COMO PROGRAMAR UN PIC

En cuatro pasos

Un PICmicro es un circuito integrado programable. Microchip, su fabricante dice: Programable Integrated Circuit.

Programable quiere decir que se puede planificar la manera como va a funcionar, que se puede adaptar a nuestras necesidades. En otras palabras que el integrado es capaz de modificar su comportamiento en función de una serie de instrucciones que es posible comunicarle.

Toda esta actividad : "Programar un PIC", se puede dividir en cuatro pasos:

EDITAR

COMPILAR

QUEMAR EL PIC

PROBAR EL PROGRAMA

Veamos estos pasos, primero rápidamente y después con más detalle.

1 .- Un Vistazo rápido.

1.1 . - Editar

Editar es escribir el programa, es hacer una lista de instrucciones en un lenguaje que nos permita indicarle al PIC lo que deseamos que haga.

Existen varios lenguajes como: Ensamblador, Basic, C, etc.

Todos ellos pretenden acercarse a nuestra manera de pensar y de hablar. Sin embargo los PIC no conocen mas que unos y ceros. Por eso es necesario el siguiente paso.

1.2 .- Compilar

Compilar es traducir el programa al lenguaje de máquina que ¡ Si ! "entiende" el PIC. Para realizar esta traducción hacemos uso de un software que transforma el "Programa Fuente", aquel que editamos en el paso 1 en otro que si podemos comunicarle al PIC.

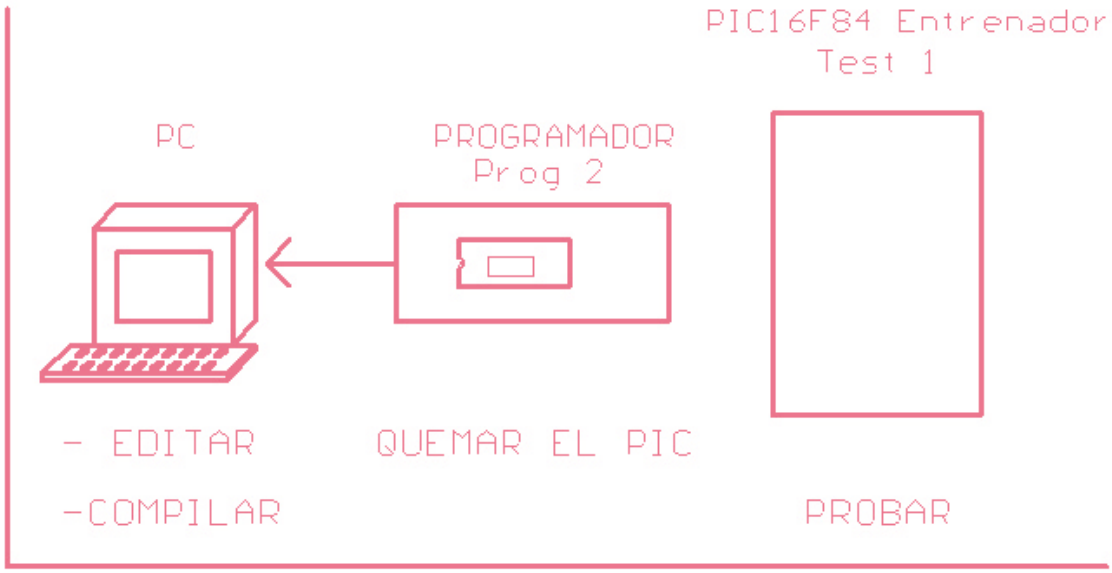
1.3 .- Quemar el PIC

En este paso se grava el programa en el PIC.
Mediante una tarjeta electrónica y un poco software se pasa el programa compilado de la PC al PIC. Son solamente unos cuantos Cliks y listo.

Es necesario hacer una aclaración en este momento. Frecuentemente le llamamos Programador de PIC a la tarjeta electrónica que transfiere el programa compilado de la PC al PIC. Esta bien mientras entendamos que este aparato no va ha pensar por nosotros y que es incapaz de programar instrucciones por sí mismo.

