

**UNIVERSIDAD CATOLICA ANDRES BELLO**  
**FACULTAD DE INGENIERIA**  
**ESCUELA INFORMATICA**  
**CATEDRA DE ARQUITECTURA DEL COMPUTADOR**

***PRACTICA #7***

***Microcontroladores PIC 16F84 (Manejo del puerto como salida).***

Objetivos:

- 1.- Aprender en uso del software de programación y simulación.
- 2.- Familiarizarse con el software que permite leer y descargar programas en el microcontrolador PIC.
- 3.- Familiarizarse con las instrucciones básicas del microcontrolador.

Materiales:

- 1 Protoboard.
- 1 Fuente de poder
- 1 Multímetro Digital
- 1 PIC16F84
- 8 Diodos LED
- 8 Resistencia de 150Ω, ¼ W
- 1 Resistencias de 1K
- 1 Pulsadores

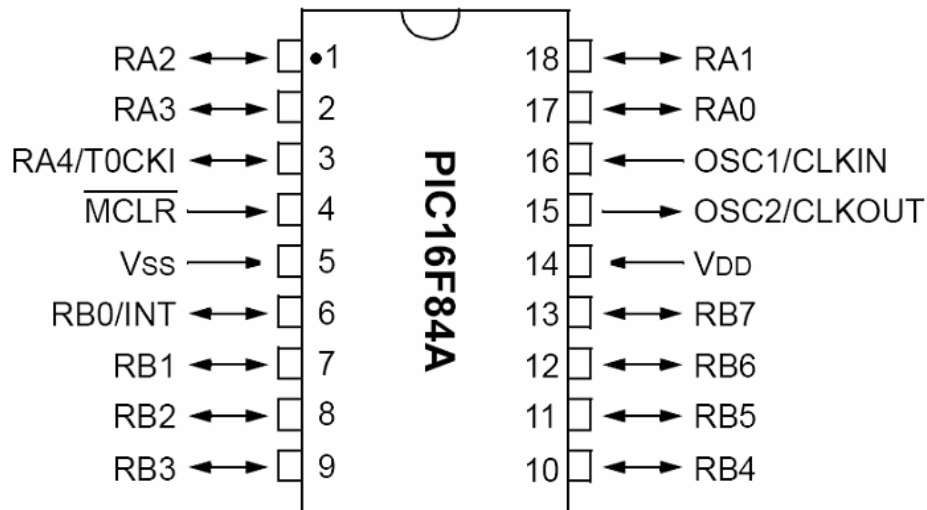
El PIC usa un juego de instrucciones tipo RISC, cuyo número puede variar desde 35 para PICs de gama baja a 70 para los de gama alta. Las instrucciones se clasifican entre las que realizan operaciones entre el acumulador y una constante, entre el acumulador y una posición de memoria, instrucciones de condicionamiento y de salto/retorno, implementación de interrupciones y una para pasar a modo de bajo consumo llamada *sleep*.

La arquitectura del PIC es sumamente minimalista. Esta caracterizada por las siguientes prestaciones:

- Área de código y de datos separadas (Arquitectura Harvard).
- Un reducido número de instrucciones de largo fijo.
- La mayoría de las instrucciones se ejecutan en un solo ciclo de ejecución (4 ciclos de clock), con ciclos de único retraso en las bifurcaciones y saltos.
- Un solo acumulador (W), cuyo uso (como operador de origen) es implícito (no está especificado en la instrucción).
- Todas las posiciones de la RAM funcionan como registros de origen y/o de destino de operaciones matemáticas y otras funciones.<sup>1</sup>
- Una pila de hardware para almacenar instrucciones de regreso de funciones.
- Una relativamente pequeña cantidad de espacio de datos direccionable (típicamente, 256 bytes), extensible a través de manipulación de bancos de memoria.
- El espacio de datos está relacionado con el CPU, puertos, y los registros de los periféricos.
- El contador de programa esta también relacionado dentro del espacio de datos, y es posible escribir en él (permitiendo saltos indirectos).

A diferencia de la mayoría de otros CPU, no hay distinción entre los espacios de memoria y los espacios de registros, ya que la RAM cumple ambas funciones, y esta es normalmente referida como "archivo de registros" o simplemente, registros.

### **DIAGRAMA DE PINES PIC 16F84A**



### **Características:**

- 1 Procesador tipo Risc
- 2 1 k de memoria FLASH
- 3 64 Bytes de data en EEPROM
- 4 68 Bytes de data en RAM
- 5 Set de 35 instrucciones
- 6 Alimentación de 5V
- 7 Frecuencia máxima de 20 Mhz.

Este Microcontrolador cuenta con 2 puertos, que son:

- 1 El puerto A con 5 líneas, pines RA0 a RA4.
- 2 El puerto B con 8 líneas, pines RB0 a RB7.

Cada línea puede ser configurada como entrada o salida, independiente unas de otras, según como se programe.

Al entregar 5 volts por Vdd, cada pin de la línea A debe entregar 25 mA, y en total de línea no debe pasar los 80 mA. En cuanto a la línea B cada pin debe entregar como máximo 25 mA y la línea no debe sobrepasar los 150 mA.

El Pic cuenta con un oscilador o reloj interno, para lectura o inserción de un nuevo reloj se cuenta con los pines OSC1/CLKIN y OSC2/CLKOUT.

El pin MCLR es el que permite reiniciar el sistema.

