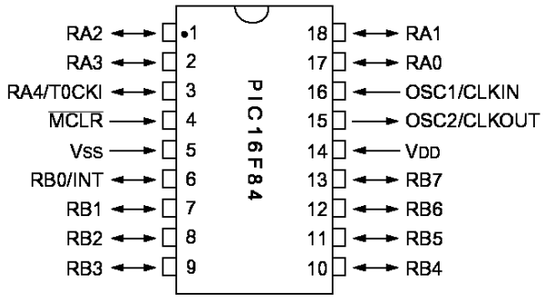
Tutorial Microcontrolador PIC (versión 3.5)

En este curso básico de microntroladores PIC se estudiará el PIC 16f84, por ser este el de uso más común entre los estudiantes que se inician en el tema.

El primer paso importante el ver el diagrama de pines del PIC16F84, en el cual se observa como están distribuidos sus pines.



Este circuito integrado cuenta con 2 puertos configurables como entradas o salidas según sea el caso y consta de 18 pines las cuales se encuentran asignadas de la siguiente manera:

VDD: Tensión positiva de alimentación.

VSS: Tensión conectada a tierra o negativa de alimentación.
O sea entre los pines 14 y 5 VDD(+)VSS(-) se coloca la alimentación la cual no debe sobrepasar los 5 Voltios.

OSC1/CLKIN: Entrada del circuito oscilador externo que proporciona la frecuencia de trabajo del microcontrolador.

OSC2/CLKOUT: Patilla auxiliar del circuito oscilador.

MCLR#: Patilla activa con nivel lógico bajo, lo que se representa con el símbolo # o con una línea superior MCLR. Su activación origina la reinicialización o Reset del PIC. El pin 4 (MCLR#), o sea, el Reset se debe conectar con una resistencia de 10 Kohm a Vcc para que el Pic funcione, si lo queremos resetear entonces pondremos un micropulsador con una resistencia de 100 Ohm a tierra.
También se usa durante la grabación de la memoria para introducir por ella la tensión VPP.

RA0-RA4: Son las 5 líneas de E/S digitales correspondientes a la Puerta A. La línea RA4 multiplexa otra función expresada por TOCKI. En ese caso sirve para recibir una frecuencia externa para alimentar al temporizador TMR0. O sea, RA4/TOCKI puede ser configurado a su vez como entrada/salida o como temporizador/contador. Cuando es salida se comporta como colector abierto, por lo tanto debemos poner una resistencia Pull-up a Vcc de 1 Kohm. Cuando es configurada como entrada, funciona como disparador Schmitt Trigger por lo que puede reconocer señales con un poco de distorsión.

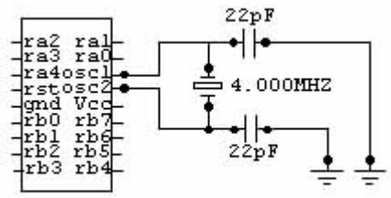
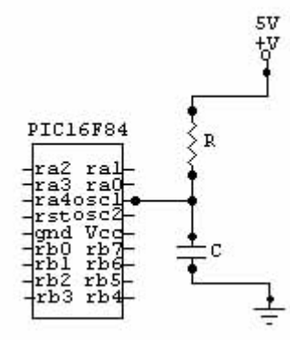
RB0-RB7: Son las 8 líneas de E/S digitales de la Puerta B. La línea RB0 multiplexa la función de servir como entrada a una petición externa de una interrupción.

La máxima capacidad de corriente para cada uno de los pines de los puertos se muestra en la tabla

	PUERTO A	PUERTO B
MODO SUMIDERO (sink)	25mA	25 mA
MODO FUENTE (source)	20 mA	20 mA

Oscilador Externo:

Es necesario para que nuestro PIC pueda funcionar, puede ser conectado de cuatro maneras diferentes. Se muestran en la siguiente tabla.

XT	Oscilador compuesto por un cristal y dos condensadores	
RC	Oscilador compuesto por una resistencia y un condensador.	

HS	Oscilador compuesto por un cristal de alta velocidad.	Con cristal.
LP	Oscilador compuesto por un cristal de baja frecuencia y bajo consumo de potencia.	Con cristal.

El siguiente paso importante para tener claro como debemos empezar a programar es conocer la tabla de registros. Esta tabla está dividida en dos partes llamadas BANCO 0 y Banco 1. Los registros más importantes para comenzar son: STATUS, PORTA, PORTB, TRISA y TRISB.

Para que nuestro PIC pueda trabajar debemos configurar sus puertos como entrada o como salida según sea el caso.

Esta asignación de pines de puertos como entrada o como salida se hace programando los registros TRISA y TRIS B.

TRISA es el registro donde se almacenan los bits que asignan un pin como entrada o salida del PUERTO A. Recordemos que el puerto A sólo tiene 5 pines, por lo tanto un ejemplo de esto sería:

Si TRISA (puerto A) es igual a 19h (en HEXA) o (0011001 en binario) entonces esto se leería,

TRISA	ASIGNACIÓN	ESTADO
RA0	1	ENTRADA
RA1	0	SALIDA
RA2	0	SALIDA
RA3	1	ENTRADA
RA4	1	ENTRADA

El bit menos significativo sé asigna desde RA0.

Si TRISB (puerto B) es igual a 32h (en HEXA) o (00110010 en binario), entonces esto se leería,

TRISB	ASIGNACIÓN	ESTADO
RB0	0	SALIDA
RB1	1	ENTRADA
RB2	0	SALIDA
RB3	0	SALIDA
RB4	1	ENTRADA
RB5	1	ENTRADA
RB6	0	SALIDA
RB7	0	SALIDA

NOTA: La asignación de valor a un registro se puede hacer en HEXA (ej:0x19), o en DECIMAL (Ej:d'12'), o en BINARIO (Ej:b'00101110').

A continuación comenzaremos a programar el PIC y veremos como ingresar estos valores en el TRIS A o TRIS B según sea el caso.

