UNIVERSIDAD DE SONSONATE

Arquitectura de Computadoras

Guía 1 –Microcontroladores

Instructor: Ricardo González

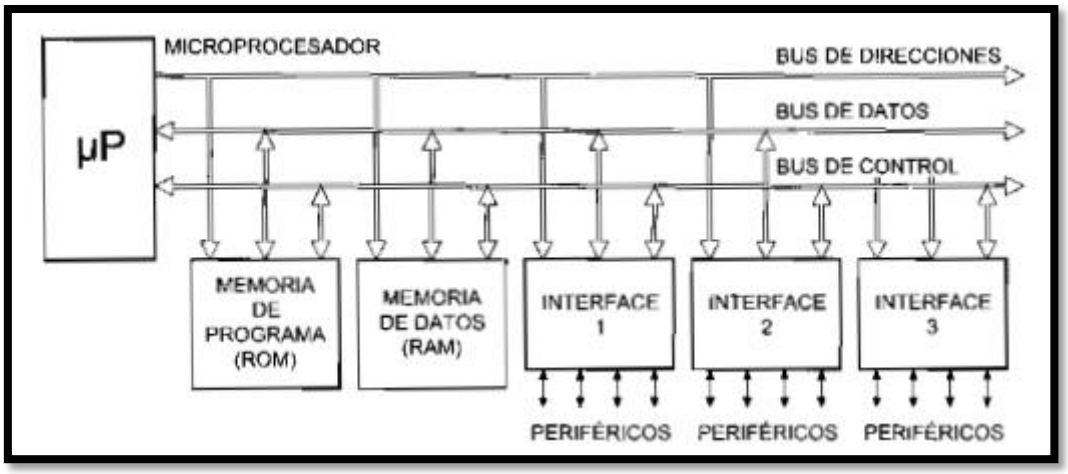
Facultad de Ingeniería y Ciencias Naturales

Contenido

¿Qué es un Microprocesadores y microcontrolador?

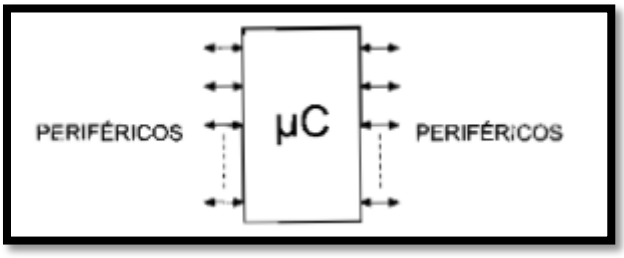
Microprocesador

Un microprocesador es básicamente un chip que contiene la CPU (***Central Processing Unit***) que se encarga de controlar todo el sistema. Un sistema digital basado en microprocesador es un sistema abierto ya que su configuración difiere según la aplicación a la que se destine. Se pueden acoplar los módulos necesarios para configurarlo con las características que se desee. Para ello saca al exterior las líneas de sus buses de datos, direcciones y control de modo que permita su conexión con la memoria y los módulos de entrada/salida. Finalmente resulta un sistema implementado por varios circuitos integrados dentro de una misma placa de circuito impreso.



Microcontrolador

Un microcontrolador es un sistema cerrado, lo que quiere decir que un solo circuito integrado se encierra en un sistema digital programable este dispositivo se destina a gobernar una sola tarea que no se puede modificar. Los microcontroladores disponen de los bloques esenciales: ***CPU, memorias de datos y de programas, reloj, periféricos de entrada/salida, etc.***



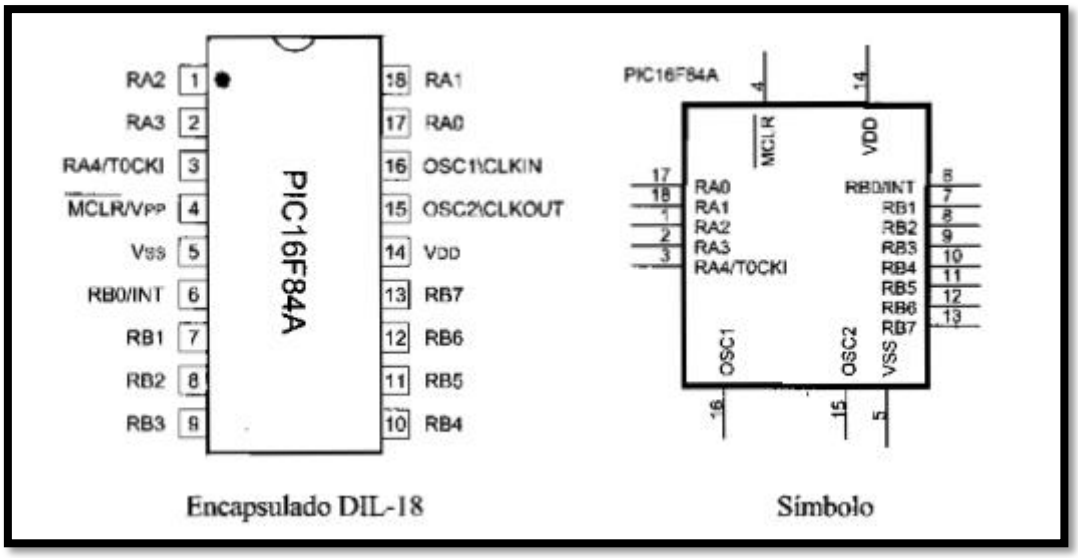
La diferencia fundamental entre ambos es que un sistema digital basado en un microcontrolador está formado por un solo circuito integrado lo que reduce notablemente el tamaño y el coste, mientras que un sistema basado en un microprocesador, al estar compuesto por varios circuitos integrados para soportar las memorias y los módulos de entrada/salida, tiene mayor tamaño, más coste y menor fiabilidad.