## SET DE INSTRUCCIONES

Instrucciones orientadas a registros						
MNEMONÍCO OPERANDOS		DESCRIPCIÓN	CÓDIGO OP	BANDERAS	NCIC	NOTAS
<a href="#">ADDWF</a>	f,d	<b>w + f → d</b>	00 0111 dfff ffff	C, DC, Z	1	1,2
<a href="#">ANDWF</a>	f,d	<b>w AND f → d</b>	00 0101 dfff ffff	Z	1	1,2
<a href="#">CLRF</a>	f	00 h → <b>f</b>	00 0001 1fff ffff	Z	1	2
<a href="#">CLRWF</a>	-	00 h → <b>w</b>	00 0001 0xxx xxxx	Z	1	-
<a href="#">COMF</a>	f,d	Complemento de <b>f → d</b>	00 1001 dfff ffff	Z	1	1,2
<a href="#">DECf</a>	f,d	<b>f - 1 → d</b>	00 0011 dfff ffff	Z	1	1,2
<a href="#">DECFSZ</a>	f,d	<b>f - 1 → d</b> (si es 0 salta)	00 1011 dfff ffff	Ninguna	1(2)	1,2,3
<a href="#">INCF</a>	f,d	<b>f + 1 → d</b>	00 1010 dfff ffff	Z	1	1,2
<a href="#">INCFSZ</a>	f,d	<b>f + 1 → d</b> (si es 0 salta)	00 1111 dfff ffff	Ninguna	1(2)	1,2,3
<a href="#">IORWF</a>	f,d	<b>w OR f → d</b>	00 0100 dfff ffff	Z	1	1,2
<a href="#">MOVf</a>	f,d	<b>f → d</b>	00 1000 dfff ffff	Z	1	1,2
<a href="#">MOVWF</a>	f	<b>w → f</b>	00 0000 1fff ffff	Ninguna	1	-
<a href="#">NOP</a>	-	No operación	00 0000 0xx0 0000	Ninguna	1	-
<a href="#">RLF</a>	f,d	Rota <b>f</b> izq por <b>carry → d</b>	00 1101 dfff ffff	C	1	1,2
<a href="#">RRF</a>	f,d	Rota <b>f</b> dcha por <b>carry → d</b>	00 1100 dfff ffff	C	1	1,2
<a href="#">SUBWF</a>	f,d	<b>f - w → d</b>	00 0010 dfff ffff	C,DC,Z	1	1,2
<a href="#">SWAPF</a>	f,d	Intercambia nibbles de <b>f → d</b>	00 1110 dfff ffff	Ninguna	1	1,2
<a href="#">XORWF</a>	f,d	<b>w XOR f → d</b>	00 0110 dfff ffff	Z	1	1,2

Instrucciones orientadas a bit						
MNEMÓNICO OPERANDOS		DESCRIPCIÓN	CÓDIGO OP	BANDERAS	NCIC	NOTAS
<a href="#">BCF</a>	f,b	Pone a 0 bit <b>b</b> de registro <b>f</b>	01 00bb bfff ffff	Ninguna	1	1,2
<a href="#">BSF</a>	f,b	Pone a 1 bit <b>b</b> de registro <b>f</b>	01 01bb bfff ffff	Ninguna	1	1,2
<a href="#">BTFSC</a>	f,b	Salto si bit <b>b</b> de reg. <b>f</b> es 0	01 10bb bfff ffff	Ninguna	1(2)	3
<a href="#">BTFSS</a>	f,b	Salto si bit <b>b</b> de reg. <b>f</b> es 1	01 11bb bfff ffff	Ninguna	1(2)	3

Instrucciones con literales y de control						
MNEMÓNICO OPERANDOS		DESCRIPCIÓN	CÓDIGO OP	BANDERAS	NCIC	NOTAS
<a href="#">ADDLW</a>	k	$w + k \rightarrow w$	11 111x kkkk kkkk	C,DC,Z	1	-
<a href="#">ANDLW</a>	k	$w \text{ AND } k \rightarrow w$	11 1001 kkkk kkkk	Z	1	-
<a href="#">CALL</a>	k	Llamada a subrutina <b>k</b>	10 0kkk kkkk kkkk	Ninguna	2	-
<a href="#">CLRWDT</a>	-	Borra temporizador del <b>WDT</b>	00 0000 0110 0100	TO,PD	1	-
<a href="#">GOTO</a>	k	Ir a dirección <b>k</b>	10 1kkk kkkk kkkk	Ninguna	2	-
<a href="#">IORLW</a>	k	$w \text{ OR } k \rightarrow w$	11 1000 kkkk kkkk	Z	1	-
<a href="#">MOVLW</a>	k	$k \rightarrow w$	11 00xx kkkk kkkk	Ninguna	1	-
<a href="#">RETFIE</a>	-	Retorno de una interrupción	00 0000 0000 1001	Ninguna	2	-
<a href="#">RETLW</a>	k	Retorno con <b>k</b> en <b>w</b>	11 01xx kkkk kkkk	Ninguna	2	-
<a href="#">RETURN</a>	-	Retorno de una subrutina	00 0000 0000 1000	Ninguna	2	-
<a href="#">SLEEP</a>	-	Modo Standby	00 0000 0110 0011	TO, PD	1	-
<a href="#">SUBLW</a>	k	$k - w \rightarrow w$	11 110x kkkk kkkk	C,DC,Z	1	-
<a href="#">XORLW</a>	k	$w \text{ XOR } k \rightarrow w$	11 1010 kkkk kkkk	Z	1	-

## DESARROLLO.

Para programar los PIC existen muchas formas de realizarlo, podemos usar el programa SIM84, que aunque es basado en MSDOS, es un programa que nos va guiando de manera muy fácil en la forma de hacer el programa, y también permite la simulación del mismo. En esta práctica se familiarizara con esta herramienta, y para la siguiente se utilizara otra herramienta conocida como SIM2000 y luego utilizara la herramienta MPLAB.

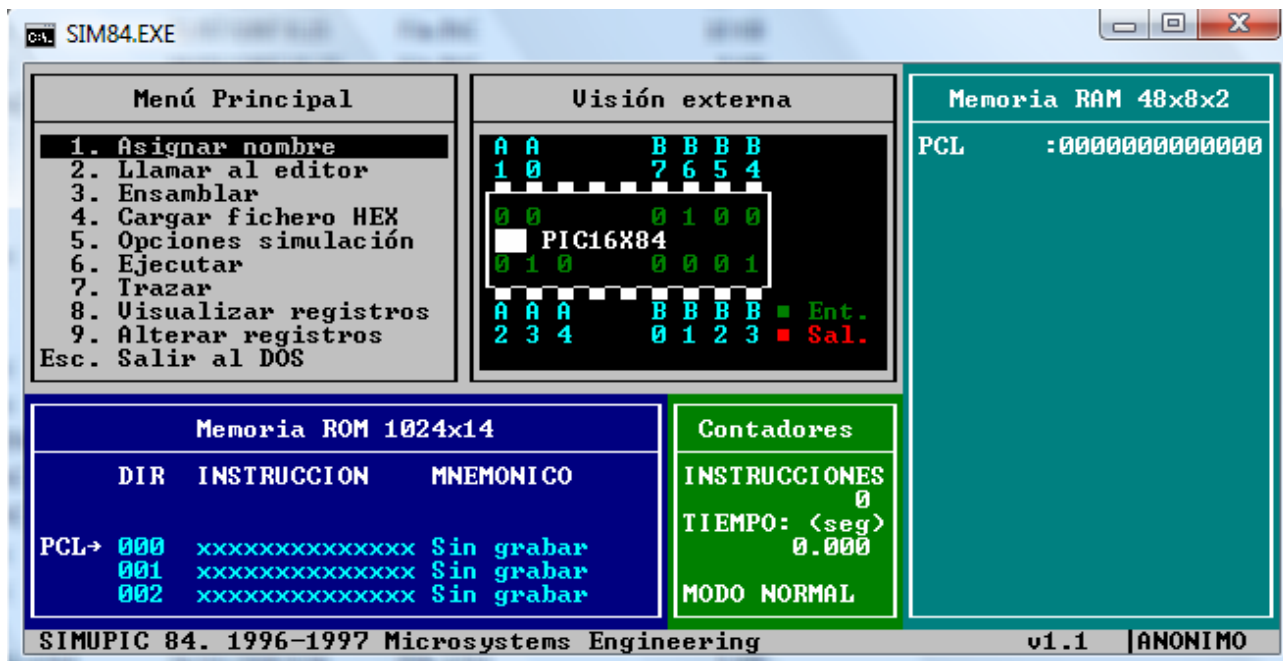
El programa que se quiere hacer en esta práctica es muy sencillo, sirve para aprender a usar el puerto del microcontrolador como salida (en este caso el puerto B). En este programa únicamente vamos a mostrar el valor de 10101010 en la salida del puerto PB, para lo cual hemos colocado una serie de led, que nos mostraran el valor que hay a la salida.

