Introducción a Los Microcontroladores RISC. En Lenguaje C
- PIC´s de Microchips -

Tito Flórez C.*

Resumen

A medida que el programa de los microcontroladores se hace más complejo, trabajar en lenguaje "assembler" se hace más dispendioso, difícil de manejar y el control de interrupciones muchas veces son un dolor de cabeza. Una muy buena alternativa para solucionar estos problemas, es usar el lenguaje C para programarlos. De esta forma, los programas se vuelven muy sencillos; lo mismo que el de interrupciones se convierte ahora en algo muy sencillo. Se presentan los elementos y las instrucciones más importantes para poder llegar a desarrollar un sin número de programas para los PICs.

Introducción

Aprender a manejar los microcontroladores desde el lenguaje "assembler" es muy fascinante, ya que de esta forma se aprende a conocer en forma muy profunda la arquitectura de los microcontroladores a la par que se aprende a manejar correctamente dicho lenguaje tan importante. Sin embargo, los problemas empiezan a aparecer cuando los programas aumentan de tamaño, y llegan a ser casi inmanejables cuando son de tamaños considerables. A lo anterior se une el problema que aparece cuando hay necesidad de manejar interrupciones, pues en el manejo de estas, a veces aparecen problemas, que solamente una persona con gran experiencia puede llegar a solucionarlos. Por las razones anteriores, es muy normal encontrar a las personas que trabajan con los microcontroladores, dedicando días y días enteros tratando de encontrar los errores del programa, no pudiendo dedicar tiempo al objetivo principal.

I. Lenguaje De Alto Nivel

Una forma de solucionar en gran medida los problemas anteriormente mencionados, es cambiando el lenguaje de programación "assembler" por uno de alto nivel. Actualmente uno de los lenguajes de alto nivel más atractivos para usar es el " C ", y es uno de los más difundidos actualmente; es un lenguaje estructurado y sirve para aplicaciones orientadas a objetos. Por estas razones, las casas productoras de *"software"*, han lanzado al mercado compiladores para microcontroladores en este lenguaje y que se presenta en el artículo como un medio de programación eficaz en los microcontroladores.

II. Instrucciones Claves Para Programar Pics En Lenguaje C

Existe un buen número de personas que trabajan con los microcontroladores y no saben lenguaje "C", pero sí saben otro lenguaje. Para estos, el tratar de aprender dicho lenguaje les llevará varios días de estudio concienzudo ya que el "C" aun cuando es un lenguaje de alto nivel y se usa mucho en nuestro medio, es también un lenguaje muy extenso y posee un gran número de opciones para poder ejecutar una misma cosa, esto dificulta el uso cotidiano del lenguaje y complica su aprendizaje.

A través de ejemplos, se muestra el uso de las instrucciones más importantes para programar PIC´s en lenguaje "C". Con dichas instrucciones se puede desarrollar casi la totalidad de los programas que se deseen.

A. Comentarios

```
// Si empieza con doble "slash", el comentario abarca una sola línea .

/*
  Si empieza con "slash asterisco" y terminan con "asterisco slash", puede  abarcar más
de una línea.
*/
```

B. Tipos de datos

```
int a,d;              // "a" y "d": enteros con signo. Rango de -128  a  +127.
unsigned int  b,c;    // "b" y "c" son enteros sin signo ( 8 bits en total)
bits    h, numero;    // "h" y "numero" son enteros y se pueden trabajar a nivel de
                      //      bits.
```

Las variables pueden se globales o locales, dependiendo del sitio en donde se haga su declaración.

C. Asignación

```
a = d ;      // En  "a" se copia lo que hay en  "d"
a = a + 1;   // El valor de "a" se incrementa en "1"
a = a - 3 ;  // El valor de "a" se decrementa en "3"
             // Es importante notar que las sentencias terminan con punto y coma " ; "
             // Las letras minúsculas se toman diferentes a la mayúsculas, o sea que la
             //       variable "a" es diferente a la variable "A" .
```

*Departamento de Ingeniería de Sistemas.Universidad Nacional. Bogotá.