¿Qué es el PIC16F84A?

El PIC16F84A es un microcontrolador del cual podemos afirmar que es entonces un circuito integrado programable, y además, que contiene todo los componentes necesarios para controlar el funcionamiento de una tarea determinada por ejemplo:

- ✓ El control de una lavadora.
- ✓ El control de un teclado de ordenador.
- ✓ Una impresora.
- ✓ Un sistema de alarmas, etc.

Esto quiere decir que la tarea que determinemos que realice el microcontrolador, esta será la única tarea que desempañara. El PIC16F84A está encapsulado en un económico DIL de 18 pines. Debido a sus múltiples aplicaciones y facilidad de uso es uno de los microcontroladores más utilizados en la actualidad para la realización de proyectos sencillos. Su encapsulado y símbolo son los siguientes:



Componentes principales

El microcontrolador PIC16F84A puede trabajar con una frecuencia máxima de 10MHz, pero además existen otros elementos que intervienen en el trabajo de este los cuales son:

Alimentación

Normalmente los PIC16F84A se alimentan con 5 voltios aplicados en los pines ***Vdd y Vss*** que son respectivamente, la alimentación del chip.

Puertos de entrada/salida

El microcontrolador se comunica con el mundo exterior a través de los puertos. Estos están constituidos por líneas digitales de entrada/salida que trabajan entre 0 y 5v. Los puertos se pueden configurar como entradas para recibir datos o como salidas para gobernar dispositivos externos.

El PIC16F84A tiene dos puertos, los cuales son los siguientes:

1. El ***PUERTO A*** con 5 líneas, pines RA0 a RA4.
2. El ***PUERTO B*** con 8 líneas, pines RB0 a RA7.

Oscilador

Todo microcontrolador requiere de un circuito que le indiqué la velocidad de trabajo, es el llamado ***oscilador o reloj.*** Este genera una onda cuadrada de alta frecuencia que se utiliza como señal para sincronizar todas las operaciones del sistema.

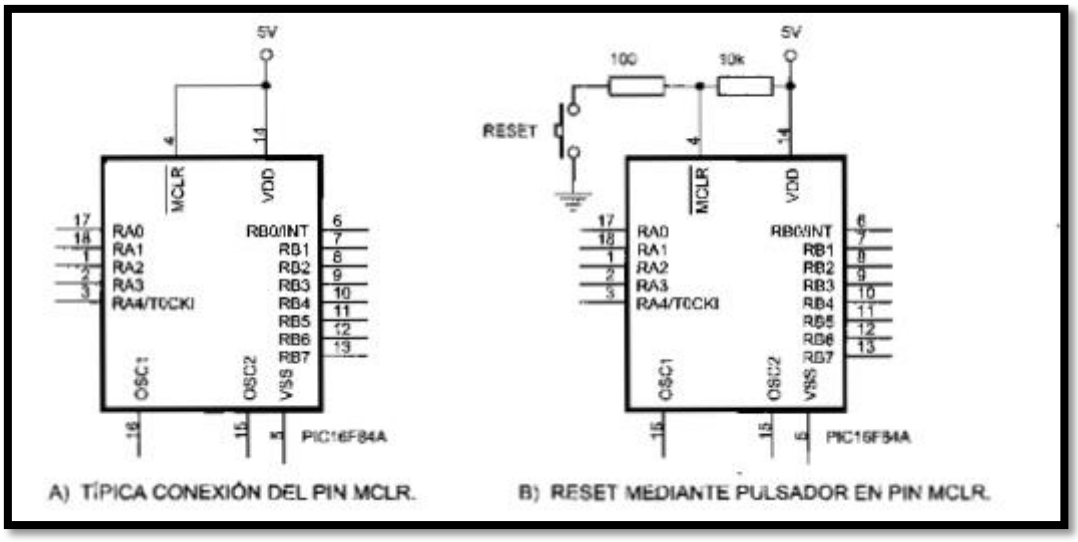
El PIC16F84A los pines ***OSC1/CLKIN y OSC2/CLKOUT*** son las líneas utilizadas para este fin. Permite cinco tipos de osciladores para definir la frecuencia de funcionamiento:

- ***XT:*** Cristal de Cuarzo.
- ***RC:*** Oscilador con resistencia y condensador.
- ***HS:*** Cristal de alta velocidad.
- ***LP:*** Cristal para baja frecuencia y bajo consumo de potencia.
- ***Externa:*** Cuando se aplica una señal de reloj externa.

Reset

El llamado reset en un microcontrolador provoca la reinicialización de su funcionamiento, un "*comienzo a funcionar a cero*". En este estado, la mayoría de los dispositivos internos del microcontrolador toman un estado conocido.

En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando es necesario. El pin de reset en los PIC se denomina ***MCLR(Master Clear)*** y produce un reset cuando se le aplica un nivel lógico bajo.