Para esta practica vamos ha utilizar un cristal de cuarzo a 4 MHz el cual utiliza también un par de condensadores de 22 pF.

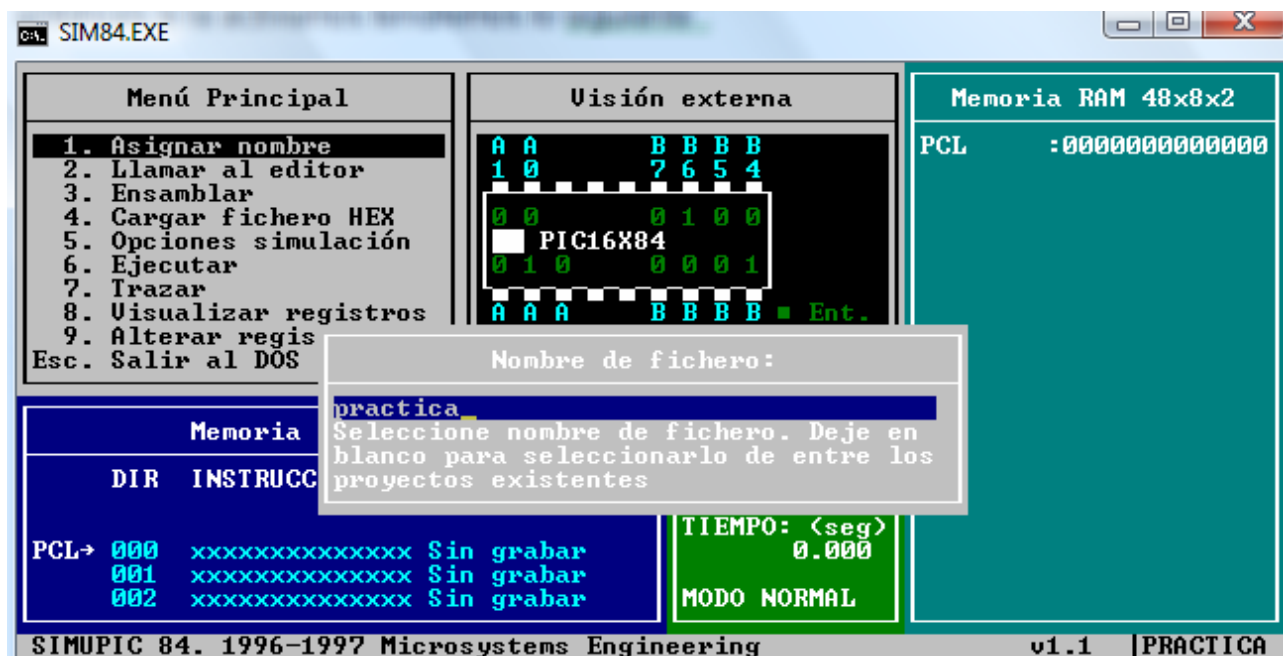
Una de las ventajas de usar SIM84 es que no ocupa muchos espacios y se puede ejecutar desde una memoria USB, con lo cual podrá utilizarlo en cualquier computador personal, sin tener que instalar aplicaciones o driver especiales.

Lo primero que debe hacer es cargar el programa ejecutando el archivo SIM84.EXE

Apenas lo ejecutemos podremos observar la siguiente pantalla:



Esta pantalla esta dividida en 5 cuadrantes principales, el primero es el de el menú principal, donde no va ha orientando de manera ordenada como debemos crear nuestro programa, como vemos la primera opción es la de Asignar Nombre, en esta le colocamos el nombre a nuestro programa (el mismo es recomendable que no exceda de 8 caracteres), entonces si la activamos tendremos lo siguiente:



Podemos llamar a nuestro programa práctica, al realizar esta operación se crea un archivo llamado practica.asm, que es en donde escribiremos nuestro programa.