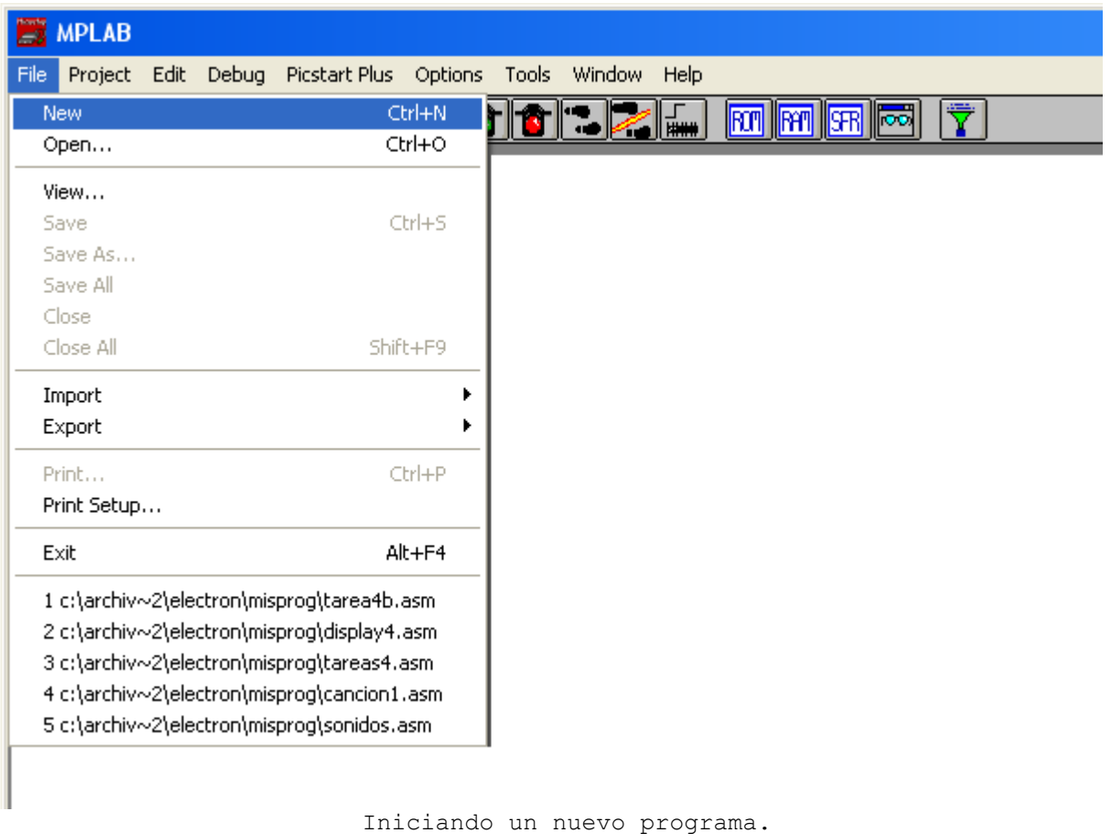
Para trabajar con los microcontroladores se debe conocer y manejar alguna herramienta de desarrollo.
MPLAB es la herramienta de desarrollo de microcontroladores PIC. (Microchip, Inc., los creadores de los microcontroladores PIC).

Si no estas familiarizado con las herramientas de desarrollo avanzadas, quizás encuentres el MPLAB un poco confuso al principio. También se necesita conocer el lenguaje de programación PIC.

En vez del MPLAB, se puedes usar el MPASM por ejemplo, un programa basado en MS-DOS, pero en esta explicación utilizaremos el MPLAB.

Todo en el MPLAB gira en torno al concepto de "project" (proyecto), que es un conjunto de archivos que necesitan ser procesados para compilar tu programa.

Para comenzar a trabajar con el programa MPLAB debemos simplemente abrir un nuevo archivo en la ventana File opción New como muestra la siguiente figura:



Iniciando un nuevo programa.

A continuación se lista un ejemplo de programa:

```
              list   p=16f84
              #include p16f84.inc
porta         equ 05h
trisa         equ 85h
status         equ 03h
mio           equ 0ch

inicio                        ;comienzo del programa
              movlw  0xaa     ;carga 10101010 en w
              clrw            ;limpia w
              clrf   mio      ;limpia registro mio
              movlw  0xf0     ; carga 11110000 en w
              movwf  mio      ;mueve w a mio
              movlw  0x00     ;carga 00000000 en w
              bsf    status,5 ;se ubica en el banco1 de la ram
              movwf  trisa    ;carga w en trisa asi define porta como salida
              bcf    status,5
              clrf   porta    ;limpia reg porta
              movlw  0x0f     ;carga 00001111 en w
              movwf  porta    ;mueve w en porta
              clrw
              movlw  0x0a     ;carga 00001010 en w
              addlw  B'101'   ;adiciona literal binario 101 a w
              movwf  mio      ;mueve w en mio
              comf   mio,1    ;complementa mio y guarda en mio por ser d=1
              incf   mio,1    ;incrementa mio y guarda en mio por ser d=1
              swapf  mio,1    ;intercambia nibles de mio y guarda en mio por ser d=1
              movf   mio,0    ;mueve mio en w porque d=0
              decf   mio,1    ;decrementa mio y guarda en mio por ser d=1
              decf   mio,1    ;idem anterior
              btfsc  mio,0    ;prueba si bit 0 de mio es cero. si lo es salta la próxima instrucción
              clrw
              goto   inicio ;salta el programa a inicio
              end             ;para informarle al compilador que termina el listado del programa
```

RECOMENDACIÓN: Siempre tener a mano el set de instrucciones.

Debemos destacar lo siguiente:

- ✓ Para mantener un orden, se recomienda presionar la tecla TAB entre la instrucción y el parámetro de la instrucción (Llamaremos parámetro a aquello que acompaña a la instrucción para que la misma se lleve a cabo).

✓ El símbolo de Punto y Coma " ; " indica el comienzo de un comentario, sirve para ubicarse mejor en el avance del programa y luego encontrar mas rápidamente una línea o bloque determinado. Todo lo que siga a un " ; " no generará código en nuestro proyecto. Lo que siga al " ; " no influye en el programa.

Analicemos las líneas de programa ejemplo:

```
            list  p=16f84
            #include p16f84.inc

porta       equ 05h
trisa       equ 85h
status      equ 03h
mio         equ 0ch
```

Primero debemos especificar con que microcontrolador estamos trabajando, en las dos primeras líneas especificamos que vamos a trabajar con el PIC 16F84.

Luego definimos la posición en memoria de nuestros registros, el mapa de memoria RAM se presenta a continuación

Para comenzar destacaremos los registros:

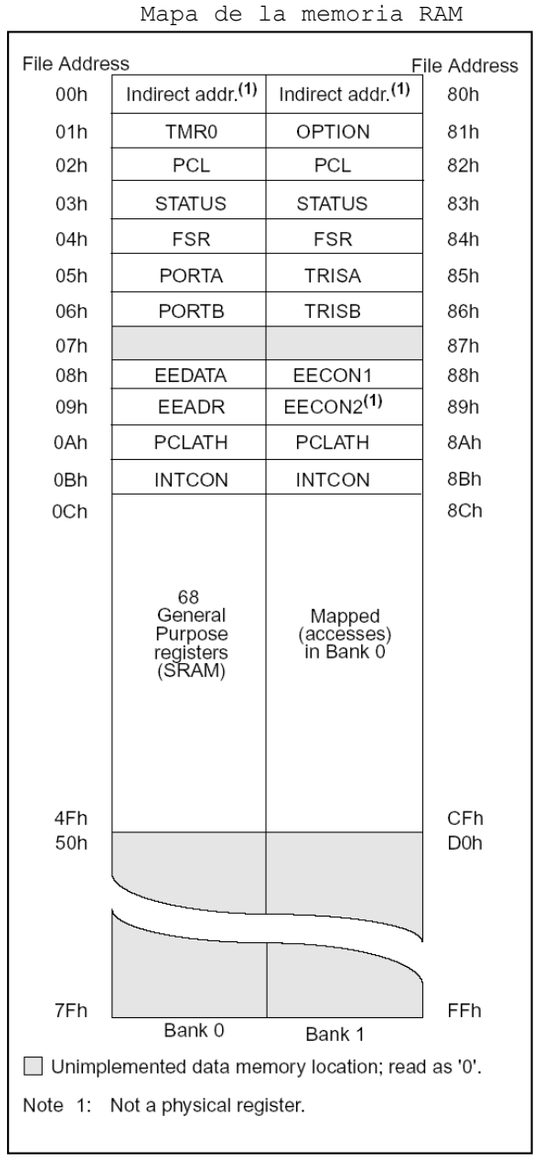
```
porta           posición 05h
portb           posición 06h
trisa           posición 85h
trisb           posición 86h
status          posición 03h
mio             posición 0ch
```

Definimos los nombres de nuestros registros, en el caso del Puerto A lo llamamos porta, es importante que siempre que nos queramos referir al Puerto A lo hagamos con el nombre asignado. Por ejemplo si en lugar de "porta equ 05h" colocamos "hola equ 05h" cada vez que queramos referirnos al Puerto A debemos hacerlo como "hola". También se debe respetar si lo escribimos en mayúsculas o minúsculas. Mi recomendación es respetar los nombres de la figura del mapa de la RAM.