D. Bloques

Cuando se desea referirse a un grupo de instrucciones como a una unidad, estas pueden especificarse fácilmente encerrándolas entre llaves. Ejemplo:

```
{
   a = 8;
   b = a + 10;
   t = t + 1;
}
```

E. Toma de decisiones

1. If

Para hacer toma de decisiones es necesario saber las siguientes convenciones:

Y lógico (**AND**) se representa con	" && "
Ó lógico (**OR**) se representa con doble barra vertical	" || "
La **negación** se representa con el cierre de admiración	" ! "

Para preguntar por:

Igual a ... se usa el **doble igual**	" == "
No igual a ... se usa **negación** y **sencillo igual**	" != "
Mayor que se usa	" > "
Mayor o igual que se usa	" >= "
Menor que se usa	" < "
Menor o igual que se usa	" <= "

Si "p" es igual a "1", **y** "valor" es menor o igual que 5, entonces se realiza el bloque:

```
            total = total + 2;
            a     = a + 7;

if (  (p == 1) && (valor <= 5) )
{
             total = total + 2;
             a     = a + 7;
}                          // end del if
```

Si "d" es igual a "0", **ó** "b" mayor o igual a 7 entonces realizar el bloque:

```
                    t = t+1;
                    a = a + t;
```

En caso contrario realizar el bloque:

```
                    t = t+9;

if (  (d == 0) || (b >= 7)  )
{
       t = t + 1;
       a = a + t;
}
else
{
       t = t + 9;
}            // end del  if
```

2. Switch

Cuando existe una variable, y según el valor poseído hay que tomar sus correspondientes decisiones, usar varios o muchos "if" puede quitarle claridad al programa. En este caso, usar el "switch" es muy buena alternativa para dar claridad al programa. El siguiente ejemplo muestra claramente su uso.

```
Switch(valor)
{
   case 0: a = a + 1;     // si valor = 0, entonces a = a + 1.....
           b = 3;         // y  b = 3
           break;         // Salir del switch.

   case 1: a = a + 9;     // si valor = 1, entonces      a = a + 9
           b = 8;         //                             b = 8
           t = 3;         //             y               t = 3
           break;         // Salir del switch.

   case 5: b = 2;         // si valor = 5, entonces b = 2
           break;         // Salir del switch.

   default:a = 0;         // Si valor no es ninguno de los anteriores,
           b = 3;         //   entonces se hace "a = 0" y "b =3"
           break;         // Salir del switch (en este caso,  no es
                          //   necesaria, por su posición.
}                         // end  del swich
```

F. Ciclos

Cuando se desea realizar una serie de ciclos, se puede escoger alguna de las siguientes estructuras. Sabiendo usar bien una sola de estas, es posible trabajar sin las demás. En esencia los tres ejemplos siguientes son equivalentes.

1. while

Para realizar una serie de ciclos **Mientras** que una condición se cumpla, se puede usar el "while".

La condición se evalúa antes de empezar cada ciclo, esto significa que el ciclo puede que nunca se realice si al tratar de entrar por primera vez, la condición no se cumple.

Mientras que "a" sea menor o igual que 9 realizar el siguiente ciclo (empezar con a = 2)

```
a = 2;              // Normalmente antes de entrar al ciclo se dan los valores de inicio.
while ( a <=9)
{
      ...
      ...
      ...           // en alguna parte se varía  "a"  . Ejemplo:   a = a + 1;
}                   // end while
```

Si se desea hacer un ciclo infinito, se puede realizar de la siguiente forma, teniendo en cuenta que lo que hay en el paréntesis se evalúa, y si es "0" se interpreta como falso, en caso contrario se interpreta como verdadero. En el presente caso "1" se interpreta como verdadero.

```
while ( 1 )
{
       ...          // Ciclo infinito
                    ...
}                   // end while
```