Luego aplicamos sobre la opción numero dos que es la de editar el programa, donde vemos que se nos abre el editor. El programa que debemos tipear es el siguiente

```
LIST P=16F84

PUERTOB EQU 06H      ; Declaración del puerto B en la dirección 06 H
TRISB    EQU 06H      ; Declaración del el registro de configuración del puerto B
STATUS   EQU 03H      ; Declaración del registro de estado

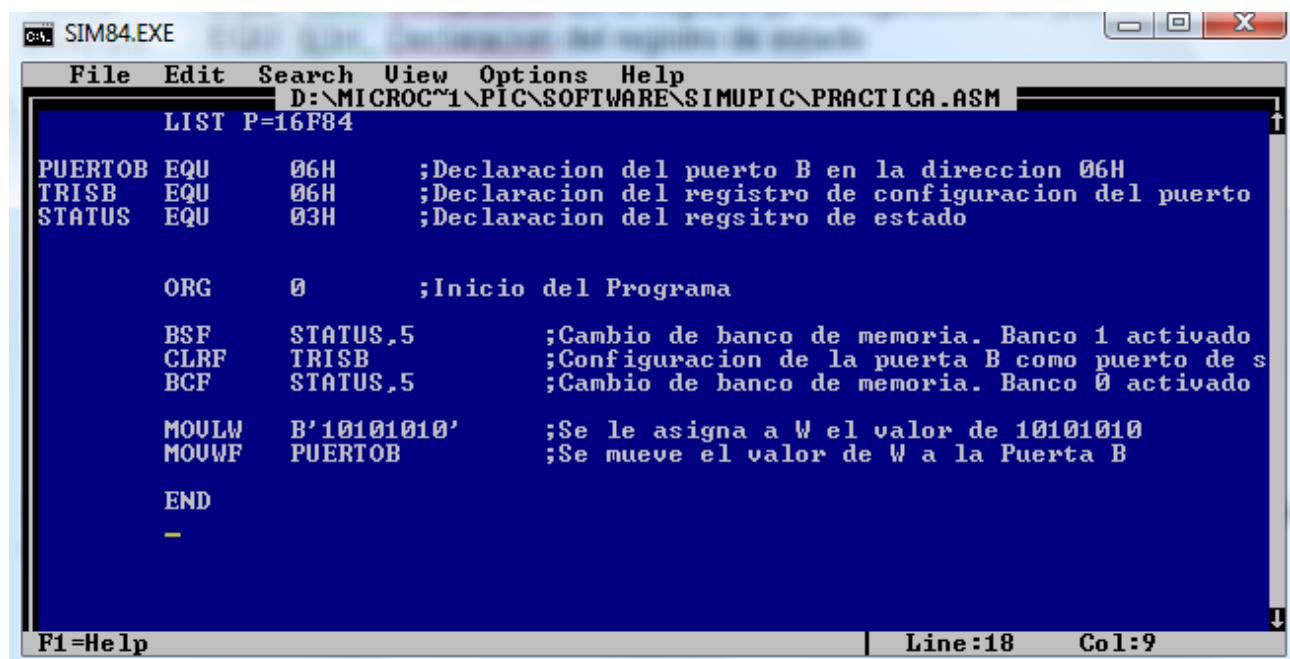
ORG 0                ; Inicio del Programa

BSF STATUS,5         ; Cambio del banco de memoria. Banco 1 activado.
CLRF TRISB           ; Configuración de la puerta B como puerto de salida.
BCF STATUS,5         ; Cambio del banco de memoria. Banco 0 activado

MOVLW B'10101010'    ; Se le asigna a W el valor de 10101010
MOVWF PUERTOB        ; Se mueve en valor de W a la Puerta B

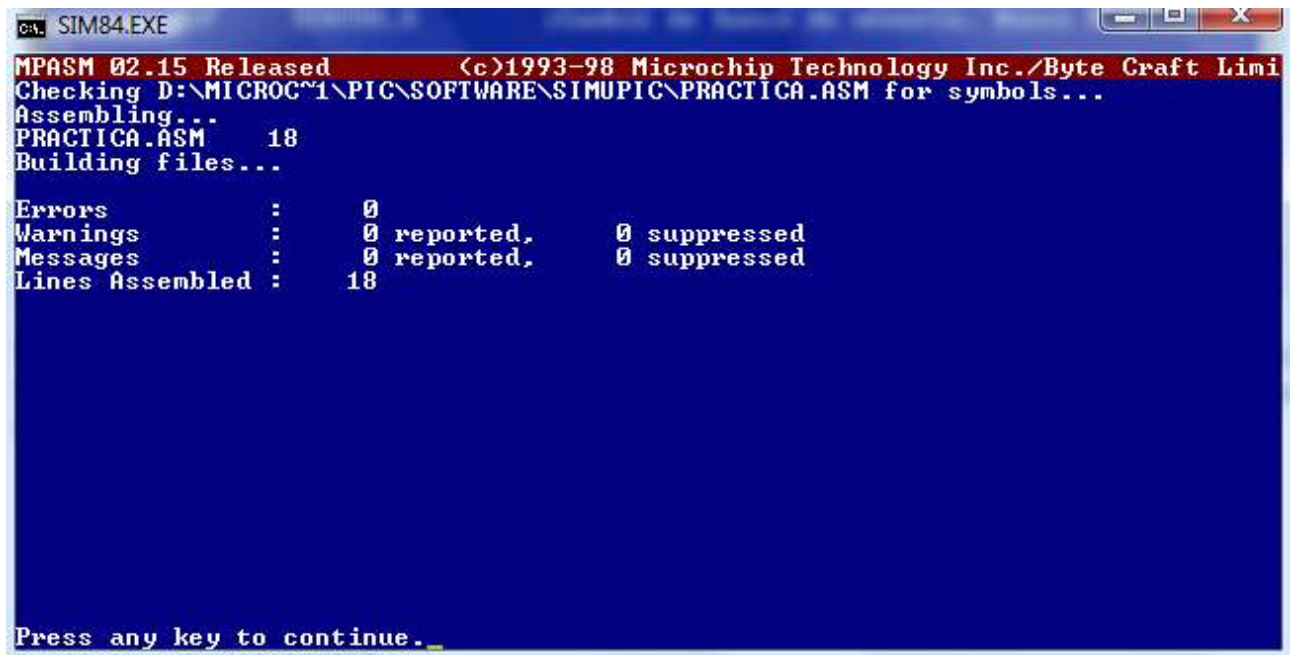
END
```

Es importante respetar que las declaraciones de variable y etiquetas se hacen sobre la columna 1, y el programa debe realizarse en otra columna distinta a la 1.



The screenshot shows a window titled "SIM84.EXE" with a menu bar (File, Edit, Search, View, Options, Help) and a title bar indicating the file path "D:\MICROC~1\PIC\SOFTWARE\SIMUPIC\PRACTICA.ASM". The main text area contains the assembly code as shown in the previous block. The status bar at the bottom indicates "F1=Help", "Line:18", and "Col:9".

Una vez terminado esto guardamos el archivo y salimos para ejecutar el tercer paso que seria ensamblar nuestro programa, si presionamos la opción 3 se ejecutara el proceso de ensamblado y podremos observar la siguiente ventana:



```
SIM84.EXE
MPASM 02.15 Released      (c)1993-98 Microchip Technology Inc./Byte Craft Limi
Checking D:\MICROC~1\PIC\SOFTWARE\SIMUPIC\PRACTICA.ASM for symbols...
Assembling...
PRACTICA.ASM      18
Building files...

Errors       :      0
Warnings     :      0 reported,      0 suppressed
Messages     :      0 reported,      0 suppressed
Lines Assembled :    18

Press any key to continue.
```

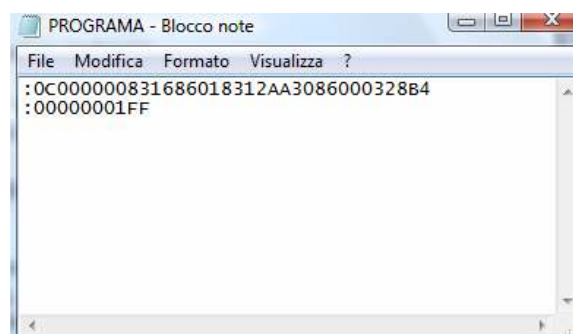
En caso de haber errores en el programa se nos indicara la cantidad de errores y apenas presionemos cualquier tecla se abrirá el archivos practica.ERR, donde se especifica el tipo de error y la línea donde esta.