1.4 .- Probar el Programa
Bueno en este paso se trata de verificar el funcionamiento del programa.
Se trata de comprobar que el PIC se comporta como lo programamos. Si todo salió bien, pues fantástico y si no comenzamos de nuevo en Editar

Para realizar esta actividad podemos hacer uso de un Protoboard, alambrar los Led´s o botones, instalar la fuente, poner el reloj , etc. etc. Pero como no se trata de aprender a armar circuitos en Protos sino de aprender a programar Pics es mejor hacer uso de una tarjeta “Proyecto” que ya tenga todo esto y este lista para ser usada.



Cuatro pasos para programar un PIC
Fig1.

2.- Detallando

2.1 .- Editar

Para Editar el programa se hace uso de MPLAB el software que ofrece Microchip gratis en su página: www.microchip.com

Icono de MPLAB
Fig2.

En este software se teclean las instrucciones. Como es un software en ambiente Windows, la historia es bien conocida: File, New, Save, Save as, etc.
Un detalle:
Si editamos en Ensamblador, que es lo que haremos en este artículo, el archivo que se genera tiene una extensión .ASM, por ejemplo si el nombre del programa es timer, el nombre completo con extensión será timer.ASM .

2.2.- Compliar

Para Compilar el programa se hace uso de MPASAM felizmente gratis y que esta integrado en MPLAB. Es decir cuando instalas MPLAB también encontraras instalado MPASAM. ¡Fácil!

Icono de MPASAM
Fig 3.

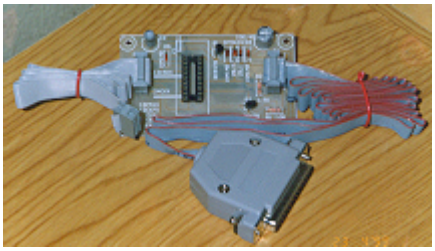
Con este software traducimos el programa “FUENTE” en un lenguaje de “MAQUINA”. Si el programa fuente es timer.ASM después de compilado será timer.hex .

2.3.- Quemar el PIC

Como se recordara para quemar el PIC se requiere un hardware, una tarjeta, pues bien, dos alternativas:

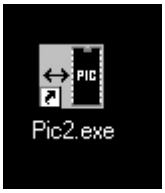
a).- PICSTART PLUS
Es el programador de MICROCHIP. Con él se puede quemar toda la línea de PICs y además esta bien integrado con MPLAB . Desafortunadamente este no es gratis.

b).-Prog2
Existen varios programadores que se ofrecen en Internet. Uno de ello es el llamado JDM84 que se ofrece con el numero de parte Prog2. Con este no se pueden grabar todos los PIC pero sí los más importantes: el PIC16F84 y el PIC12C508, de los que ya hablaremos más adelante. El software necesario para usar este programador viene en el disco que acompaña a la tarjeta. A saber: Icprog.exe y Pic2.exe . Este programador tampoco es gratis pero si es más económico .



Programador Prog2
Fig. 4a

Icono de Icprog.exe
Fig. 4b



Icono de Pic2.exe
Fig. 4c

Entonces colocamos el PIC en el Programador Prog2, abrimos el archivo *.hex , por ejemplo timer.exe , hacemos CLIK en programar y listo.

2.4 .- Probar el Programa

Ya tenemos el PIC con su programa dentro. Lo que resta por hacer es insertarlo en la tarjeta Test1 y probar que lo que pesamos que debería de hacer es exactamente lo que queremos. Si no pues volveremos a Editar

Tarjeta para probar el programa Test1
Fig. 5

3.- Requerimientos mínimos.

Los recursos mínimos que requiere el programador de PICs son los siguientes.