2. FOR

Para un valor inicial de "a = 2", realizar ciclos **Mientras** que "a sea <= 9", y una vez "terminado" cada ciclo, incremente "**a**" en 1 (a = a + 1). Esta instrucción se comporta exactamente igual a la anterior "while", o sea que la condición se evalúa antes de empezar cada ciclo, y puede significar que el ciclo nunca se realice, si al iniciar la instrucción, la condición no se cumple.

```
for( a=2;  a <= 9;  a = a+1 )
{
                ....
                ...
}               // end del   for ( a ...
```

3. Do { .... } While

Cuando se desea que un ciclo se realice **al menos una vez** y que se siga realizando ciclos **Mientras** se cumpla una condición, se puede pensar en el "do". Lo anterior indica que la evaluación de la condición se realiza al final de cada ciclo.

```
a = 2;          // Normalmente antes de entrar al ciclo se dan los valores de inicio.
do
 {
                ...
                ...
        ... // en alguna parte se varía "a". Ejemplo: a = a + 1;
 } while ( a <= 9 );
```

G. Ruptura del ciclo

1. BREAK

Para salir de un "while", un "for", un "switch" o de un "do" se puede usar la instrucción " **break** ; ". Es importante notar que en los casos en los cuales existe anidamiento de estructuras, el "break" transfiere el control solamente un nivel hacia afuera.

2. GOTO IDENTIFICADOR;

Hace que el programa salte al rótulo ("label") **"identificador".**

```
goto Otra_Vez;      // Va a la instrucción cuyo rótulo es Otra_Vez.
```

H. Funciones

Es muy importante cuando se trabaja en C++, que para el paso de parámetros, una función puede:

- No utilizar parámetros.
- Utilizar parámetros por valor (llamados también por copia). En este caso, el valor del parámetro de entrada es el mismo que el de salida, no importa si dentro de la función llamada se alteró este valor (aparentemente).
- Utilizar parámetros por referencia o dirección. En este caso, el valor del parámetro de salida puede ser cambiado desde función llamada.

III. El Pic 16c84

Dentro de los microcontroladores, hay un grupo en auge conocido con el nombre de PICs, que poseen características RISC (Computadoras con una Serie Reducida de Instrucciones). Esta serie de microcontroladores son similares en muchos aspectos, uno de ellos es la gran similitud en su arquitectura, ya que su programación en lenguaje C, es en esencia idéntica. Este hecho facilita enormemente poder llegar a manejar cualquiera de esto PICs, conociendo como funciona uno de ellos.

El ejemplo a desarrollar se realiza sobre el PIC 16C84 ya que es fácil de conseguir, es de costo reducido (cerca de 4 dólares), es un buen prototipo dentro de los PIC, y posee memoria EEPROM (Memoria ROM Borrable Eléctricamente) que lo hace apto para programarlo directamente una y otra vez, sin tener que estar sometiéndolo a la acción del borrador de luz ultravioleta, lo cual exige tener el borrador y poseer la paciencia y tiempo que esto conlleva. De forma semejante, particularizamos con el MPC *Code Development System (Optimizing C Compiler for PIC16/17Cxx).*

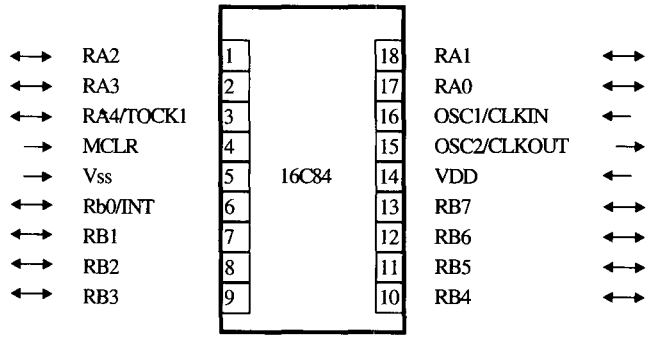

Fig. 1. Configuración de los pines del 16C84

A. Sistema de desarrollo

Para programar el PIC en lenguaje C, se debe poseer un computador, un compilador de **C** (ejemplo el MPC) y el sistema de desarrollo (*Development System*) para los Microcontroladores PIC16/17. Dicho sistema cuesta cerca de 200 dólares, es relativamente fácil de conseguir, y viene con:

- Un programador (también conocido como "quemador"). Actualmente existe un programador universal, el cual es muy versátil. También se puede conseguir versiones anteriores no universales, en tal caso asegurarse que sea el PICSTART-16B1, ya que existen otras referencias como el PICSTART-16C que sirven solamente para otros PIC. tales como el PIC16C64 y el PIC16C74.