Luego que se realizo este proceso se generan varios archivos:

- Practica.HEX: Es el archivo resultado del proceso de ensamblado, que es el que vamos a bajar al microcontrolador.
- Practica.ERR: Es el archivo donde se guarda el listado de errores y advertencias si da a lugar.
- Practica.LST: Es un archivo que especifica el resultado de proceso de ensamblado, dándonos ciertos valores como cantidad de variables, fecha y hora del ensamblado, cantidad de errores, y no muestra el código en HEX resultante entre otras cosas mas.

Estos archivos también los podemos editar usando el block de notas que trae Windows, lo cual nos permitirá otra manera de acceder a ellos. Aunque el único que debemos de trabajar es el .ASM, ya que si modificamos los otros lo mas seguro es que dañemos la aplicación que realizamos y cuando lo bajemos al microcontrolador este no funcione de la manera deseada por nosotros.

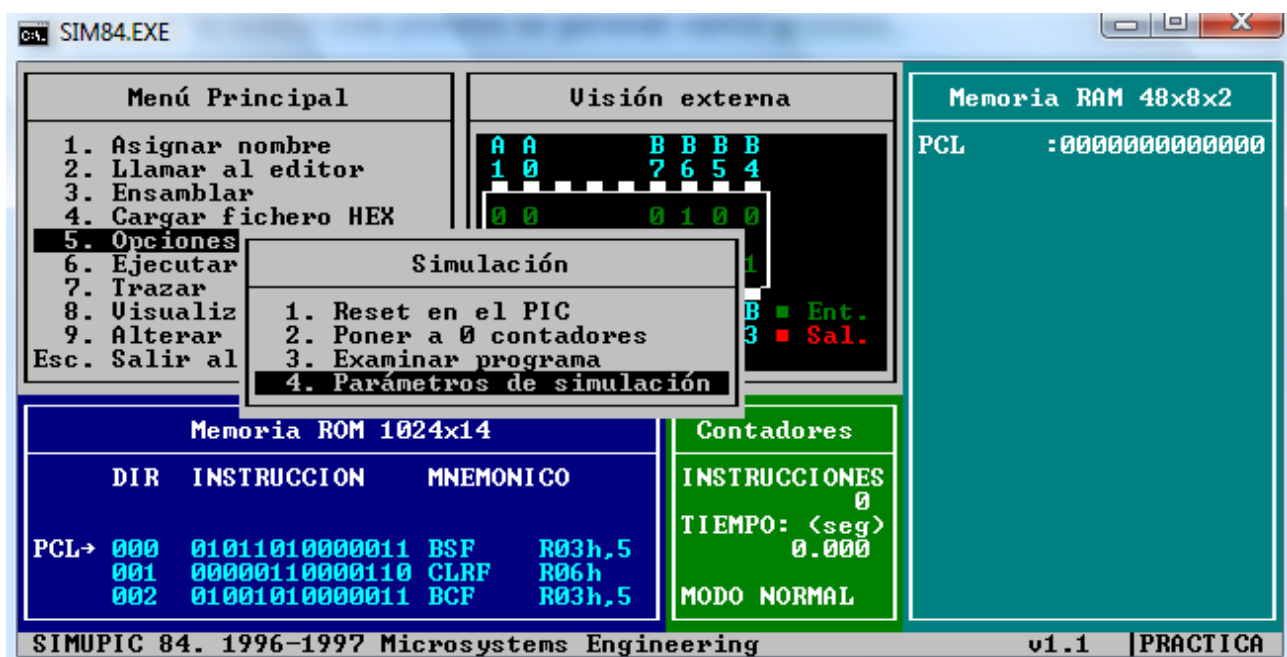
Si mandamos abrir el archivo .HEX, observaremos que estará representado en hexadecimal ya el programa listo a bajar el microcontrolador.



Este programa nos permite hacer una simulación de el funcionamiento del pic una vez bajemos el programa al IC. Para hacer la simulación debemos de cargar el programa en memoria, con la opción 4.



Luego debemos configurar nuestro simulador, y eso es en la opción 5 y luego la opción 4:



Inmediatamente se no preguntara con que tipo de IC estamos trabajando y es la versión CMOS o la versión FLASH, seleccionamos la versión FLASH. Para el tipo de oscilador decimos que será XT (cristal de cuarzo), asignamos la velocidad 4000 KHz y le decimos que vamos a deshabilitar al perro guardián, una vez listo esto nos pregunta si estamos de acuerdo y presionamos intro.