- Una PC

  Mínimo
  - 386, 486 o Pentium
  - Microsoft Windows 95/98
  - 16 MB de RAM
  - CD-ROM drive

  Se recomineda
  - Pentium
  - 32 MB RAM
  - Internet Explorer 5.0

- Un Editor y un Compilador.

  MPLAB contiene ambos, el editor y el compilador
  Pude bajarlos de la dirección de Microchip www.microchip.com
  Por favor instálelos en su PC lo mas pronto posible.

- Un Programador

  El más económico lo ofrece PICmicroEstudio con el numero de parte:
  Debería adquirirlo e instalar el software: Icprog.exe y Pic2.exe

- Una tarjeta para probar sus programas.

La más económica la ofrece PICmicroEstudio con el numero de parte:
Debería adquirirla e instalar su software en su PC.

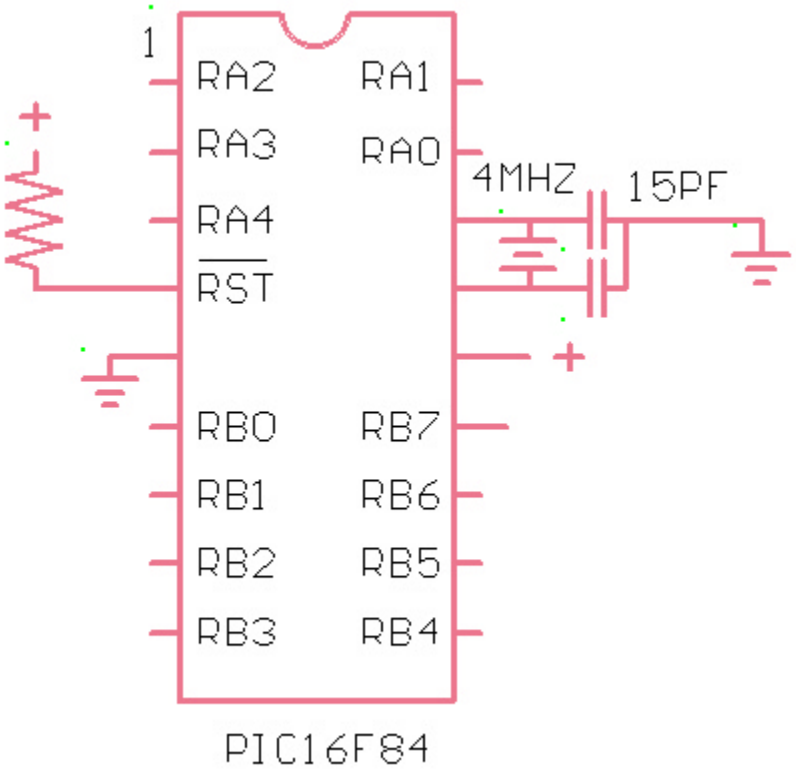
- Un par de integrados PIC16F84-04P

4.- A Programar el PIC16F84

Microchip ofrece un gran número de circuitos integrados programables PIC. Uno de los más populares es el PIC16F84. Es tan amigable que casi se podría decir que es “EL mejor amigo del programador de Pic’s”.

4.1.- Los pines del PIC16F84

En la Fig. 6 se presenta un dibujo del integrado PIC16F84 con los nombres de sus pines. Por favor tome muchos minutos para observar esta figura y por favor apréndasela de memoria.



Los pines que se programan son RA0 a RA4, Puerto A y RB0 a RB7, Puerto B

Fig. 6

Este PIC se alimenta por los pines 5 y 14. Pin 5 a tierra y 14 a +5 VCD.

Los pines 15 y 16 son para conectar el CLOCK (reloj) Un cristal de 4 MHZ y un par de condensadores de 15 pf bastan para que el PIC este listo para trabajar.