Arquitectura interna del PIC16F84

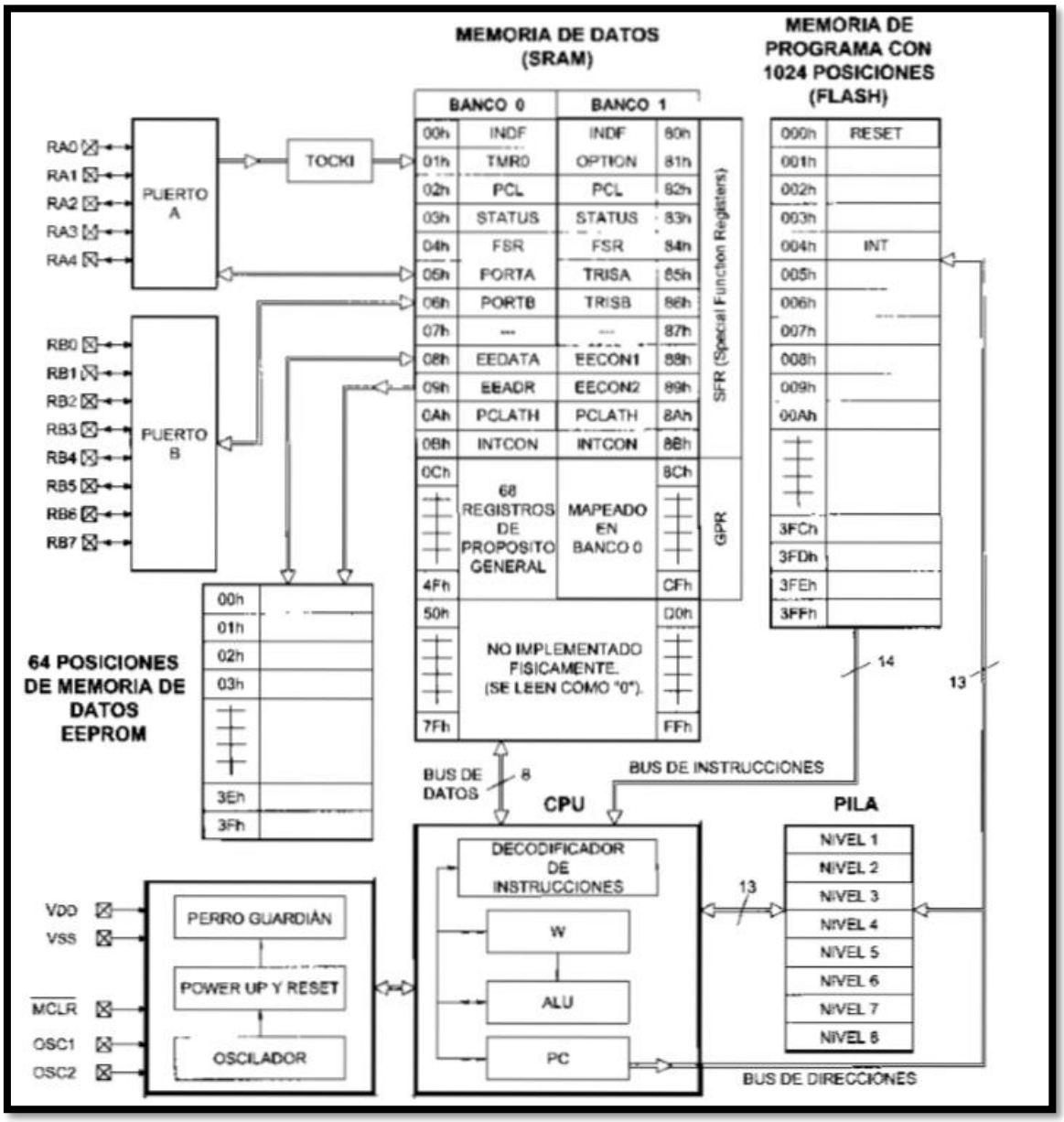
En los componentes principales que desatacan en el PIC16F84:

- Memoria de tipo ROM Flash de 1k x 14bits.
- Memoria de datos divida en 2 areas:

- Área RAM constituida por 22 registros de propósito específico (SFR) y 68 de propósito general.

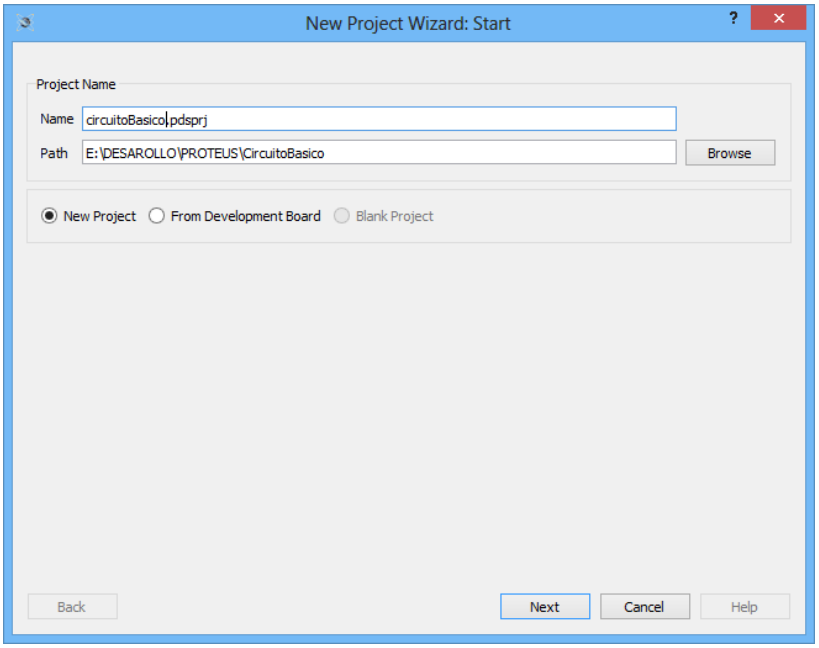
- ALU de 8 bits y registro de trabajo W, del que normalmente recibe un operando que puede ser cualquier registro, memoria, puerto de entrada/salida o el propio código de instrucción.
- Dos puertos para la comunicación con el exterior; PORTA de 5 bits <RA4:RA0> y PORTB de 8 bits <RB7:RB0>
- Contador de programa de 14 bits, lo que en teoría permitiría direccionar 4k de palabras de memoria, aunque el PIC16F84 solo dispone de 1k de memoria implementada.