Puede suceder que cuando vemos el listado de un programa, omitan el listado de registros especiales (o sea los que aparecen en el mapa de la RAM), esto es debido a que la instrucción **#include p16f84.inc** hace el llamado a un archivo librería que contiene este listado. Mi recomendación es colocar el listado de los registros especiales que usaremos en el programa para así mejorar la lectura y revisión del mismo.

El registro **mio** es un registro de propósito general, en este caso localizado en la posición de memoria **0ch**, los nombres de estos registros los elegimos nosotros.

Mapa de la memoria RAM



En algún momento deberemos estudiar la función de cada bit de los registros especiales, por ejemplo el registro STATUS se compone de la siguiente forma:

7	6	5	4	3	2	1	0
IRP	RP1	RP0	TO	PD	Z	DC	C

Pero aquí mencionaremos solo la función de dos de sus bits

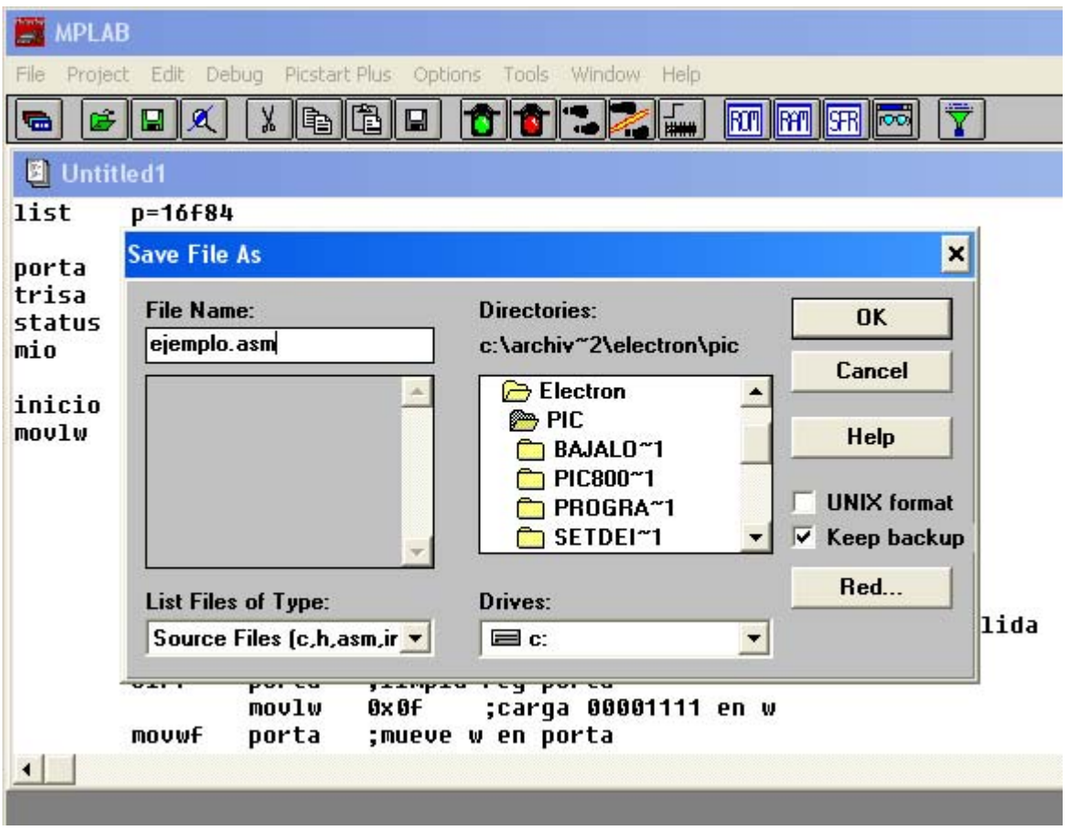
✓ **Si en el bit 5 (RP0) del registro STATUS hay un CERO entonces estamos en el BANCO 0.**

✓ **Si en el bit 5 (RP0) del registro STATUS hay un UNO entonces estamos en el BANCO 1.**

Anteriormente ya mencionamos la función de los registros **trisa** y **trisb.**

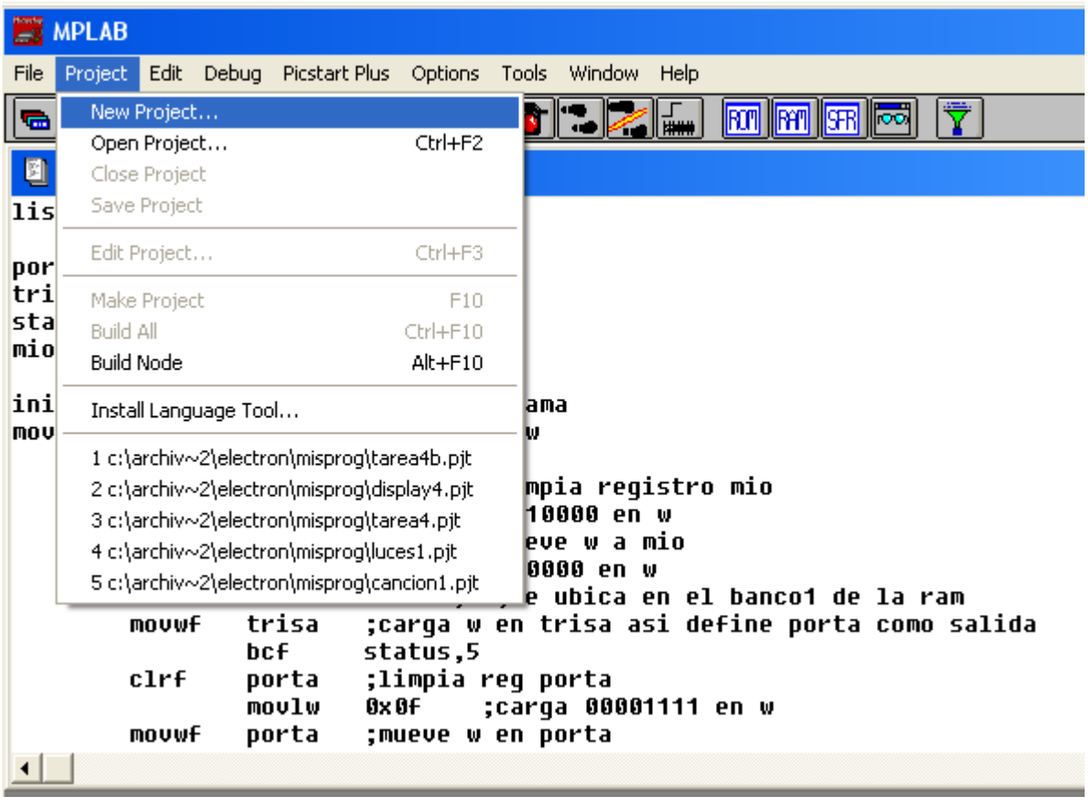
Al lado del resto de las instrucciones aparece una breve descripción de su función.