En el pin 4 se conecta el RESET (reiniciar) Una resistencia a positivo +5 es suficiente para que el PIC arranque. Si este pin se mantiene en “0” el PIC esta quieto, pero cuando se pasa a positivo “1” el PIC arranca y comienza a ejecutar el programa ¡Siempre desde el principio¡

El resto de los pines cuyos nombres son:

RA4, RA3, RA2, RA1, RA0 y

RB7, RB6, RB5, RB4, RB3, RB2, RB1, RB0

¡Son los que vamos a programar¡

Al conjunto de pines RA se le llama “puerto A” y al conjunto de pines RB se le llama “puerto B”. No pierda de vista que RA0 es el pin 17 y que RB3 es el pin 9 y así.

Lo primero y quizá más importante que podemos programarle a estos pines es:

Si van a ser ENTRADAS o SALIDAS

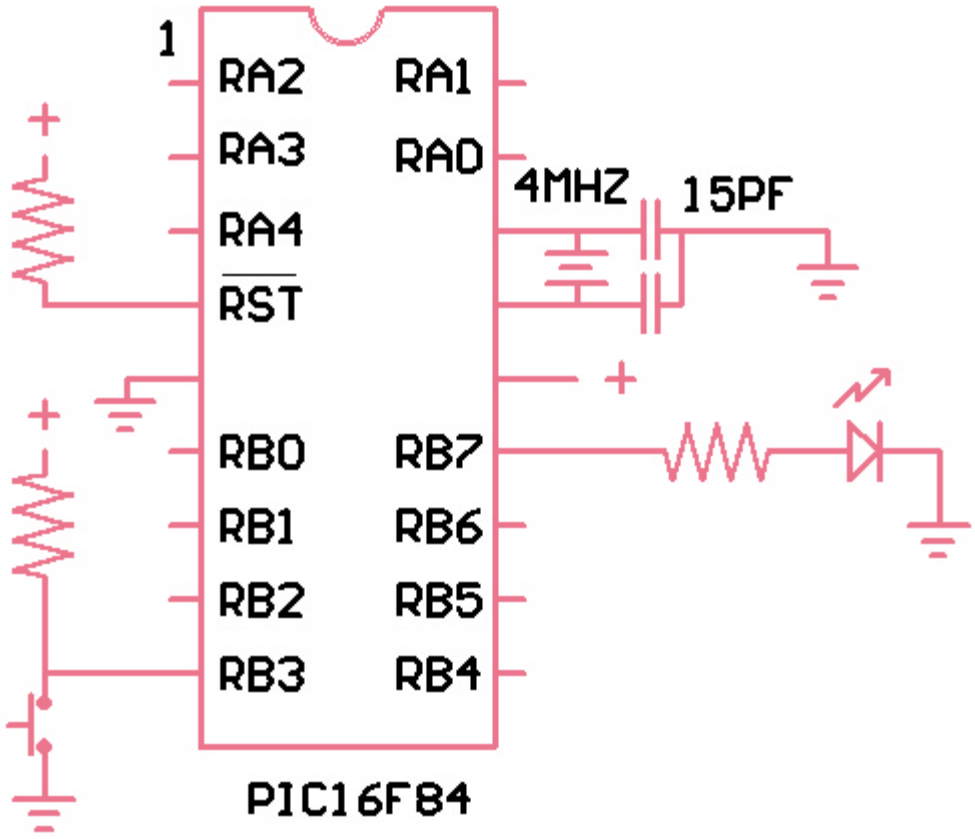
Cualquiera de estos pines puede ser programado como entrada o como salida.

Si un pin se programa como ENTRADA, entonces podrá detectar un voltaje: “0” o “1”, que a su vez puede ser la apertura o el cierre de un interruptor, un botonazo, o la acción de un sensor.

Si un pin se programa como SALIDA, será capaz de prender un led, energizar un relevador o un solenoide, etc.

En la Fig. 7 si RB7 se programa como salida. Entonces un “1” prendería el led y un “0” lo apagaría.

En cambio si RB3 se programa como entrada. Al aire se detecta un “1” y al oprimir el botón se detecta un “0”.