La organización de la arquitectura del PIC16F84 es la siguiente:

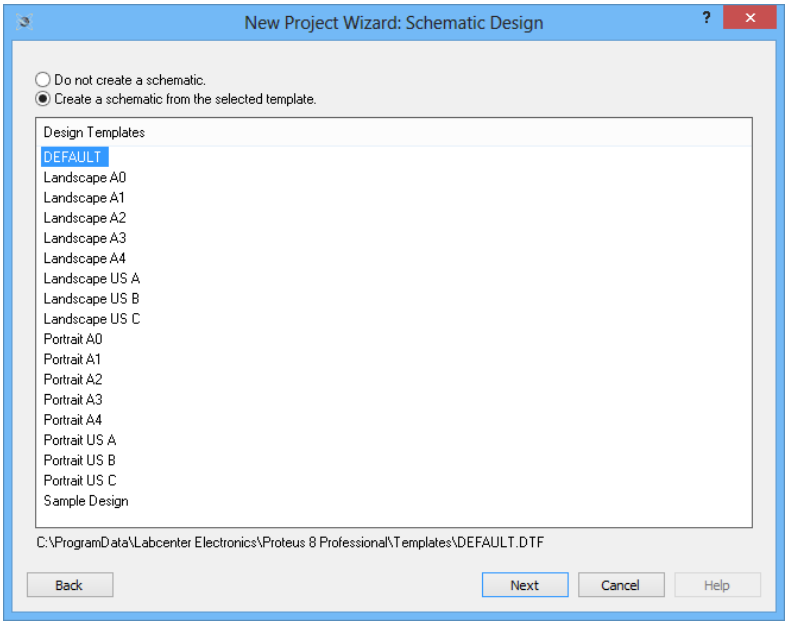


Diseño Básico de un circuito implementado el PIC16F84A.

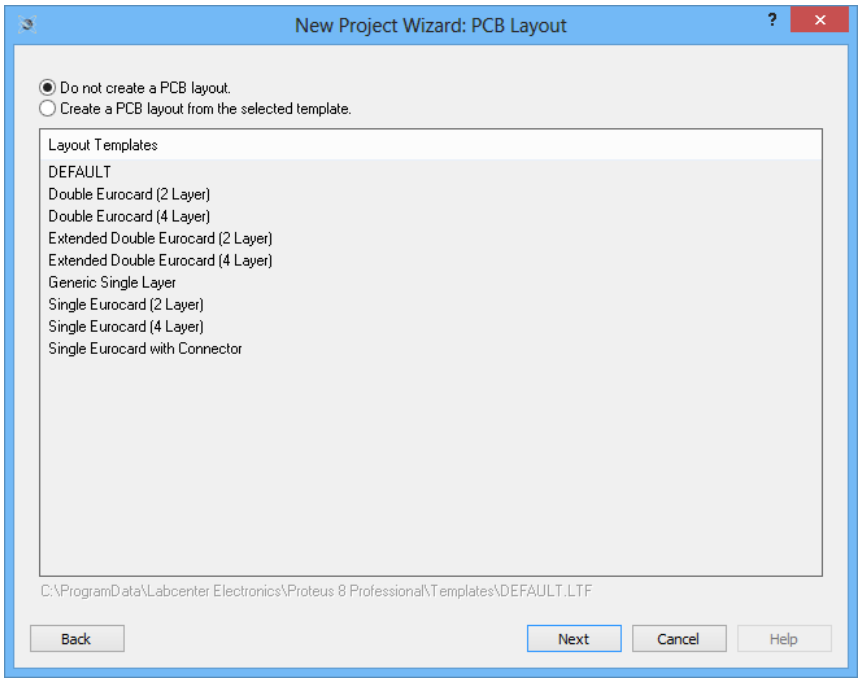
Crearemos el diseño básico de nuestro circuito implementando el PIC16F84A, para ello iniciamos el programa Proteus, y creamos un nuevo proyecto, que llevara el nombre de ***circuitoBasico***:



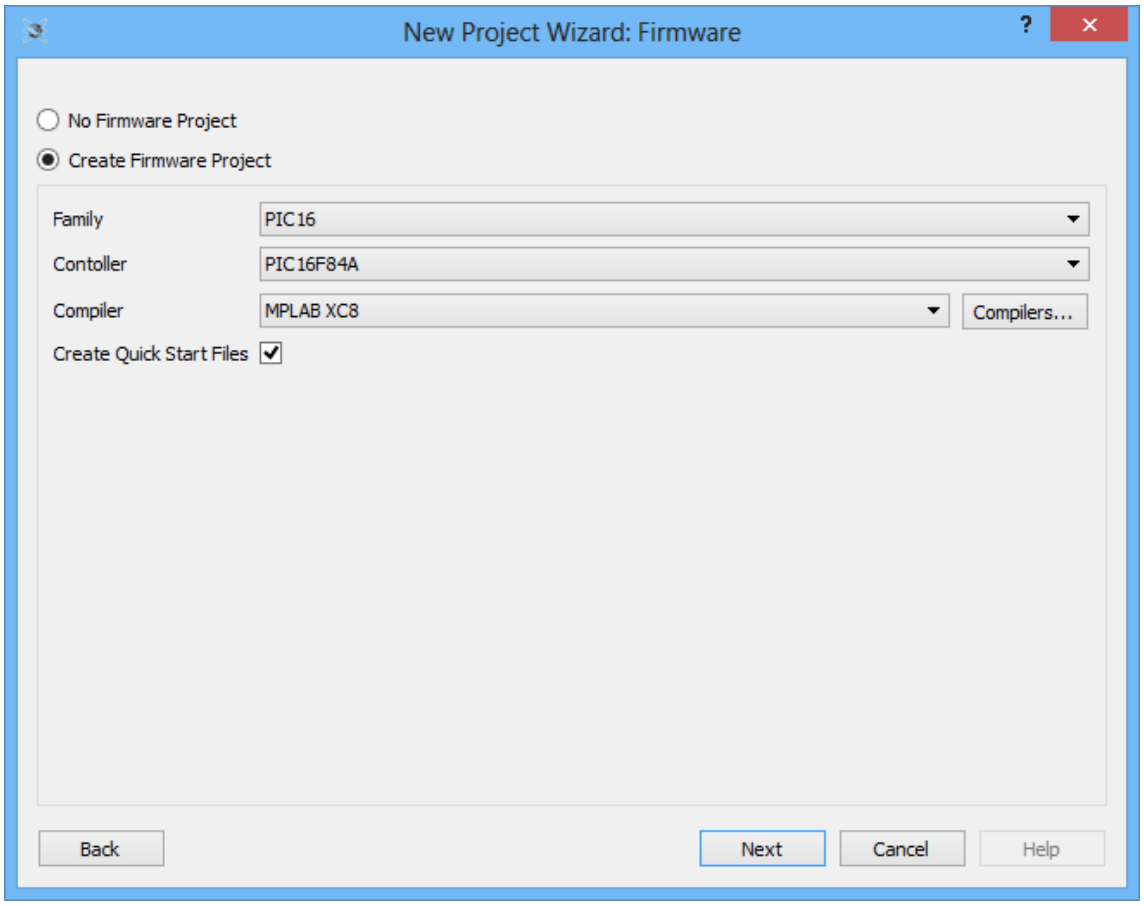
Luego seleccionamos la opción que nos permite crear un esquema desde un template, seleccionaremos la opción por ***DEFAULT.***



Luego al asistente le mencionamos que no necesitamos crear un PCB Layout.

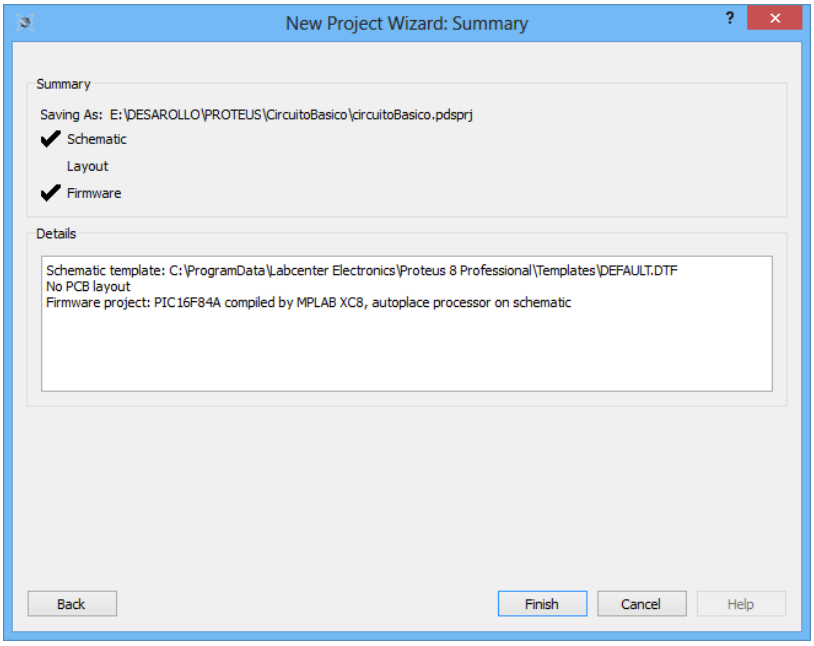


A continuación pasa a la opción de crear un firmware, que será nuestro aplicativo que se ejecutara en nuestro PIC16F84A, las opciones son las siguientes.