- Un cable para conectar entre el programador y el puerto serial.

- Un adaptador de potencia para conectar al programador.

- Su respectivo *Software*. La instalación del *Software* en el computador, no presenta ningún problema, y se hace de forma similar a la instalación de cualquier programa. Dicho *Software* posee generalmente 3 módulos principales, de los cuales por el momento nos interesa solamente el MPS16B.EXE, que tiene por función coger el programa que se le indique (nombre.HEX) y controlar al programador ("quemador") para que este programe el respectivo PIC.

- Los manuales. Dentro de estos pueden venir unos 5, pero momentáneamente nos interesa solamente el MICROCHIP DATA BOOK.

Hace relativamente poco tiempo salió al mercado una versión mejorada conocida como la "PICSTART PLUS" cuyo *software* es más amigable, el programador sirve para todos los PICs, y casi todos los manuales vienen en un CD.

B. Información Clave

Leer cuidadosamente los manuales anteriores requiere mucho tiempo, pues existe en ellos una gran cantidad de información. Para alguien que se esté iniciando en este tema, no es muy aconsejable, muy seguramente quedará desorientado dentro de un mar de información. Por tal razón, es mejor aprender primero la cosas fundamentales para su correcto manejo.

Para empezar tomar el MICROCHIP DATA BOOK. Una vez en este, ubicarse en la sección: *8-Bit Microcontroller Product Specifications*, y dentro de esta, ubicarse en las páginas correspondientes al PIC16C84 (si se posee el CD, ubicarse en el PIC16C84).

Dentro de estas páginas existe información clave que se debe poseer para llegar a manejar los PIC. de forma sencilla y confiable. Dicha información es la siguiente:

1. Características Eléctricas.

Esta información se debe tener en cuenta para no llegar a dañar el PIC. y asegurarse de su correcto funcionamiento. Dentro de estas las más importantes son:

a. Voltaje de funcionamiento (Vdd) va de 4 a 6 voltios (usar 5 voltios es muy atractivo; es un voltaje fácil de obtener y es compatible con los circuitos TTL).

b. Máxima corriente que puede recibir cualquier pin es de 25 miliamperios.

c. Máxima corriente que puede recibir en total el puerto A es de 80 miliamperios.

d. Máxima corriente que puede recibir en total el puerto B es de 150 miliamperios.

e. Máxima corriente que puede dar cualquier pin es de 20 miliamperios.

f. Máxima corriente que puede dar en total el puerto A es de 50 miliamperios.

g. Máxima corriente que puede dar en total el puerto B es de 100 miliamperios.

Lo anterior significa que siempre que un pin sea escogido para salida, se debe estar atentos a no manejar corrientes que excedan las especificadas, y un método, para empezar, es colocando una resistencia a la salida de dicho pin (un buen valor puede ser usar una resistencia de 1000 ohmios a ¼ de Watt en cada pin de salida).

2. Registros del Sistema.

Es importante manejar adecuadamente los siguientes registros, los cuales aparecen en una de las figuras del manual, bajo el nombre de REGISTER FILE MAP. También aquí, aparece sus direcciones con sus respectivas páginas, las cuales no nos interesa por ir a trabajar desde un lenguaje de alto nivel.

a. TRISA. Se usa para definir los pines que serán de entrada y los pines que serán de salida, obviamente con relación al puerto A. Si un bit lo ponemos a "1", el pin asociado con dicho bit será de entrada, y si el bit lo ponemos a "0", dicho pin asociado será de salida.