Finalmente debemos guardar este programa, para esto vamos al menú **File**, luego **Save As** y elegimos el directorio y un nombre, en este caso le pondremos ejemplo.asm, luego **OK**.

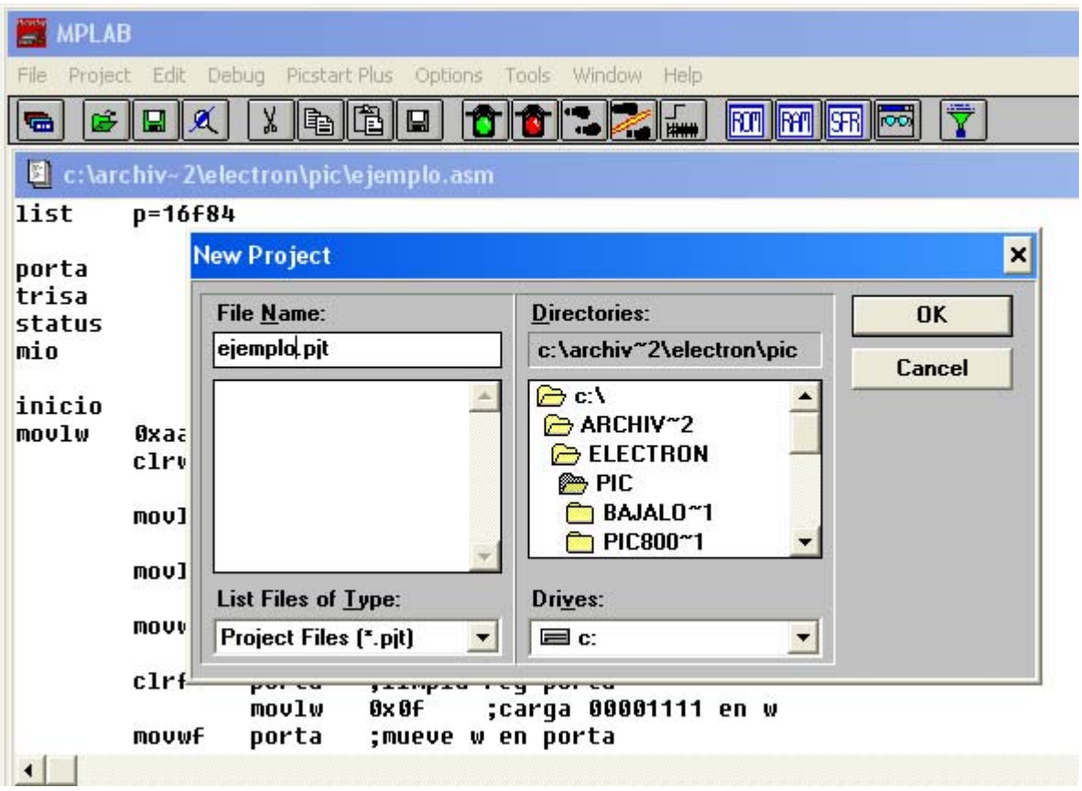


Veremos a ahora como compilar este programa y detectar los errores de programación.

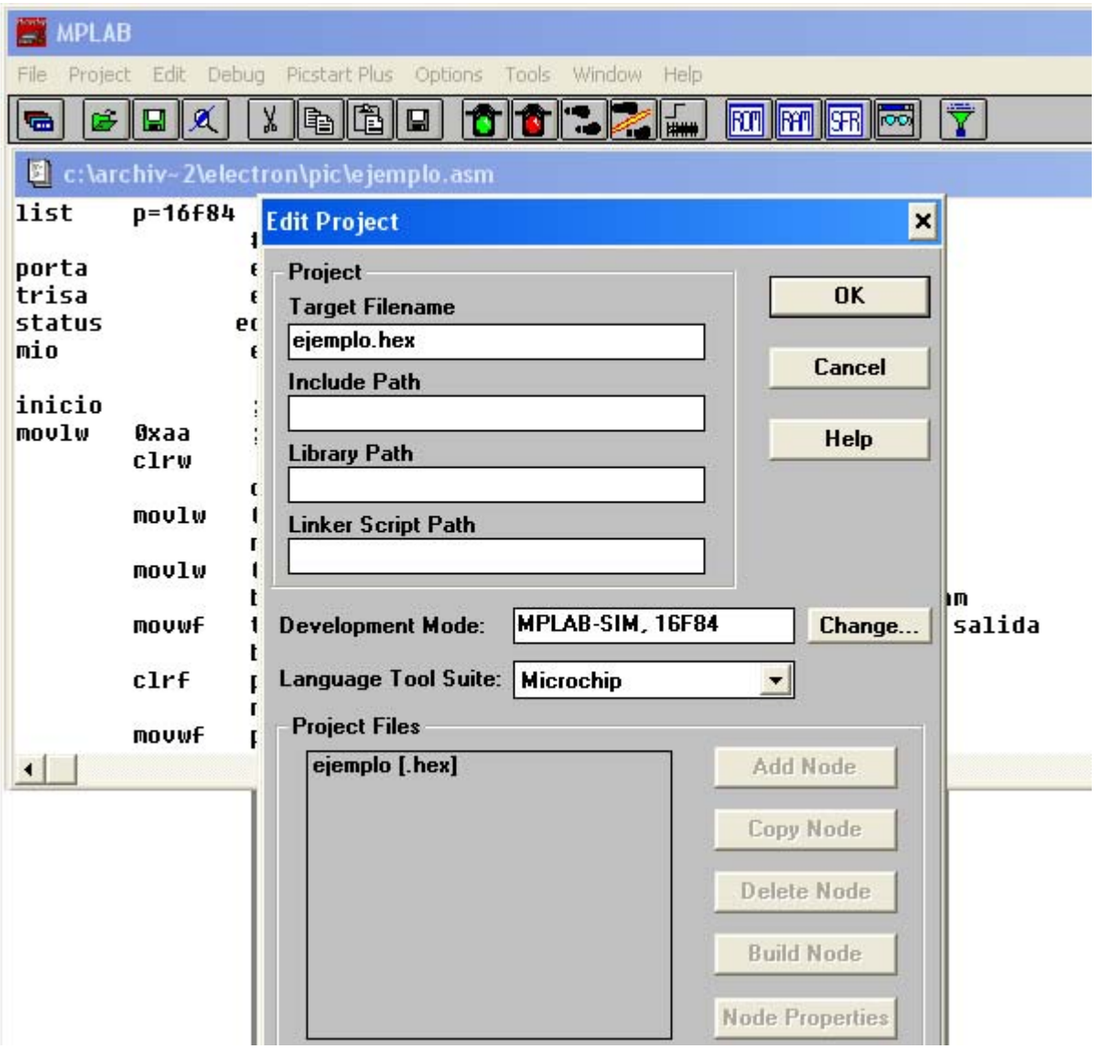
Lo primero es hacer clik en el menú **Project** y luego en **New project**, como indica la figura:



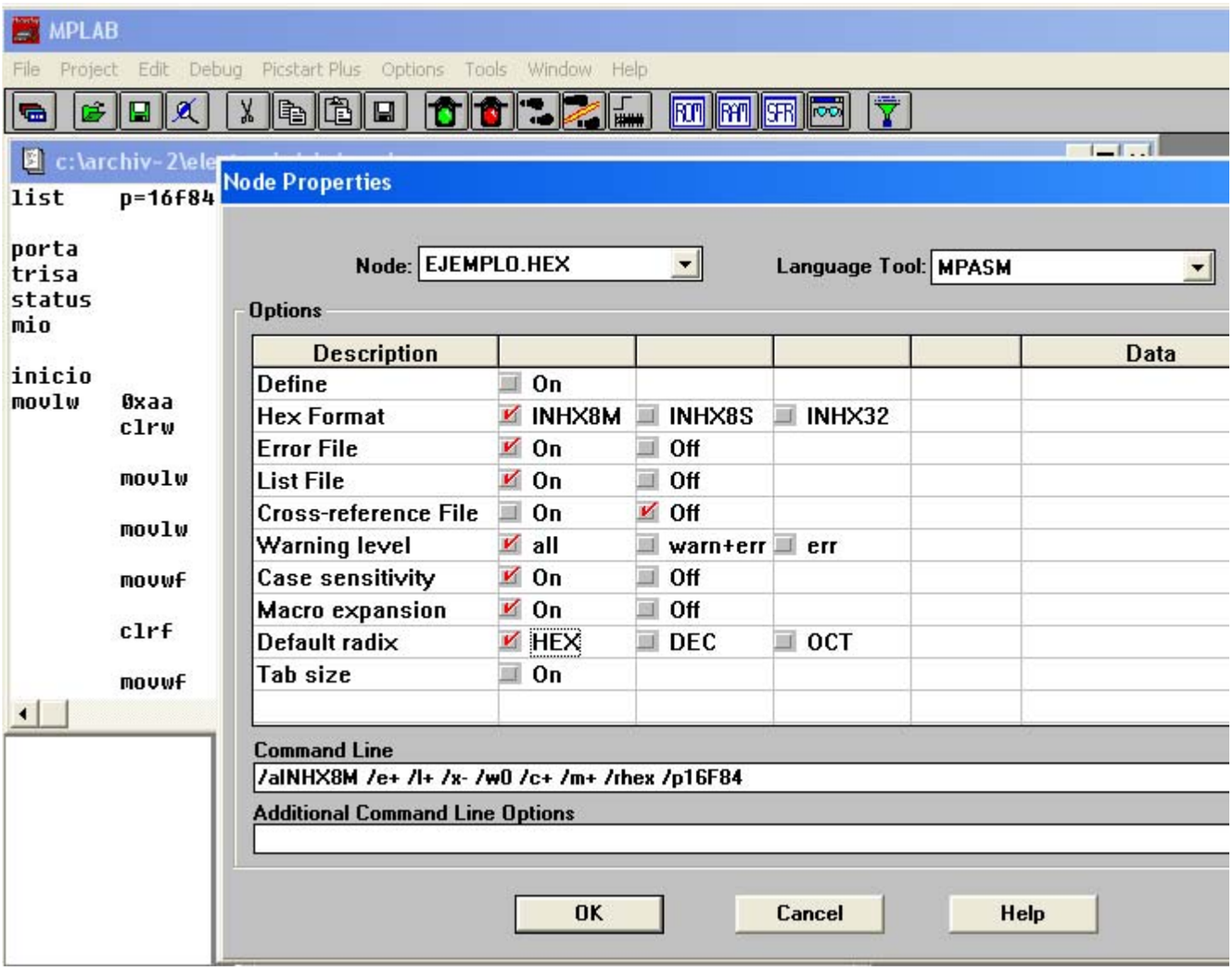
El siguiente paso es darle un nombre al proyecto, se recomienda darle el mismo nombre que al archivo en assembler, en este caso ejemplo.pjt.



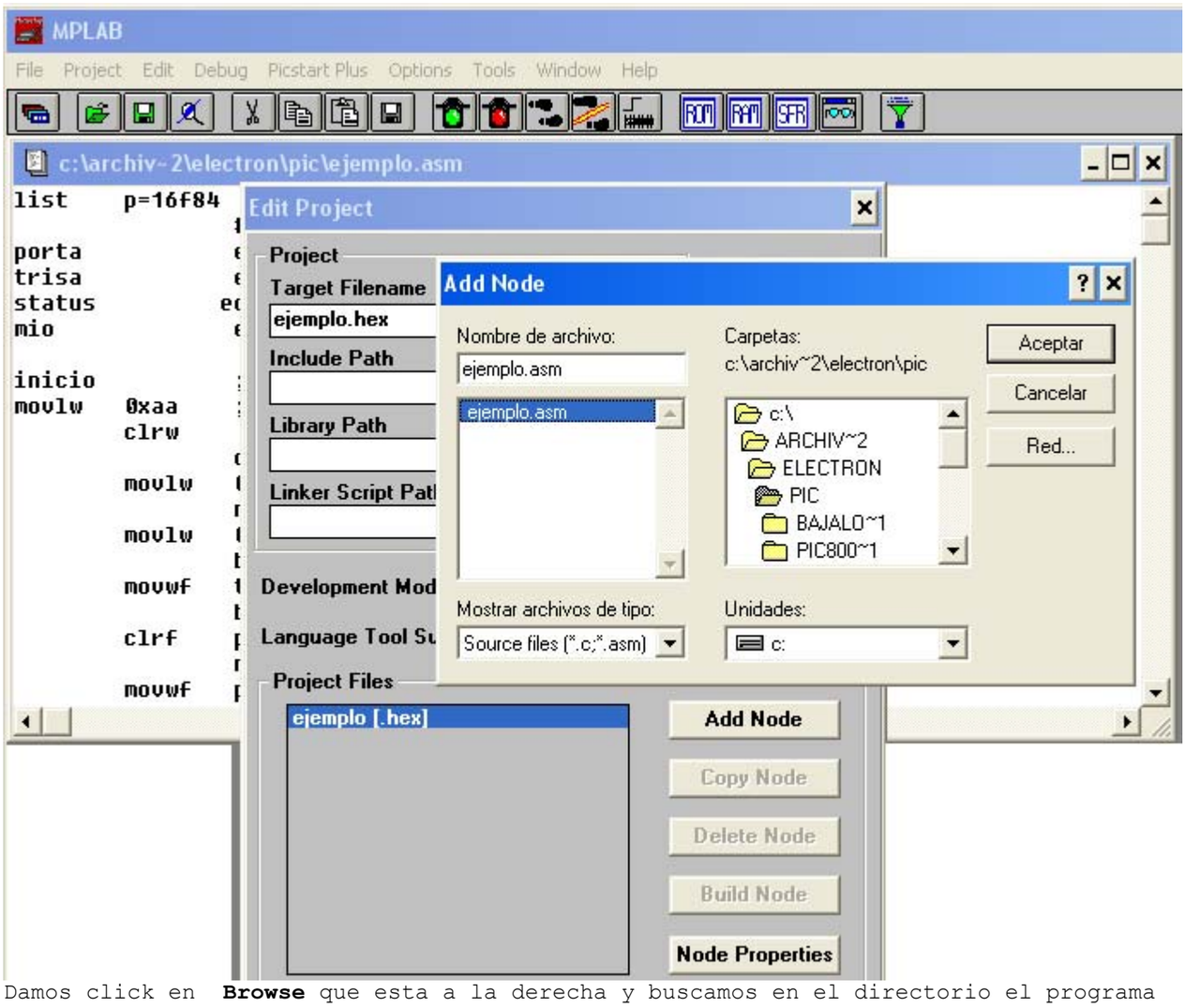
Una vez echo esto, aparece la siguiente ventana:



En la parte de abajo, donde dice **Project Files** pinchamos la opción ejemplo [.hex] al hacer esto se activa la opción **Node Properties** que esta abajo a la derecha, debemos pinchar esta y se abre la siguiente ventana:



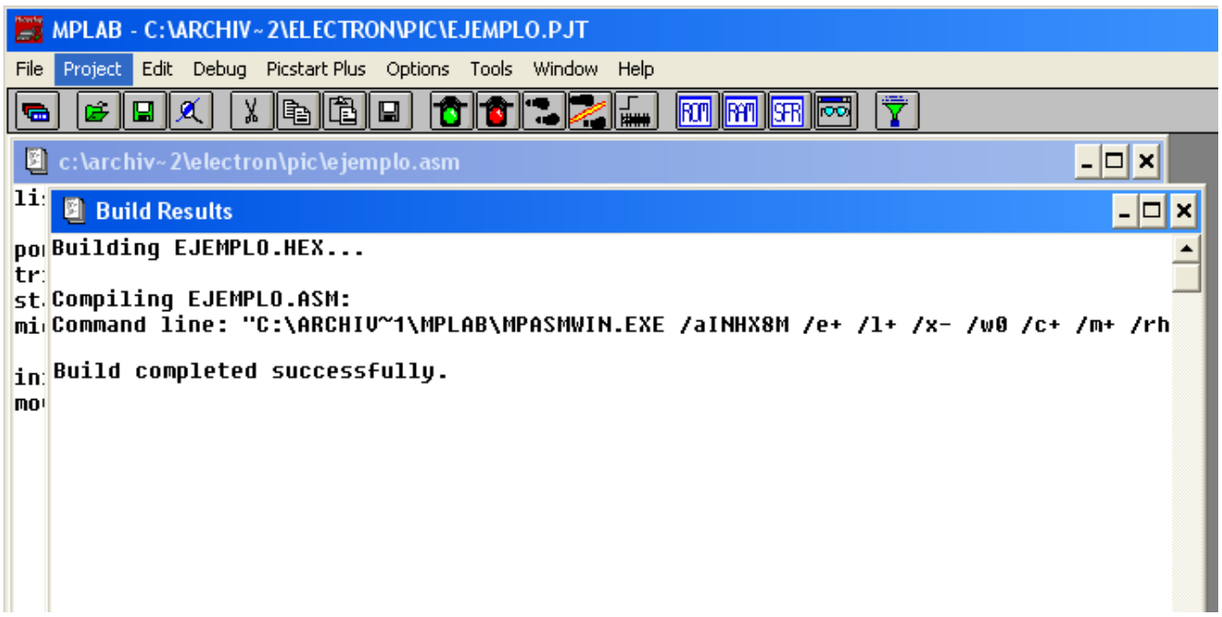
En esta sección, debemos marcar como muestra la figura y luego OK. Luego volveremos a la ventana anterior solo que ahora se activó la opción Add Node algunas opciones más arriba de la última que pinchamos, por supuesto elegimos esta y se ve la siguiente pantalla:



Damos click en **Browse** que esta a la derecha y buscamos en el directorio el programa original, en este caso ejemplo.asm y luego en **OK**, con esto estamos relacionando un archivo de assembler con el proyecto en hexa.

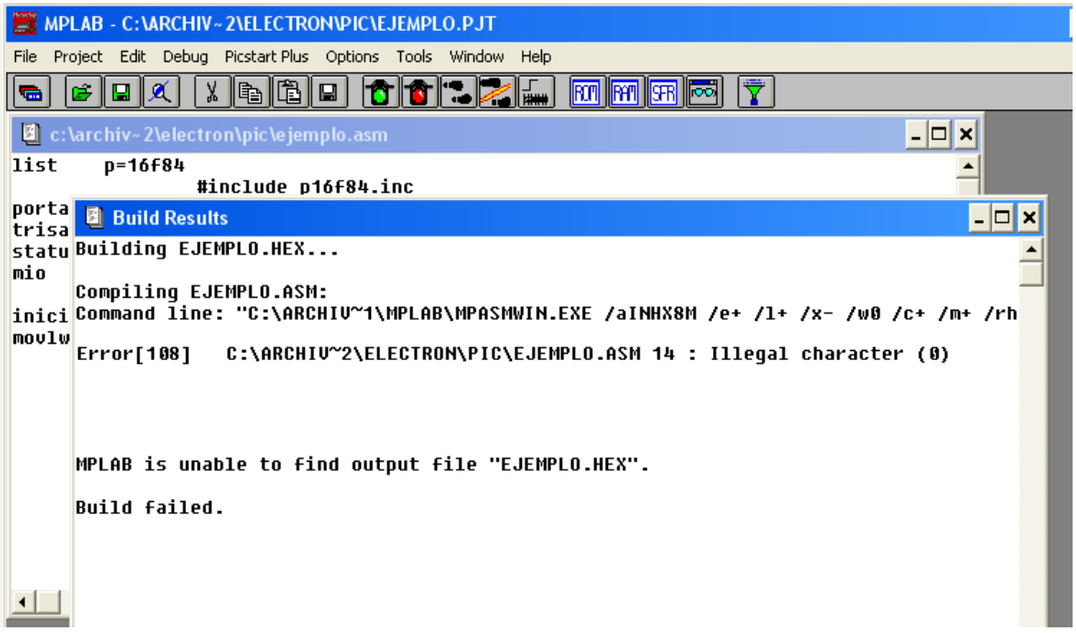
Y finalmente volvemos a la primera pantalla y le damos al **OK** que esta en la parte superior.

A continuación debemos verificar que no haya errores para eso vamos nuevamente a la opción **Project** y veremos que hay nuevas opciónes que antes no estaban disponibles, elegimos **Build All**, y si todo esta correcto aparecerá una pantalla como esta:



Donde en la parte final podemos leer: **Build completed successfully**. Esto implica que el program se compilo satisfactoriamente.

En caso de haber algún error, podemos hacer doble clic en el mensaje de error y este nos enviará directamente a esta línea.