Nota: *Si en las opciones de compilador no aparece la opción* **"MPLAB XC8"**, *se le facilitara el instalador, para realizar la instalación del compilador.*

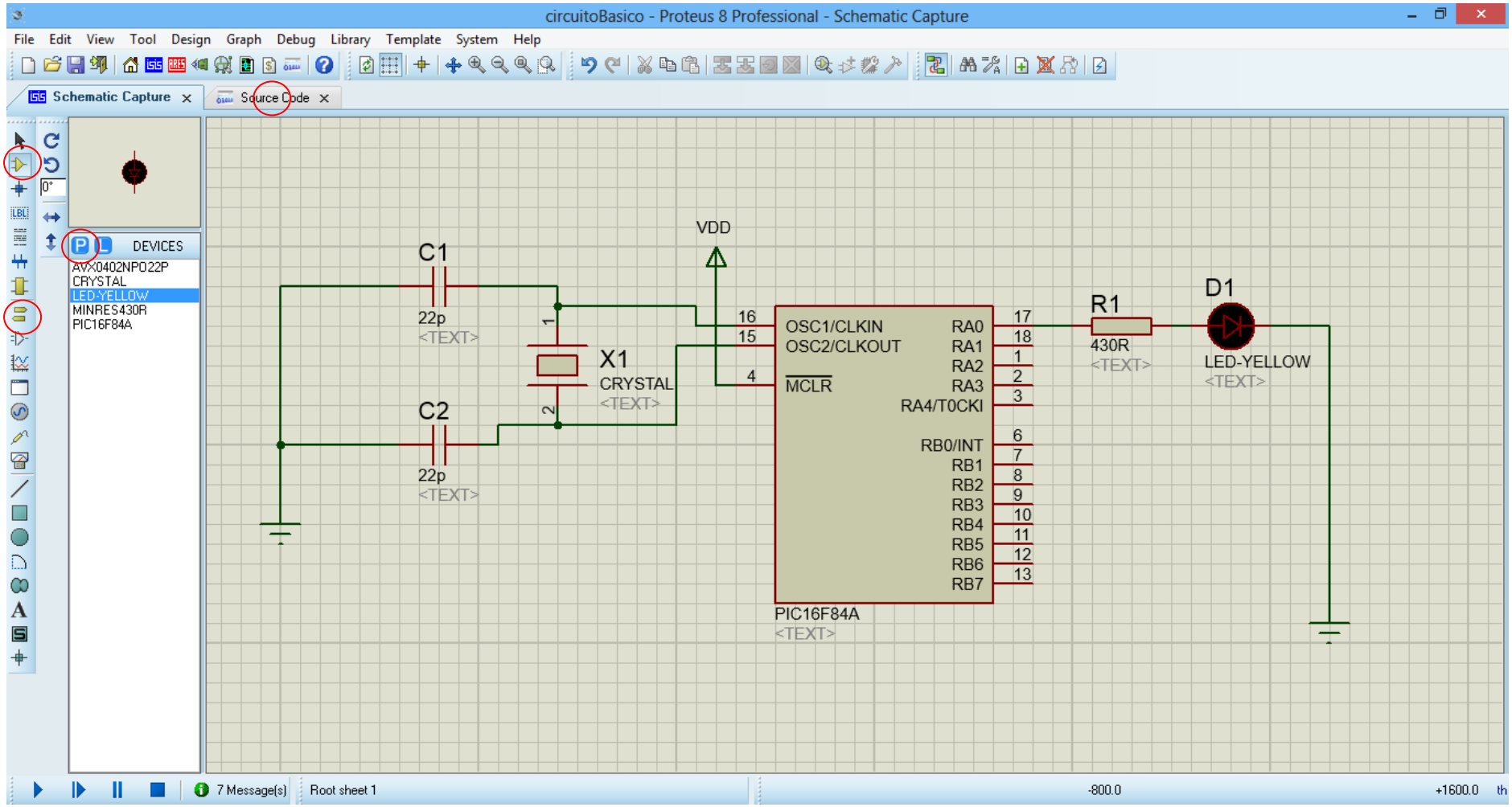
Con estas opciones finalizamos la configuración de nuestro proyecto.



Después de estos pasos se nos desplegara nuestra área de trabajo y dentro de ella visualizaremos el PIC16F84A. Ahora agregaremos los siguientes componentes:

- ✓ 2 Capacitores: ***AVX0402NP022P.***
- ✓ 1 Cristal de Cuarzo: ***CRYSTAL.***
- ✓ 1 Resistencia: ***MINRES430R.***
- ✓ 1 LED: ***LED-YELLOW.***

La resistencia y el led no forman parte esencial de los componentes del circuito básico de nuestro PIC16F84A, pero son elementos que utilizaremos para nuestro primer ejemplo, la colocación de los elementos se realizaran de la siguiente manera:


circuitoBasico - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
Source Code
DEVICES
AVX0402NP022P
CRYSTAL
LED-YELLOW
MINRES430R
PIC16F84A
VDD
C1
22p
<TEXT>
C2
22p
<TEXT>
X1
CRYSTAL
<TEXT>
OSC1/CLKIN
OSC2/CLKOUT
MCLR
RA0
RA1
RA2
RA3
RA4/T0CKI
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
PIC16F84A
<TEXT>
R1
430R
<TEXT>
D1
LED-YELLOW
<TEXT>
7 Message(s)
Root sheet 1
-800.0
+1600.0

Los círculos de color rojo que se encuentra en la figura, si los observamos de arriba hacia abajo, de izquierda a derecha, tendremos la siguiente descripción de ellos.