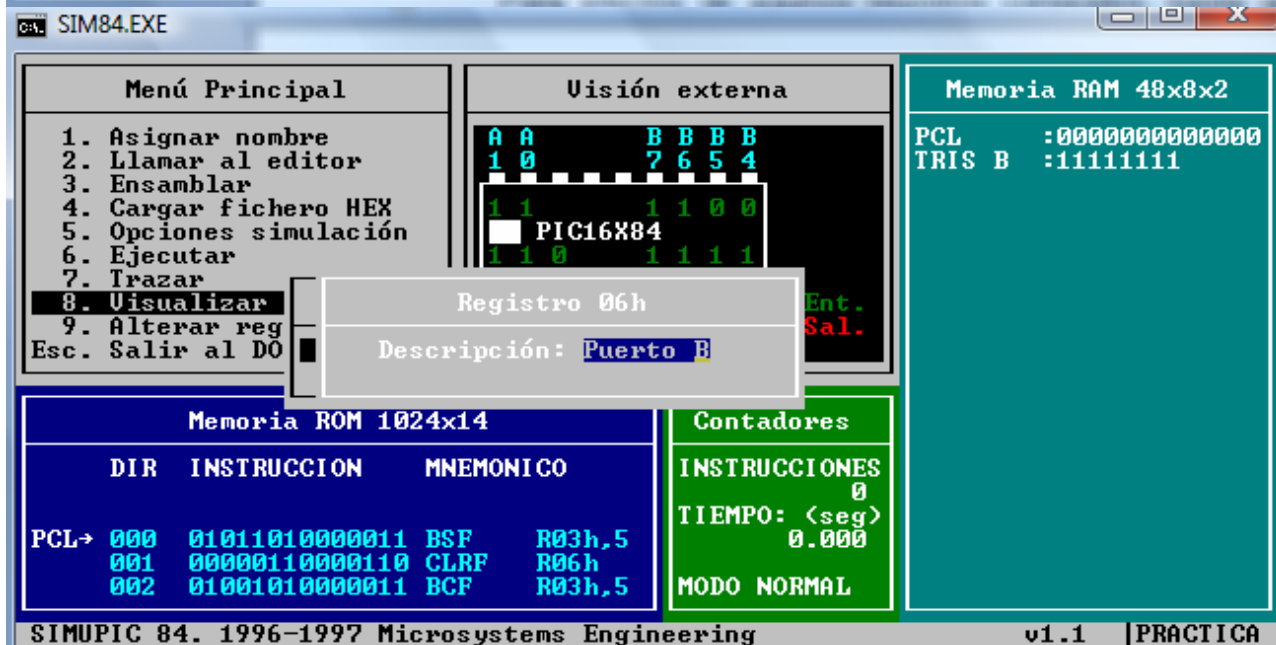
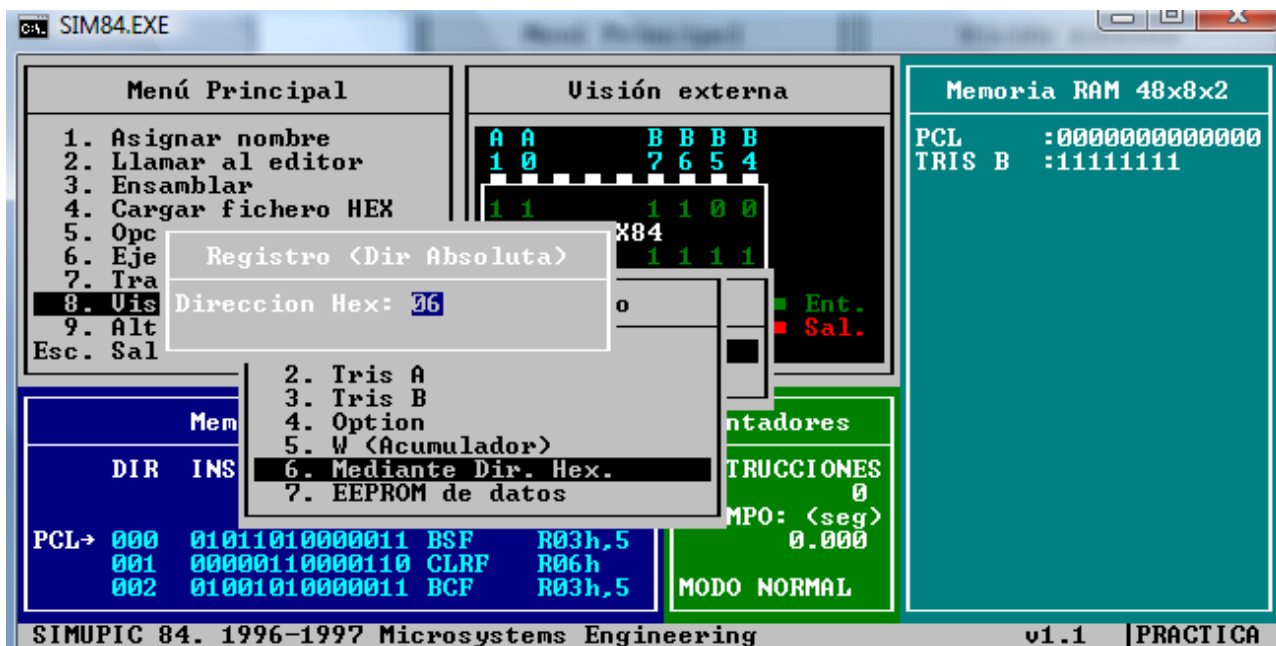




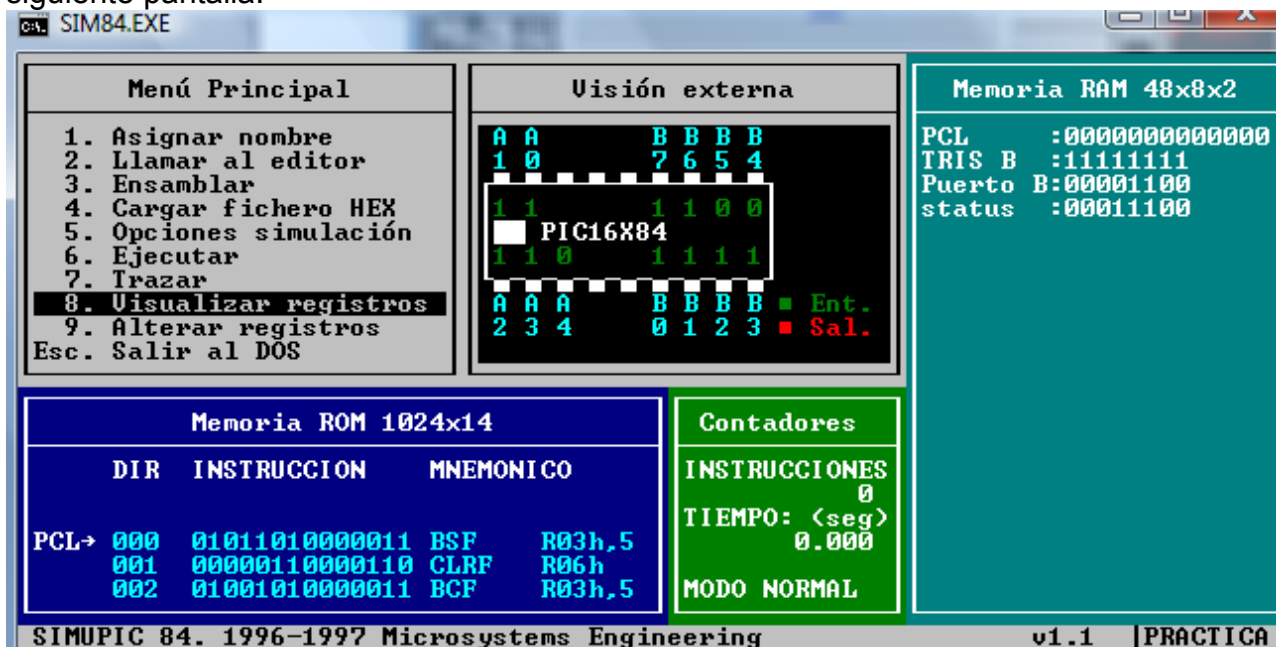
Luego que tenemos cargado el programa y configurado las características del mismo, entonces empezamos con la simulación, pero antes es bueno tener a la vista algunas variables de interés (STATUS, PB, TRISB). Primero seleccionamos la opción 8 en visualizar registros, y seleccionamos luego añadir registros.



Para efectos de agudos registros conocidos, basta que los seleccionemos y de inmediato aparecerán en el cuadro de memoria RAM. Si esa variable no esta en la lista debemos seleccionar la opción 6 (mediante dirección en hexadecimal) e indicarle la dirección de la variable, luego nos preguntara por el nombre que le daremos a la variables y terminado esto aparecerá en la pantalla de memoria.