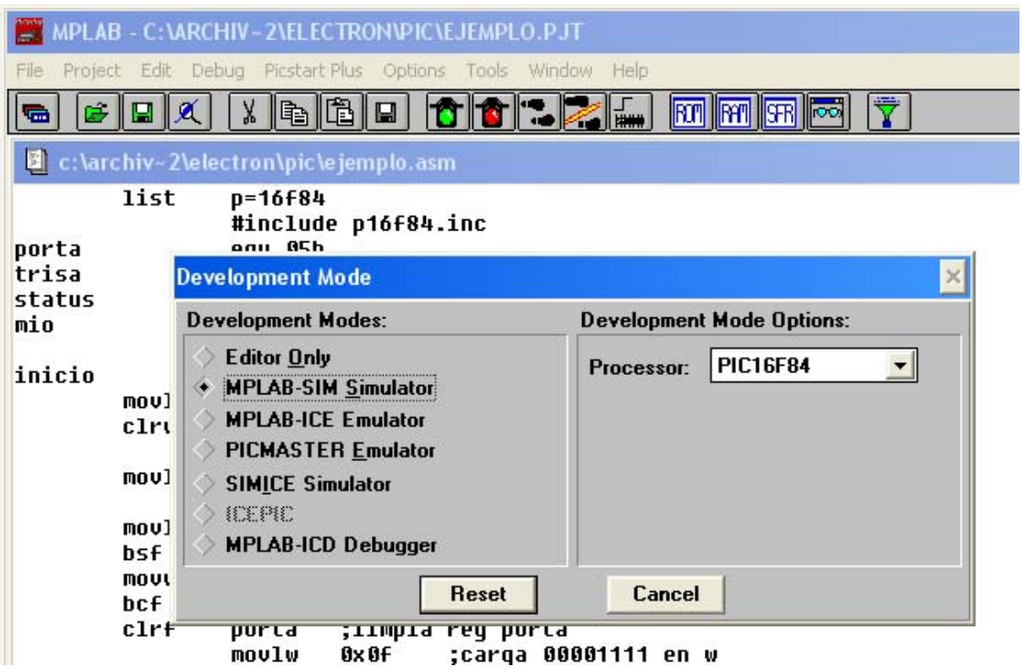


Una vez que hemos pasado esta etapa, nos dedicamos a simular el programa, todo dentro del MPLAB.

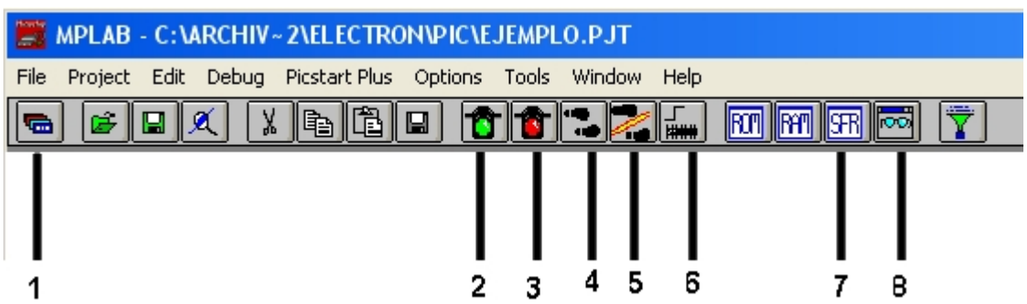
Esto consiste en ver paso a paso que está realmente haciendo el PIC, ya que si lo hacemos en tiempo real es demasiado rápido. Claro que también se puede dar el caso contrario, es decir tener un programa con tiempos largos que tampoco podríamos simular porque tardaría demasiado, en ese caso la solución es alterar algunas líneas de programa que maneje los tiempos y luego proceder a simularlo.

Lo primero es activar la simulación, para esto vamos al menú **Options** y le elegimos la opción **Development Mode.**

En la ventana que aparece marcamos donde dice MPLAB-SIM Simulator y además buscamos el PIC que estamos usando, en este caso el PIC16F84.



Primero analizaremos las herramientas básicas para hacer una buena simulación, en la figura siguiente están marcados los principales botones para este trabajo.



1. Conmutador de la barra de herramientas: Al pinchar esta venta tenemos mas opciones a nuestra disposición.

2. (Run)Ejecuta la simulación en "tiempo real"(no olvidemos que en el simulador emula el funcionamiento del microcontrolador y es mucho más lento que este).

3. -Halt the processor, se detiene la ejecución del programa.

4. Step: Avanza paso a paso por las instrucciones del programa.

5. -

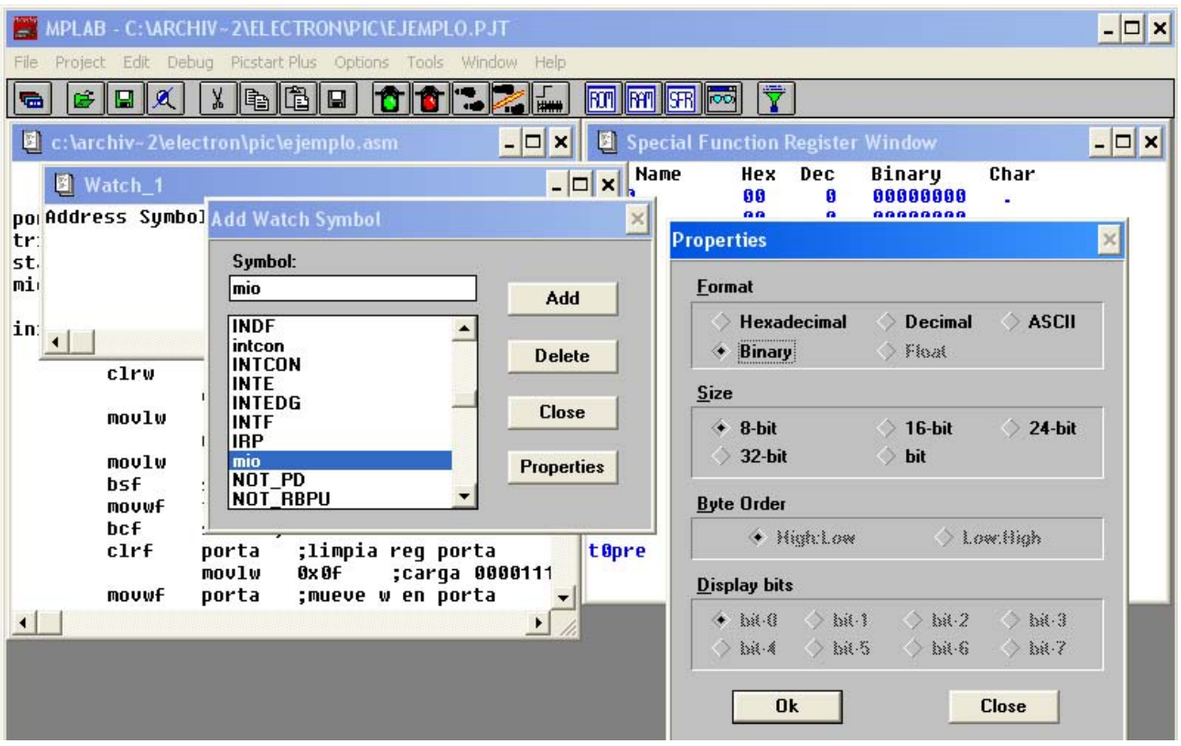
6. Reset: comienza de cero el programa, equivale a activar el pin N°4 del PIC, Master Clear (MCLR).

7. Special Function Register Window: Muestra el estado de los registros especiales, por ejemplo el registro status, puerto a y b, el acumulador, etc.