Elemento	Descripción
Component Mode	*Permite que busquemos los distintos componentes electrónicos, que se pueden utilizar para la creación de circuitos, mostrando además la lista de los componentes que ya hemos seleccionado.*
Terminals Mode	Permite que visualicemos las terminales más comunes para la creación de circuitos entre ella encontramos ***terminales a tierra, terminal de energía, etc.***
La letra "P"	Esta que se visualiza en la lista de los elementos que hemos seleccionado, nos permite buscar ya sean nuevos componentes o en el caso nuevas terminales que no se encuentren en la lista de selección
Source Code	Esta pestaña nos permitirá visualizar el código fuente de nuestro firmware, acorde al lenguaje que hemos selecciona (Para nuestro caso C).

Programación en C

Para programar la tarea que ejecutara nuestro PIC16f84A, tendremos opciones con respecto al lenguaje que podemos utilizar, podemos utilizar:

- Ensamblador.
- C

Como nos dimos cuenta al inicio que creamos nuestro proyecto estamos utilizando el compilador ***MPLAB XC8***, el cual nos permite crear aplicaciones bajo el lenguaje de C, es de tener muy en cuenta que es necesario tener instalado el compilador correspondiente para programar en determinado lenguaje, por defecto la configuración está establecida para programar en lenguaje ensamblador (Lenguaje se estudiara en otra asignatura).

Estructura básica de un programa para PIC16F84A

Como ya tenemos definido nuestro lenguaje (***C***), en la pestaña de ***Source Code*** nos encontraremos con las siguientes secciones:

#include <xc.h>: *La palabra reservada* ***include ,*** *nos permite utilizar funciones que se encuentren definidos en el archivo de cabecera(.**h**) que se encuentra entre los signos* ***"<>",*** *podemos definir nuestro propios archivos de cabecera dentro de nuestro proyecto, para hacer referencia a ellos solamente cambiamos los signos a utilizar, en este caso serían la comillas.*

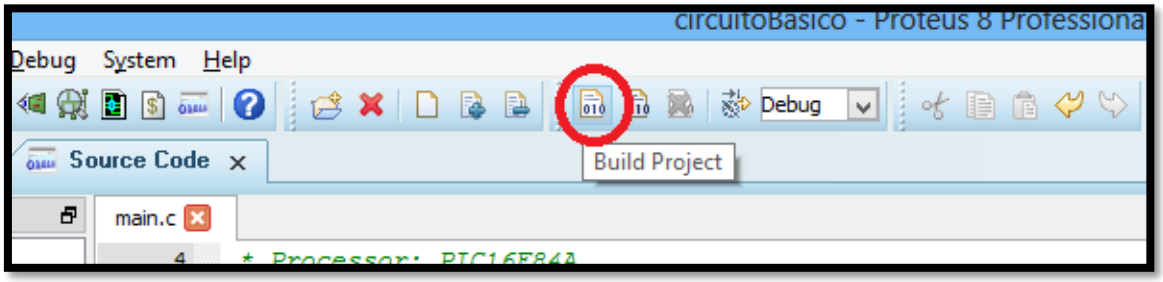
Void main(void): *Como toda aplicación creada en un lenguaje alto nivel es necesario definir una entrada principal donde se iniciara nuestra aplicación.*

***While(1)**: Recordemos que los microcontroladores están hechos para realizar una sola tarea específica, en este caso este ciclo while permite ejecutar dicha tarea durante todo el ciclo de vida del microcontrolador y siempre y cuando este se encuentre encendido.*

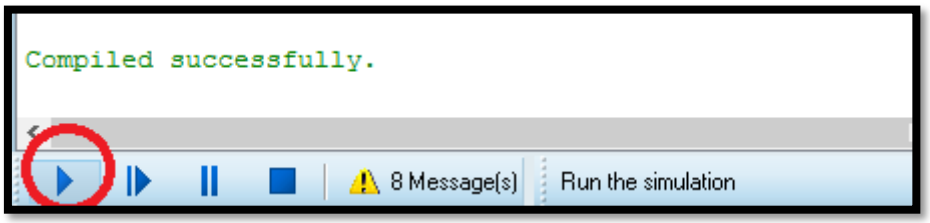
Ejemplo: Ya que tenemos definido nuestro circuito básico realizaremos la codificación de nuestro firmware, en este caso sería nuestro ***"Hola Mundo"*** pero con un LED, el código de nuestro firmware será el siguiente:

```
main.c
 4   * Processor: PIC16F84A
 5   * Compiler:  MPLAB XC8
 6   */
 7
 8  #include <xc.h>
 9
10
11  void main(void)
12  {
13    //Establecemos los puertos que seran entrada y salida
14    //--------------------------------------
15     TRISA = 0;                          //Configuramos que el puerto sera salida OUT
16    //--------------------------------------
17
18    while (1){
19       //Determinamos que PIN del puerto recibira corriente.
20       PORTA = 0b00001;
21    };
22  }
```

Luego de codificar nuestro firmware, compilamos, siempre en la misma pestaña de ***source code,*** se encuentra el siguiente icono:



Si la compilación ha sido exitosa, y queremos visualizar el funcionamiento de nuestro firmware, presionamos el botón ***Run Simulation.***



Si todo funciona de forma correcta veremos como el LED que tenemos colocado se enciende. ¿Qué se ha realizado en el código? Lo que se ha realizado es lo siguiente:

- Al definir ***TRIS*** y a continuación la letra del puerto ***A o B*** y esto es igual al valor ***0***, el puerto es configurado como salida (OUT), si es igualado al valor de ***1,*** el puerto es configurado como entrada.
- El definir ***PORT*** y a continuación la letra del puerto ***A o B*** y esto lo igualamos a un valor binario, acorde a la cantidad de pines que tenga el puerto, con valor de ***0*** indicamos que este se encuentra inactivo (Sin paso de corriente), si es igualado al valor de ***1*** indicamos que se encuentra activo (Con paso de corriente).

