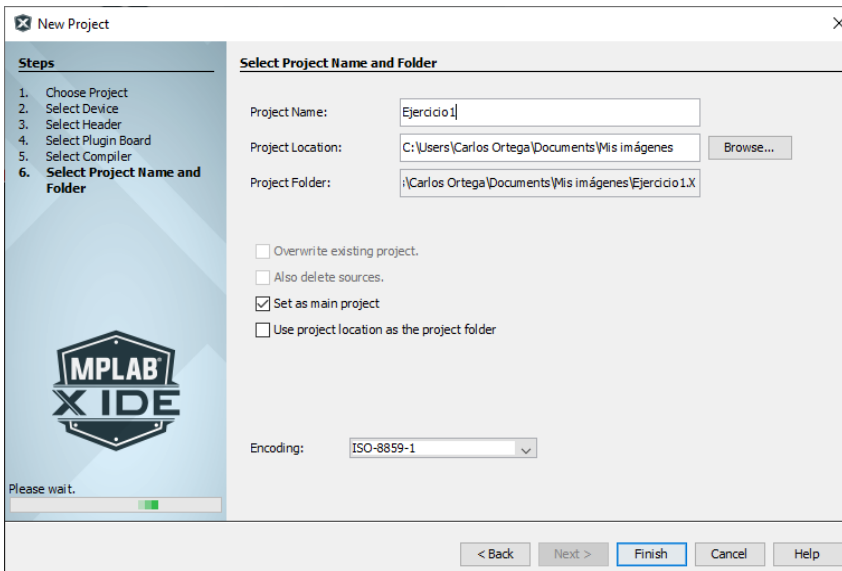
1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
- 5. Select Compiler**
6. Select Project Name and Folder

Select Compiler

Compiler Toolchains

- XC8
 - XC8 (v2.20) [C:\Program Files\Microchip\xc8\v2.20\bin]
- C18
 - HI-TECH PICC18-PRO
 - HI-TECH PICC18-STD
- pic-as

< Back Next > Finish Cancel Help



New Project

Steps

1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
- 6. Select Project Name and Folder**

Select Project Name and Folder

Project Name:

Project Location: Browse...

Project Folder:

☐ Overwrite existing project.

☐ Also delete sources.

☒ Set as main project

☐ Use project location as the project folder

Encoding:

Please wait.

< Back Next > Finish Cancel Help

4.1 Parte 1: Configuración de un pin como Salida.

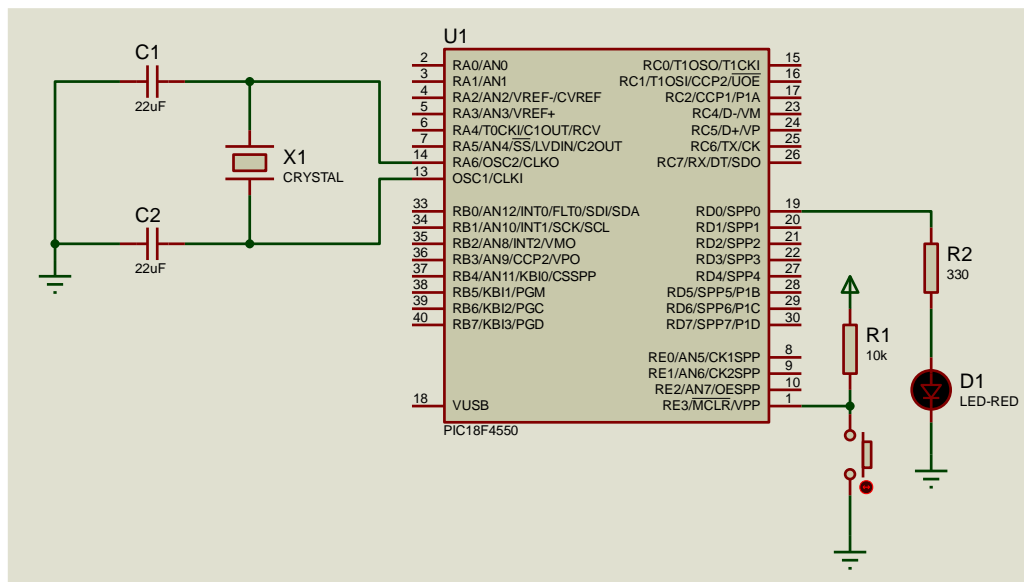
- b) Crear dentro del proyecto MPLAB, el archivo fuente con el nombre de **led**. El código se encargará de configurar un pin de un puerto D para encender un led y realizar un parpadeo. El archivo de **ConfigurationBits.h** se puede encontrar en el repositorio (Link) para ser agregado al proyecto.

```
#include <xc.h>
#include "ConfigurationBits.h"
#define _XTAL_FREQ 4000000
#include <PIC18F4550.h>

void initMain(){
    TRISDbits.TRISD0 = 0; //Configurar el pin D0 como Salida
    LATDbits.LATD0 = 0; //El valor de D0 vale 0
}

void main(void){
    initMain();
    while(1)
    {
        PORTDbits.RD0=1;
        __delay_ms(500);
        PORTDbits.RD0=0;
        __delay_ms(500);
    }
}
```

- c) Una vez creado los archivos, compilar el proyecto para generar el archivo .hex que utilizaremos para la simulación, siguiendo el diagrama de la Figura.



- d) El código anterior representa la ejecución del encendido de un led con dos tiempos: uno de encendido (500ms) y uno de apagado (500ms). Este código no es óptimo por ocupar dos tiempos, por lo tanto, se modificará para que solo ocupe un tiempo de retardo.

```
#include <xc.h>
#include "ConfigurationBits.h"
#define _XTAL_FREQ 4000000
#include <PIC18F4550.h>

void initMain(){
    TRISDbits.TRISD0 = 0; //Configurar el pin D0 como Salida
    LATDbits.LATD0 = 0; //El valor de D0 vale 0
}
void main(void) {
    initMain();
    while(1){
        LATDbits.LATD0 = ~LATDbits.LATD0;
        __delay_ms(500);
    }
    return;}

```

4.2 Parte 2: Configuración de un pin como Entrada.

- e) Crear otro proyecto con el nombre de **pulsador**, en el cual ahora configuraremos un pin como entrada agregando un pulsador que defina los dos estados del led.

```
#include <xc.h>
#include "ConfigurationBits.h"
#define _XTAL_FREQ 4000000
#include <PIC18F4550.h>

void initMain(){
    TRISCbits.RC0=1;//Esto configura como B0 como entrada
    TRISDbits.TRISD0 = 0; //Configurar el pin D0 como Salida
    LATDbits.LATD0 = 0; //El valor de D0 vale 0}
void main(void) {
    initMain();
    while(1){
        if(PORTCbits.RC0==1){
            PORTDbits.RD0=1;}
        else{
            PORTDbits.RD0=0;}}
    return;}

```


5. Actividades Propuestas.

Realizar la Simulación en Proteus de una secuencia de leds que ocupe todo el puerto D que se pueda cambiar con el estado del pulsador en el pin C1. La secuencia de los leds quedara de esta forma:

Si pulsador es igual a 0 los leds en el puerto D equivale a 01010101

Si pulsador es igual a 1 los leds en el puerto D equivale a 10101010