b. TRISB. Se usa para definir los pines que serán de entrada y los pines que serán de salida, obviamente con relación al puerto B. Si un bit lo ponemos a "1", el pin asociado con dicho bit será de entrada, y si el bit lo ponemos a "0", dicho pin asociado será de salida.

c. PORTA (Puerto A). En él están los datos que están entrando por los pines del puerto A, y en él se colocan los datos que se desea salgan por los pines del puerto A.

d. PORTB (Puerto B). En él están los datos que están entrando por los pines del puerto B, y en él se colocan los datos que se desea salgan por los pines del puerto B.

e. INTCON (Registro de interrupciones) es útil cuando se van a manejar interrupciones.

3. Registros de Propósito General (SRAM).

De ellos nos interesa que son de 8 bits y un total de 36 para el PIC16C84, mientras que el PIC16F84 posee un total de 68.

EJEMPLO

```
// ============================================================

/*
El siguiente programa suma dos números binarios, cada uno de tres bits.
El primer numero entra por los dígitos menos significativos del puerto A,
  es decir por RA0, RA1 y RA2.
El segundo por los siguientes tres bits del puerto B: RB1, RB2 y RB3.
  (NO se empieza desde RB0/INT, el cual se deja para meter la interrupción).
La suma (que será máximo de 4 dígitos) se sacara por los 4 dígitos más
  significativos de puerto B (RB4-RB7).
Cuando produzcamos una interrupción por flanco de bajada en el
  pin RB0/INT, si el valor del primer numero (dato1) es mayor o igual que 5, se
  sacara un "1" por RA3, en caso contrario se sacara un "0".
*/

// ============================================================

#include <16c84.h>

void inicializar(void);


// ============================================================

// variables globales

unsigned int suma;     // entero sin signo (rango:  0 a 255).
bits  dato1, dato2;    // bits: permite trabajar a nivel de bit.
// ============================================================
// ============================================================

void main(void)
{
    inicializar();
    while(1)
    {
      dato1.0 = PORTA.0;          // Se pasan los 3 dígitos del primer...
      dato1.1 = PORTA.1;          // numero (Puerto A. Del RA0-RA2)...
      dato1.2 = PORTA.2;          // a dato1 (dato1.0 - dato1.2)

      dato2.0 = PORTB.1;          // OJO. Se pasan los 3 dígitos del ...
      dato2.1 = PORTB.2;          // segundo número (Puerto B. Del RB1-RB3)...
      dato2.2 = PORTB.3;          // a dato2 (dato2.0 - dato2.2).

      suma  = dato1 + dato2;
      suma <<= 4;                 // Se rota 4 posiciones a la izq. para
                                  // alinear con RB4-RB7 (salida).
      PORTB = suma;               // Se saca la respuesta.
    }                             // Fin del while.
return;
}

// ============================================================

// ============================================================

void __INT(void)          // Nombre con que viene la función de interrupción.
{
    SaveContext;          // Evite problemas: SIEMPRE salvar registros del ...
                          // sistema al entrar a un programa de interrupción.
    INTCON.INTF = 0;      // Al ocurrir una interrupción por RB0/INT, el bit#1...
                          // de INTCON (INTF), se pone en "1", el cual hay ...
                          // que colocarlo en "0" antes de salir de este ...
                          // subprograma. De no hacerlo, al salir, se vuelve ...
                          // a entrar inmediatamente al presente subprograma, ...
                          // y no se podrá regresar al punto correcto para ...
                          // continuar la tarea interrumpida. Algo análogo ...
                          // sucede con los otros tipos de interrupciones.
    if (dato1 >= 5)
    {
      PORTA.3 = 1;
    }
    else
    {
      PORTA.3 = 0;
    }                     // end del if

    RestoreContext;       // Se restauran registros salvados.
return;
}
// ============================================================

// ============================================================

void inicializar(void)
{
    INTCON.INTE = 1;      // Se permite interrupción por el pin RB0/INT
                          // INTerrupt Enable bit (bit # 4 de INTCON).

    INTCON.GIE  = 1;      // Se activa el sistema para que empiece a...
                          // recibir interrupciones.
                          // GIE = Global Interrupt Enable (bit # 7).

    OPTION.INTEDG = 0;    // Bit # 6 (INTerrupt EDGe) de OPTION = 0.
                          // La interrupción ocurre cuando...
                          // la información que entra ...
                          // por RB0/INT pasa desde un "1"a "0" ...
                          // o sea que es por flanco de bajada.

                          // En TRISA y TRISB se definen los pines de....
                          // entrada y los pines de salida.
    TRISA   = 0b00000111; // RA0-RA2 entrada de "dato1" .(0 son salidas).
                          //    RA3 salida para indicar si "dato1 >= 5".

    TRISB   = 0b00001111; // RB0/INT entrada de interrupción.
                          // RB1-RB3 entrada de "dato2". (0 son salidas).
                          // RB4-RB7 salida de "suma".

    dato1   = 0;          // Empezamos con estas variables en  "0".
    dato2   = 0;
    suma    = 0;
return;
}

// ============================================================
// FIN
// ============================================================
```