RB3 se programa como entrada. Al oprimir el botón el PIC detecta un “0”
RB7 se programa como salida un “1” enciende el LED

Fig. 7

En realidad de esto trata la programación de los PIC. Detectar botonazos o niveles de voltaje para energizar o apagar un led, un relevador, un motor, etc.

Quizá algún día exista algún programa que diga:

Has RA0 entrada y
RA1 entrada

Has RB0 a RB1 salidas

Si RA0 se prende, entonces
Prende RB0
Espera 10seg
Prende RB1
Etc..

Dar este tipo de instrucciones al PIC sería maravilloso.

4.2.- La memoria para el programa (program memory)

El programa que Editamos, Compilamos y que finalmente gravamos en el PIC, cuando lo quemamos, se almacena en su memoria.

En esta memoria se guardan las instrucciones del programa. Una por una, como en renglones de un cuaderno. Un renglón, una instrucción, otro renglón otra instrucción. La cantidad de renglones disponibles esta limitada por la capacidad de la memoria. En el PIC16F84 contamos con 1024 renglones para escribir en cada uno de ellos las instrucciones del programa.

La memoria del PIC16F84 es reprogramable. Esto quiere decir que si el programa no nos gusta podremos volverlo a grabar una y otra vez.

Otra característica de esta memoria es que es permanente. Una vez que gravamos el PIC, lo podemos retirar del programador, guardarlo en la bolsa y llevarlo a otro lugar, entonces insertarlo en alguna tarjeta de prueba o de una aplicación industrial. No se borra al desconectar el PIC.

Como esta memoria se puede grabar y volver a grabar mediante señales eléctricas se llama Flash (en el PIC16C84: EEPROM)

Entonces diríamos:

¡El PIC16F84 tiene una memoria de programación Flash de 1024 instrucciones (1K)!

Finalmente y para llevar las cuentas bien ordenadas, a cada instrucción le asignamos un numero consecutivo: del 0 al 1023 a este numero le llamamos dirección. Así podríamos decir la instrucción numero 55 ... o en la dirección 55 se encuentra la instrucción fulana o zutana.

Dirección (No. De instruccion)	Memoria del programa Instrucciones ↓
1	
2	
3	
4	
1023	

Memoria del programa del PIC16F84
Memoria Flash de 1024 instrucciones

Fig. 8

4.3.- La memoria para los datos (data memory)

Esta memoria se utiliza principalmente para almacenar las variables del programa, por ejemplo el valor de un contador que va cambiando según el numero de veces que se activa un interruptor o el tiempo que dura un proceso.

La principal característica de esta memoria llamada RAM es que es volátil. Es decir cuando el PIC se desconecta, esta memoria pierde sus valores.

El PIC16F84 tiene 80 “renglones” de memoria de datos que están numerados del 0 al 79. Cada uno de estos registros (renglones) tiene 8 bits. En cada bit podemos escribir/leer un “0” o un “1”.

La memoria de datos del PIC16F84 es de 8 bits.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Dir.	Memoria de datos
0	7 6 5 4 3 2 1 0
1	
2	
3	
4	
5	Puerto A
6	Puerto B
7	
8	
9	
10	
11	
12	
13	
14	
77	
78	
79	

Memoria de datos del PIC16F84
Fig. 9

Para referirnos a estos registros, también llamados localidades de memoria, lo podemos hacer por su dirección (número consecutivo) Pero es más fácil que se les ponga un nombre. Por ejemplo para referirnos al registro 57 seria preferible llamarle “Contador de Tiempo”. Así no tendríamos que recordar su dirección sino solamente su nombre. Esto se logra con una declaración que se coloca al principio del programa:

ContadorDeTiempo equ d’57’

Empecemos con los peros...

Lo siento, este lenguaje no admite espacios en el interior del nombre.

¡Ah¡ todos los números decimales se escriben entre apóstrofos y antecedido por la letra d. Por ejemplo 15 se escribe d’15’ ... 45 se escribe d’45’... así es este lenguaje.