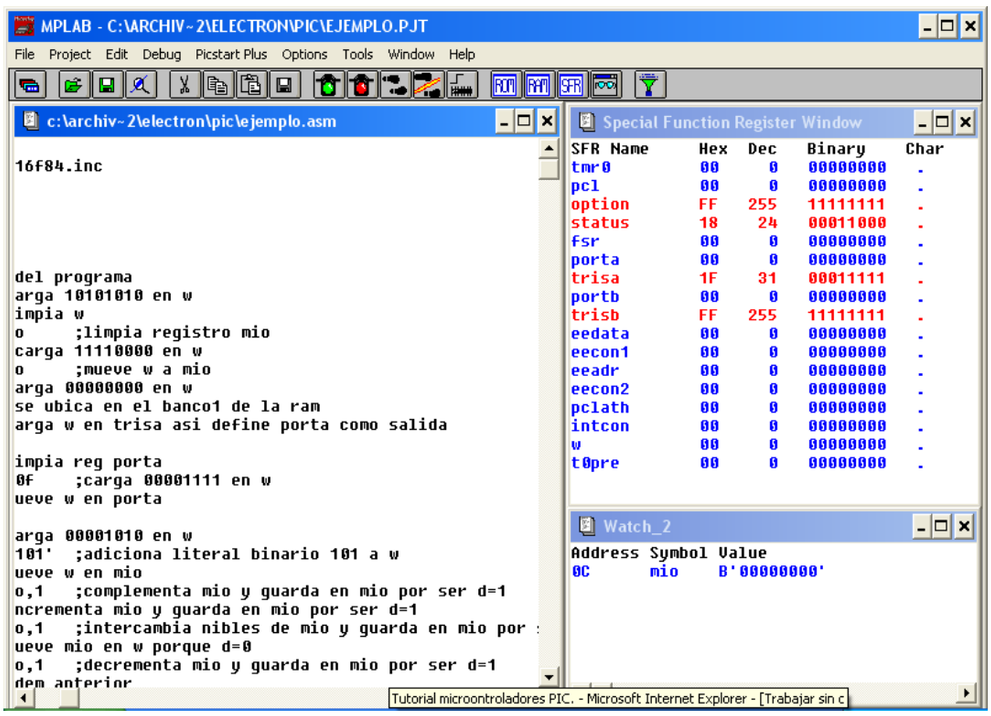
8. Watch Symbol: Ventana en la que podemos ver los registro de propósito general, que son los registro que usamos en nuestro programa y a los cuales de damos nombres propios.

Al presionar sobre el botón N°7 (**FSR**) aparece una ventana donde observamos los registros especiales.

Al presionar sobre el botón N°8 aparece otra ventana donde debemos elegir (**Add**) los registros de propósito general que queremos controlar, en **Properties** el formato de presentación (BINARIO - DECIMAL etc), al terminar hacemos click en **Close.**



Ahora, para comenzar la simulación es recomendable ordenar las ventanas de forma que entren todas en la pantalla.



Luego sólo debemos pinchar el botón Step (N°4) sucesivamente y con eso se ejecutará y se marcará la línea del programa que esta trabajando, otra opción es presionar F7.

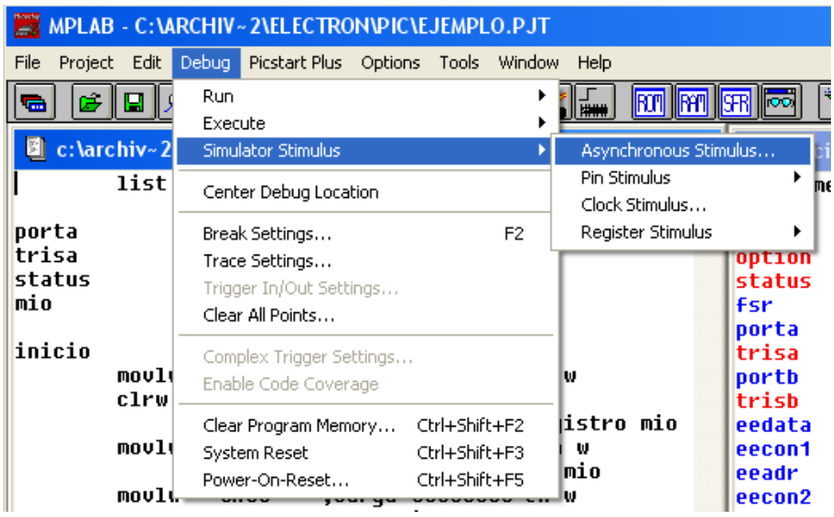
Para simular

las entradas de un PIC

Vamos al menú **Debug** -

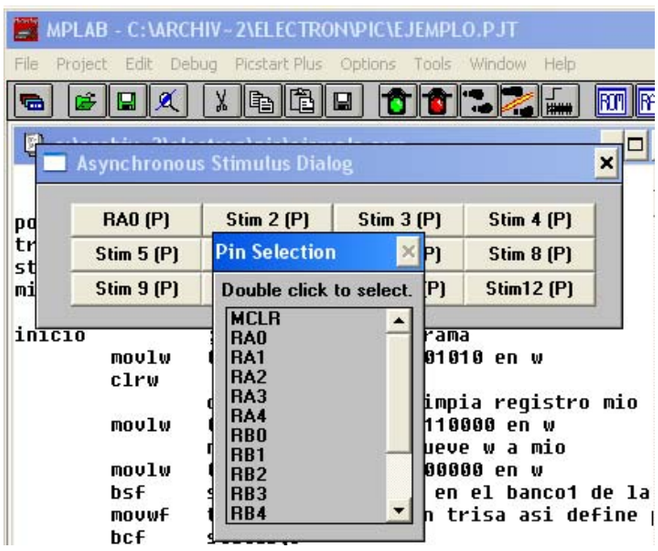
Simulator Stimulus -

Asynchronous Stimulus.

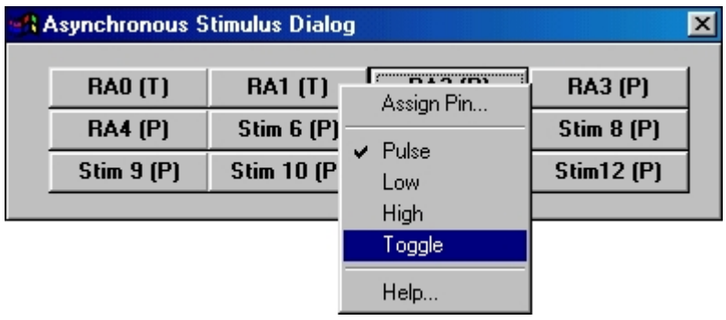


Luego aparecerá una ventana en la cual debemos elegir que pin le vamos a asignar a cada opción.

Posicionados sobre el primero elegimos con el botón derecho del mouse la opción **Assign Pin** y luego elegimos el pin que le vamos a designar haciendo doble click sobre este.
En la figura se va a elegir el RA0.



Luego es muy importante decirle al MPLAB que vamos a hacer una simulación paso a paso y no en tiempo real, para esto una vez que le hemos designado el pin correspondiente, volvemos a hacer click con el botón secundario del mouse pero esta vez elegimos la opción **Toggle.** Después de esto podemos observar que ya no hay una (P) a la derecha de botón, sino que ahora hay una (T).



Esta será una ventana mas que quedará abierta en pantalla. Cada vez que demos click sobre RA0 (T) cambiara su estado. Obviamente esto deberá hacerse durante la simulación.

RESUMIENDO:

Se debe seleccionar la ventana del programa para que al presionar F7 en el teclado o la botón Step en la parte superior, se produzca la simulación, es decir, para que corra el programa.
En la ventana de funciones especiales, se pone rojo él ultimo registro en cambiar.
Al igual que en el punto anterior, en la ventana de registros de propósito general, se pone rojo el ultimo en cambiar. Si queremos agregar mas registros para poder verlos en esta ventana debemos presionar en la parte superior izquierda y elegir la opción **Add Watch.**

Una vez satisfechos con la simulación, se debe cargar el programa en el PIC. Para esto usaremos el ICPROG. Hay apunte sobre el tema.