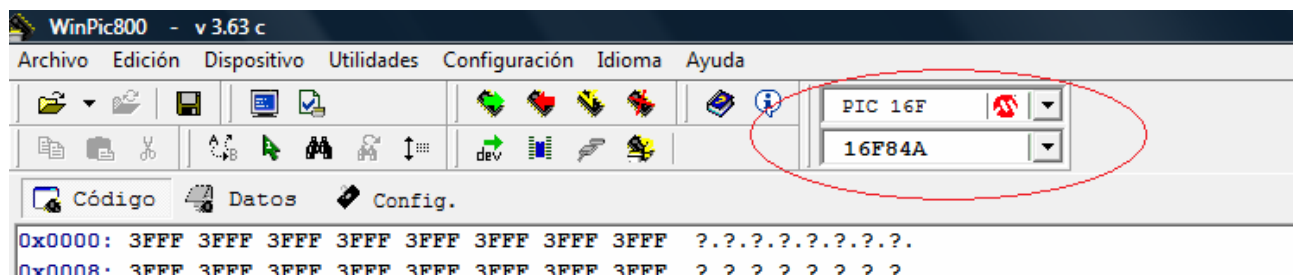


Repita los pasos anteriores y añada también al registro STATUS, y deberá tener la siguiente pantalla:

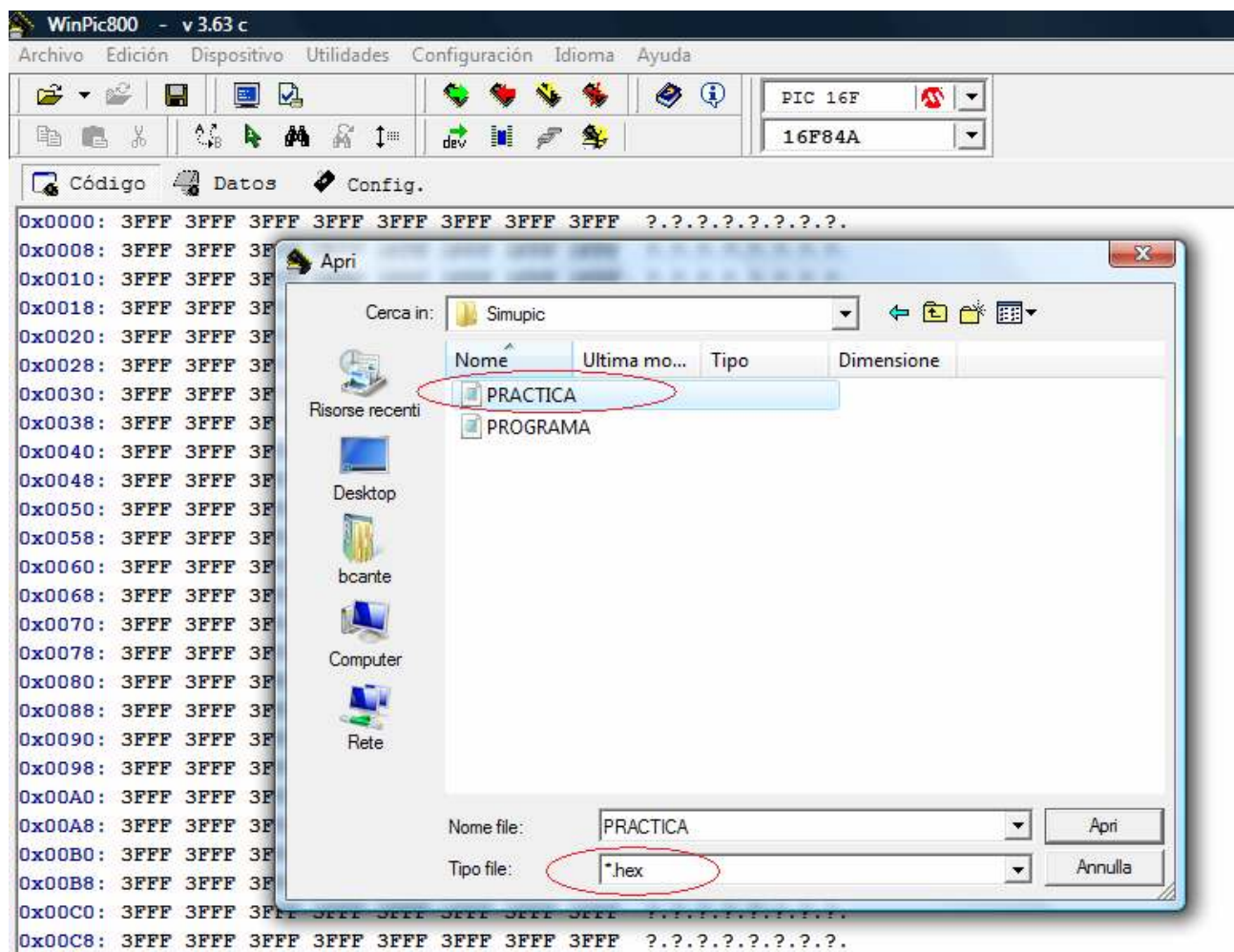


Una vez realizada esta operación podrá ejecutar el programa línea a línea, utilizando el la opción trazar, o de manera continua usando la opción ejecutar. Podrá apreciar los distintos cambios en las variables y la corrida del programa en los cuadros de memoria ROM y RAM, aparte de tener una visión ilustrada de los cambios de estados de los puertos de microcontrolador.