El nombre del registro lo escoge el programador (mi estimado lector). Des este modo no tendrá que recordar el número de registro sino su nombre que esperemos este asociado con la función que tiene el registro en el programa.

Para referirnos a los bits de los registros seguiremos esta convención:

ContadorDeTiempo, 0

Se refiere al Bit0 del registro ContadorDeTiempo

ContadorDeTiempo,6

Se refiere al BIT6 del Registro ContadorDeTiempo

Otro ejemplo. Si declaramos al inicio del programa

CuentaPiezas equ d'13'

CuentaPiezas,4

Se refiere al BIT4 del registro CuentaPiezas de la memoria RAM

CuentaPiezas,2

Se refiere al BIT2 del registro numero 13, llamado CuentaPiezas

4.3.1.- Dos Registros muy Especiales

La memoria de datos (RAM) esta dividida en dos grupos: a las primeras 12 localidades se les llama "registros especiales" y al resto se les llama "registros de uso general". Fig. 9

Los registros especiales tienen usos muy particulares. Ya los veremos en otros artículos. Pero dos de ellos no podemos dejarlos para después: el numero 5 y el numero 6, que tienen que ver con el Puerto A y el Puerto B.

En Dirección 5 de la memoria de datos esta el Puerto A

No	No	No	RA4	RA3	RA2	RA1	RA0

En la Dirección 6 de la memoria de datos esta el Puerto B

RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

El fabricante hizo el PIC así. En la dirección 5 esta el Puerto A y el la dirección 6 esta el Puerto B.

Al igual que otras localidades de la memoria RAM, mediante la declaración equ podremos asignar un nombre al Puerto A o al Puerto B.(el nombre que le daremos a esta dirección en nuestro programa) Por ejemplo:

```
PortA        equ        d'5'
```

Quiere decir que el Puerto A se llamará PortA en nuestro programa

```
Reles_De_Salida    equ    d'6'
```

Quiere decir que el Puerto B se llamará Reles_De_Salida en nuestro programa. Claro que este sería un buen nobre adecuado si todos los pines del Puerto B estuvieran conectados a relevadores.

Hagamos la siguientes declaraciones al principio del programa

```
PuertoA      equ        d'5'
PuertoB      equ        d'6'
```

PuertoA,1
Se refiere a RA1 es decir el pin 17

PuertoB,7
Se refiere a RB7 es decir el pin 13

4.3.2.- Tres instrucciones para la RAM

Las tres instrucciones siguientes se aplican a los registros de la memoria de datos RAM (registros especiales y también de uso general)

Veamos el siguiente Ejemplo.