C. COMPILAR EL PROGRAMA

Aun cuando pueden haber varios compiladores para el lenguaje **C**, para poder continuar con el ejemplo, se hace necesario escoger uno de ellos, y arbitrariamente se escoge el MPC, que es un ambiente integrado de desarrollo para lenguaje **C**.

Una vez escrito en solo texto el programa, cuyo nombre debe terminar en ".C" (nombre.C), se corre el programa IDE (*Integrated Development Environment*) y se procede preferiblemente en la siguiente secuencia, para dar la información solicitada por el IDE.

1. Se escoge "Option Selection" (generalmente se escoge solamente la primera vez que se entra al IDE) y dentro de esta se puede escoger:

a. "Hex Dump File" la opción "INHX8M"

b. "Source Extension" la opción " .C "

c."List File" la opción "Yes". Esta opción es muy conveniente; el archivo es muy útil para detectar errores.

2. Escoger "Load New File" y dentro de esta darle el nombre del archivo que se hizo en lenguaje C (nombre.C)

3. Finalmente escoger la opción " Assemble / Compile " la cual automáticamente nos produce los archivos escogidos en "Option Selection" . Dichos archivos, en este caso particular serán:

a. "nombre.HEX" : es el que sirve para programar el PIC.

b. "nombre.LST" : importante para encontrar errores, el cual puede ser analizado por el programador, cuando lo llame desde el Editor o procesador de palabra.

D. Programar (¨Quemar ¨) El Pic

Antes de proceder a programar ("quemar") el microcontrolador, verificar que el PIC esté correctamente situado en el programador, ya que si se usa el:

1. PICSTART-16B: debe colocarse justificado hacia el extremo derecho.

2. PICSTART PLUS: debe colocarse justificado hacia el extremo izquierdo.

Para programar el PIC, se corre un programa como el MPS16B.EXE, el cual inicialmente trata de hacer conexión con el programador a través del puerto serial COM1 del computador. Si el COM1 está ocupado, colocarlo en COM2 - COM4 y suministrar esta información, seleccionando del menú: "Options" y después "*Comm Port Selection*" (si no se posee sino el COM1, es necesario desconectar el "ratón", y en su lugar conectar el programador).

Si no se tiene el "ratón", el movimiento a través de este menú, se puede hacer con la teclas de "Tabulación", "las flecha" y el "espaciador", este último sirve también para escoger la opción deseada.

Una vez dentro del menú:

1. "**D**evice edit": para escoger el PIC se desea programar (en el ejemplo a desarrollar se escoge el 16C84).