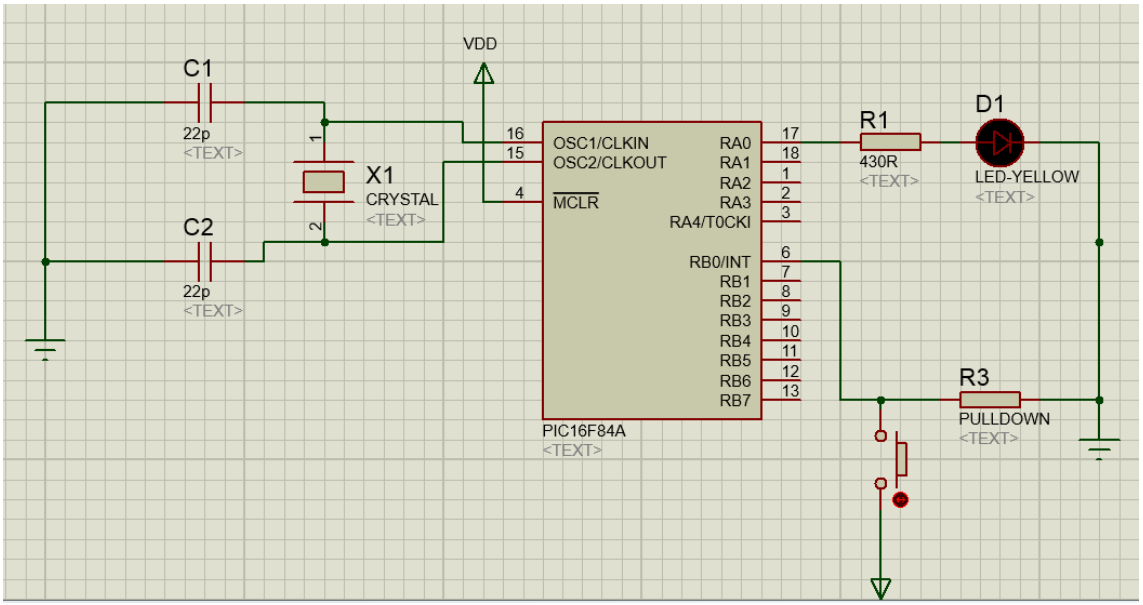
Cada uno de los puertos soportan un valor máximo de corriente, estos valores se definen en la siguiente tabla:

Modo (I/0)	Puerto A	Puerto B
Modo sumidero	80 mA	150 mA
Modo fuente	50 mA	100 mA

Ejemplo: En este ejemplo estableceremos el ***PUERTOB*** como entrada este quiere decir que tendrá un valor lógico de ***1***; nos apoyaremos siempre de nuestro esquema básico para simular nuestro circuito, y crearemos un nuevo proyecto llamado ***entradaPin,*** a este nuevo proyecto agregaremos los siguientes elementos

- ✓ 1 Boton: ***Button.***
- ✓ 1 Resistencia para un interruptor Pull-Down: ***PULLDOWN.***

Nuestro circuito deberá visualizarse de la siguiente manera:



Nuestro código fuente será el siguiente:

main.c

```
11  void main(void)
12  {
13    //Establecemos los puertos que seran entrada y salida
14    //----------------------------------------
15     TRISA = 0;                              //Configuramos que el puerto sera salida OUT
16     TRISB = 1;                              //Configuramos que el puerto sera entrada IN
17    //----------------------------------------
18
19    while (1){
20       //Determinamos el estado logico del PIN 0 del PUERTO B
21       //-----------------------------------------------------
22       if(PORTBbits.RB0){
23          //Determinamos que PIN del puerto recibira corriente.
24          PORTA = 0b00001;
25       }else{
26          PORTA = 0b00000;
27       }
28       //-----------------------------------------------------
29    };
30  }
```

Al momento de compilar y ejecutar la simulación observaremos que el LED se encenderá solamente que tengamos presionado el botón, el cual permite el paso de corriente.

EJERCICIOS.

Para cada uno de los ítems se tiene que realizar en distintos proyectos utilizando el PIC16F84A.

1. Realizar un circuito donde los puertos del PIC16F84A, se encuentre configurados como salida, y cada uno de los pines de los puertos debe conectar LED, haciendo que cada uno de ellos se encienda uno por uno, haciendo un efecto de intermitente con los LEDS.
2. Realizar un nuevo proyecto con el PIC16F84A donde en cada pin del PIC se colocaran LED, pero en este ejercicio es necesario que se encienda todos los LED del PUERTOA y luego del PUERTOB, realizando el mismo efecto que el anterior.
3. Realizar un nuevo proyecto donde se configure el PUERTOA como entrada, y cada uno de los pines del PUERTOA puedan encender un LED que se encuentre conectado al mismo número de pin del cual se presiona. Esto quiere decir que se utilizara un botón si presiono el botón del PIN0 del PUERTOA debe de encenderse el PIN0 del PUERTOB.