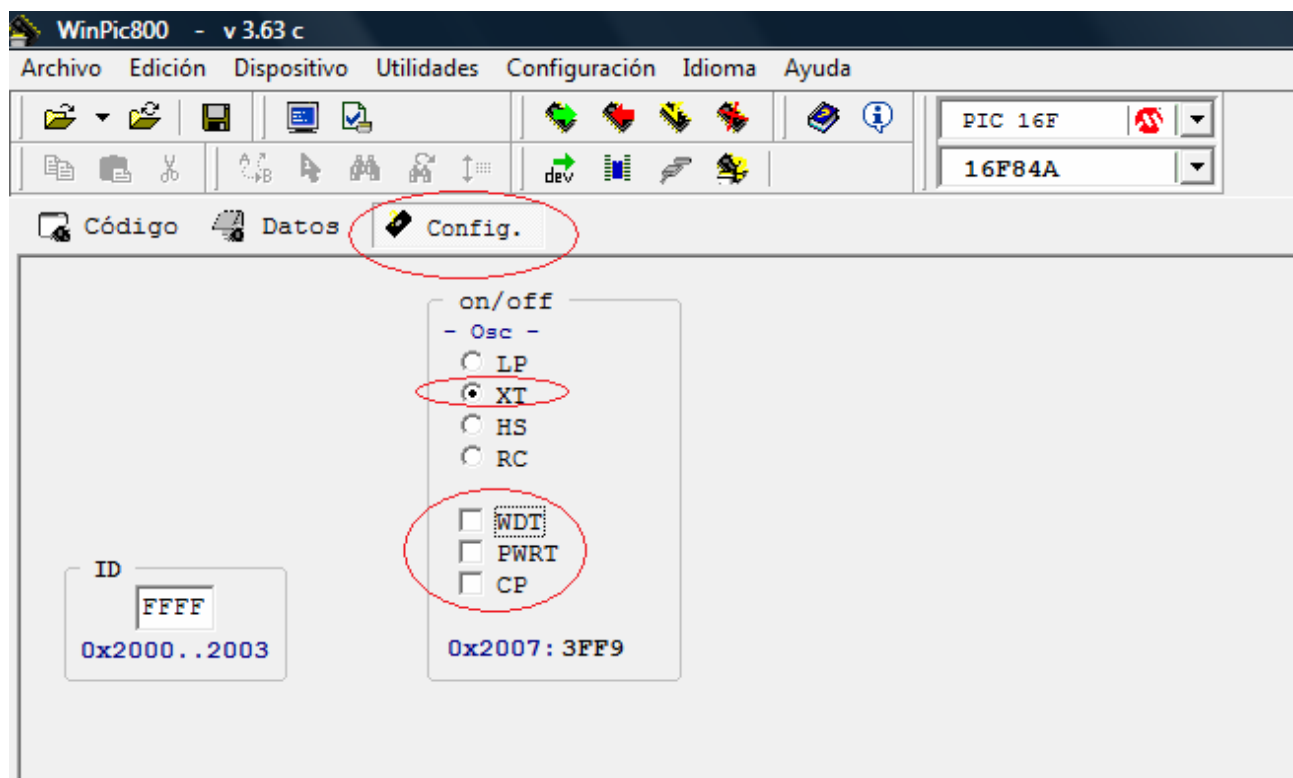
Para bajar el programa al microcontrolador deberá de usar el programador de PIC, en este caso en el laboratorio se dispone de programadores modelos iP3 (JDM Programmer), el cual se conecta al puerto serial de la computadora. Una vez conectado, debe correr la aplicación WinPic800, donde deberá de seleccionar el tipo de microcontrolador a programar (PIC16F84A).



Luego deberá de abrir el archivo .hex, que se encuentra en la misma carpeta donde esta el programa sim84.

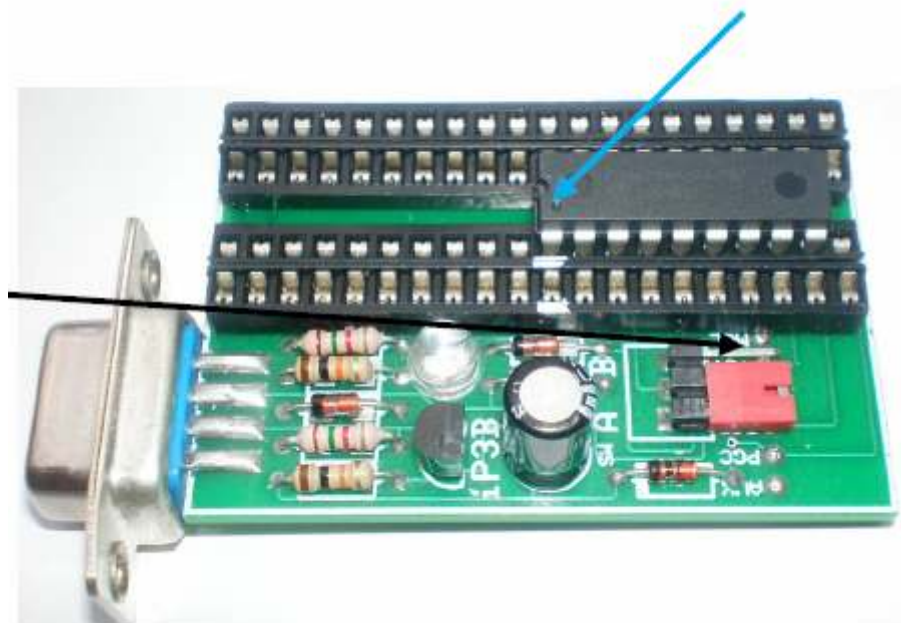


Una vez abierto el código deberá configurar, el tipo de oscilador a utilizar, si el perro guardián estará activado o si desea guardar el programa en el pic de manera que quede encriptado y que nadie lo pueda leer (esta opción de estar siempre desactivada), para esto despliegue el menú de configuración y selecciones las opciones como se muestra:



Una vez realizada esta operación, coloque el PIC16F84 en el modulo de programación tal cual lo muestra la figura:

PIN1



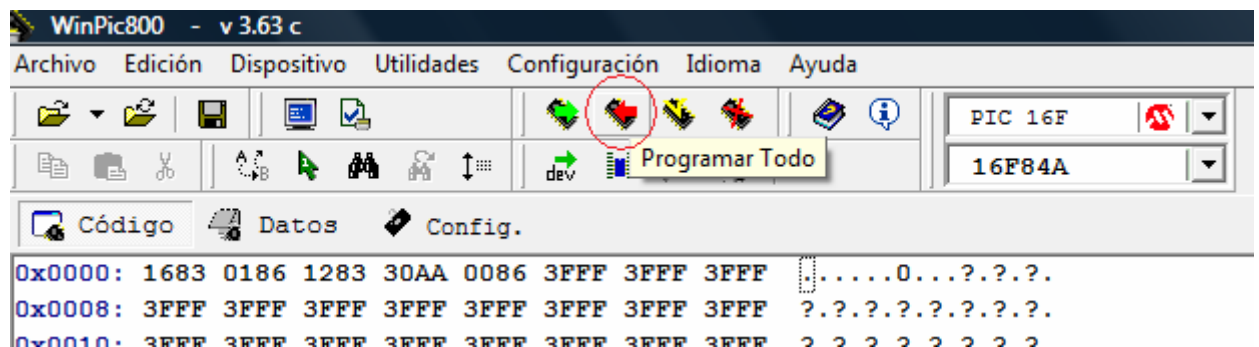
A través de esta aplicación puede verificar que tipo de integrado esta programando con la opción detectar dispositivo, como se muestra en la figura:



Al presionar inmediatamente saldrá el siguiente cuadro de dialogo y nos indicara que modelo de microcontrolador hemos colocado en la base del programador.



Luego de realizada esta operación inicie la descarga del programa hacia el PIC, presionando el botón de descarga, como se muestra en la figura:



Luego vera el siguiente cuadro de dialogo donde se confirma la correcta programación del dispositivo:



Una vez concluido esto, retire el microcontrolador de la base del programador, construya el siguiente circuito y verifique su correcto funcionamiento.