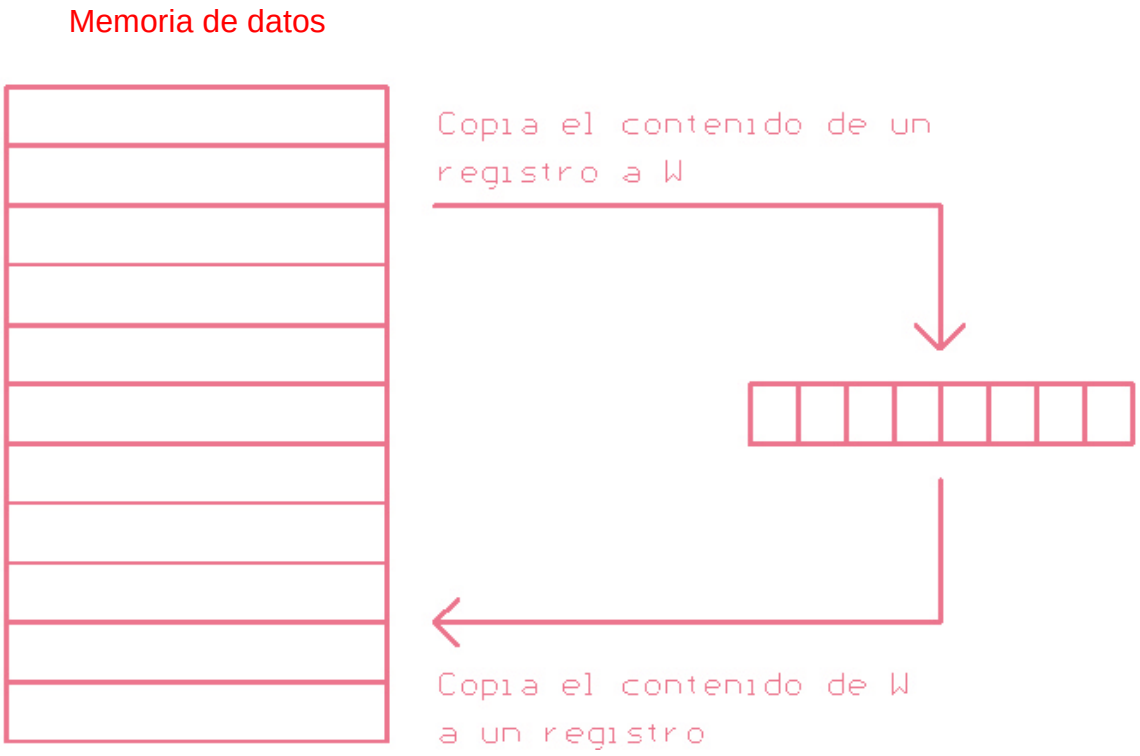
```
PortA        equ        d'5'
PortB        equ        d'6'
Contador     equ        d'13'

clrf         PortB      Pone en cero los 8 bits de Puerto B
                         RB7, RB6...RB0
```

clrf	Contador	Pone en cero los 8 bits de Contador
bsf	PortA,1	Pone en “1” RA1
bcf	PortB,7	Pone en “0” RB7

4.4.- El Acumulador

El acunulador es un Registro de 8 bits. También es llamado registro de trabajo (Work) se identifica con la letra W. Fig10.



El acumulador
Fig10

Como se puede ver en la Fig. 10 el acumulador se usa esencialmente para:

- Mover el contenido de un registro a otro
- Para inicializar un registro con un valor determinado
- Para realizar alguna operación lógica o aritmética.

Supongamos que queremos cargar el Puerto B con unos en sus ocho bits, tenemos que pasar forzosamente por el acumulador: “carga el acumulador con unos y mueve el contenido del acumulador al Puerto A”.

4.4.1.- Dos instrucciones para el acumulador.

Las dos instrucciones siguientes se aplican al acumulador y a la memoria de datos: movlw, movwf

```
PuertoA     equ         d’5’
PuertoB     equ         d’6’

clrf        equ         PuertoB

movlw       b’11111111’
movwf       PuertoB
```

b’11111111’ es la manera de representar el patrón de bits de un registro.Observe: los ocho bits entre comillas precedidos por la letra b.

movlw quiere decir: “carga en el acumulador el patrón de bits siguiente”

movlw	b’11111111’	quiere decir: “carga en el acumulador el patrón de bits b’11111111’ “

movwf quiere decir: “transfiere el contenido del acumulador al registro ...”

movwf	PuertoB	quiere transfiere el contenido del acumulador que era b’11111111’ al PuertoB

Claro que al final de estas dos instrucciones el Puerto B tendría “uno” en todos sus bits y si fueran salidas pues encenderían todos los pines de este puerto.

Otro ejemplo:

```
PuertoA     equ     d'5'
PuertoB     equ     d'6'
Contador    equ     d'13'

clrf        PuertoB

movlw       b'10101010'
movwf       PuertoB

movlw       b'00000001'
movwf       Contador
```

Al final de este programa tendríamos en PuertoB b'10101010' y en el Contador b'00000001'.

5.- Programando en Ensamblador.