2. "fuse **E**dit" :

a. "**LP**", "**XT**" o "**HS**" : si se va a usar un cristal como oscilador (dependiendo de su frecuencia de oscilación).

b. "**RC**" : si se va a usar como oscilador una Resistencia (Rext) y un Condensador (Cext). Un buen valor para la resistencia puede estar entre 3 kohm. y 100 kohm. y para el condensador, valores mayores de 20 picofaradios.

c. "**W**atchdog Timer On" : cuando se vaya a usar el"Watchdog", la forma de seleccionarlo es colocando la "X" es su respectivo renglón (como en nuestro ejemplo no se usará el "Watchdog", anulamos la "X").

3. "**F**ile" : en él se escoge el archivo deseado "nombre.HEX".

4. "**P**rogram" : en él se programa finalmente el PIC.

Si lo que se posee es el sistema de desarrollo PICSTART PLUS, el cual es diseñado para correr desde Windows, la información anterior se suministra de forma más cómoda así:

1. Ejecutar el **MPLAB.EXE,** haciendo doble clic sobre este.

2. Del menú principal, seleccionar **Picstart Plus.**

3. Seleccionar **Enable Programmer.** Si sale un aviso que dice "PICSTART PLUS not found on COM1", verificar que el cable que va al computador esté en COM1, y que las demás conexiones estén correctas.

4. Aparece una ventana que dice "PICSTART PLUS Device Programmer", la cual hay que llenarla correctamente.

5. Del menú principal escoger **File**.

6. De este, **Import.**

7. Luego **Download To Memory.**

8. Escoger el programa que desea grabar en el PIC, el cual debe terminar en **.HEX.**

9. Una vez verificado que el PIC esté correctamente montado sobre el quemador, regresar a la ventana cuyo nombre es "PICSTART PLUS Device Programmer", y dar clic sobre **Program.**

10. Esperar algunos segundos a que se termine la quemada, y retirar el **PIC.**

E. Montaje Del Pic

Una vez montado en el "Protoboard" el PIC16C84, verificar que:

1. El pin # 5 vaya a 0 voltios (Vss)

2. Los pines # 4 y #14 vayan al voltaje de alimentación Vdd.

3. Si se va a trabajar con RC, del pin # 16 debe salir la resistencia Rext hacia el voltaje Vdd. y el condensador Cext hacia 0 voltios.
   Si no se va a trabajar con RC, remitirse al manual a *"Oscillator Configurations"*.

4. Para este ejemplo en particular, si se van a usar LEDs para detectar los bits de salida, cerciorase primero que el pin # 2 correspondiente a RA3, y los pines # 10, #11, #12 y #13, que corresponden a RB4-RB7 salen sus respectivas resistencias de 1 kohm. hacia los LEDs.

Conclusiones

- Se justifica plenamente aprender a programar los microcontroladores desde un lenguaje de alto nivel, ya que ello le permite al programador un mayor control sobre su programa.

- El manejo de las interrupciones se hace de manera más clara y presenta menos problemas cuando se realiza desde un lenguaje de alto nivel que desde el "assembler".

- Si bien es cierto, que trabajar desde un lenguaje de alto nivel tiene sus ventajas, también es cierto que esto hace que el programador no necesariamente deba tener un conocimiento claro de la estructura interna del microcontrolador.

- Usar lenguaje de alto nivel para el manejo de los microcontroladores, permite trabajar de forma sencilla con los microcontroladores. Esto colabora enormemente a que el manejo de los microcontroladores cada día sea menos exclusivo del personal con gran conocimiento en sistemas y electrónica.

- Un individuo con algunos conocimientos en programación, puede perfectamente empezar a manejar los microcontroladores.

Bibliografía

1. AGUILAR Luis Joyanes, ***Curso de Programación con Turbo C++,*** McGraw-Hill / Interamericana de España, S.A., 1996.

2. MPC ***Code Development System, An Integrated Development Environment for C,*** 1996 Byte Craft Limited, Ontario Canada.

3. MICROCHIP DATA BOOK, 1994 *Microchip Technology* Inc.,USA.

4. PICSTART-16B1 USER'S GUIDE, 1994 *Microchip Technology* Inc.,USA.

5. PICSTART PLUS, DEVELOPMENT SYSTEM USER'S GUIDE, 1996 Microchip Technology Inc.,USA.