5.1.- Un programa de una instrucción.

```
; Este es un programa de una instrucción
;-------------------------------------------------

        PuertoA     equ     d'5'
        PuertoB     equ     d'6'

        org         d'0'                ;define el origen

        movlw       b'00001111'         ;carga acumulador con
                                        ;b'00001111'

        End                             ;fin del programa
```

El punto y coma ; se utiliza para hacer comentarios a las instrucciones que vamos editando. El Compilador no las traduce, simplemente no las toma en cuenta. El punto y coma le indica al Compilador que lo que sigue en ese renglón no debe ser considerado

El ; es muy útil para hacer un encabezado o usándolo después de una instrucción para hacer un comentario sobre las intenciones de la misma. Un programa bien comentado será más fácil de entenderse.

La declaración org define la dirección de la memoria donde iniciamos a colocar las siguientes instrucciones.

```
        org         d'0'            ;define el origen
```

Quiere decir que las instrucciones que siguen serán gravadas en la memoria a partir de la dirección d'0' .

Parecería lógico que comencemos a escribir el programa a partir de la dirección d'0', de acuerdo, pero los PICS no dejan de ser maquinitas, has que explicarles todo con detalle, no presuponen nada.

End se usa para terminar el programa... org para iniciar y End para terminar.

5.2.- Un programa para encender algunas salidas

```
; Este es un programa de para encender algunas salidas
;-------------------------------------------------------------------------

        PuertoA     equ   d'5'
        PuertoB     equ   d'6'

        Org         d'0'
Inicio
        Movlw       b'00001111'
        Tris        PuertoB

PrendeSalidas
        movlw       b'11111110'     ;carga acumulador con
                                    ;b'11111111'
        movwf       PuertoB

        goto        Inicio

        End                         ;fin del programa
```

En este ejemplo, las palabras: Inicio y PrendeSalidas son Etiquetas, las usamos para mantener el programa bien documentado y como referencia para otras instrucciones que se coloquen más adelante. Las Etiquetas son como nombres del renglón. Siempre se escriben a partir de la columna 1.

En el renglón que sigue a la etiqueta Inicio, se instruye al PIC cuales pines del Puerto B deseamos como Entradas y cuales como Salidas.

Esto se logra mediante dos instrucciones:

```
        movlw   b'11111111'
        Tris    PuertoB
```

Tris PuertoB quiere decir: "define las Entradas / salidas del PuertoB según el patrón del acumulador".

Un "0" asigna unas Salida, Un "1" asigna una Entrada.

Entonces:

RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
0	0	0	0	1	1	1	1

RB7, RB6, RB5, RB4 se definen como Salidas.

RB3, RB2, RB1, RB0 se definen como Entradas.

Después de la etiquete PrendeSalidas encontramos:

```
        movlw   b'11111110'     ;carga acumulador con
                                ;b'11111110'
        movwf   PuertoB
```

Esto hace encender RB7, RB6, RB5, RB4, RB3, RB2, RB1, pero no RB0

Antes de la instrucción End encontramos la instrucción

```
goto            Inicio
```

La instrucción goto hace que el micro regrese a la etiqueta Inicio y repita las instrucciones en un bucle infinito.

6.- A practicar.

Hasta aquí la teoría. Es hora de practicar.

Tecleé en MPLAB el ejemplo anterior y los ejemplos que acompañan la tarjeta Test1.

Compile con MPLAB

Grave sus programas con Prog2

Pruebe sus programas en la tarjeta Test1

En el siguiente artículo pondremos manos a la obra y explicaremos con detalle esta lista de tareas, que usted puede intentar por sí mismo desde ahora.

También vamos a profundizar en el lenguaje Ensamblador y daremos muchos ejemplos más.

7.- Comentarios Finales a esta artículo.

Este escrito se comenzó diciendo que PIC viene de Programable Integrated Circuit, esta definición es muy buena porque explica con una gran exactitud de que se trata, muchos autores la usan y la dan por buena.

Pero otros autores aseguran que PIC viene de Periherical Interface Controler.

Bueno, lo importante es saber usarlos.

Picmicro es una marca registrada de Microchip.