Programación y diseño de dispositivos mediante Microcontroladores PIC.

Dr. Eugenio Martín Cuenca
Ing. José María Moreno Balboa
Facultad de Ciencias. Universidad de Granada.
E-mail:emartin@goliat.ugr.es

Este mes lo dedicaremos a realizar algunos ejercicios con el compilador **PBC** y el compilador gratuito **LET BASIC** de la empresa *Leading Edge Technology*. Para dicho propósito, se empleará el módulo-01 y el módulo de aprendizaje tal y como se muestra en la foto 1. El lector aficionado al soldador, puede realizar su propio montaje siguiendo las explicaciones que se han dado en los anteriores artículos.

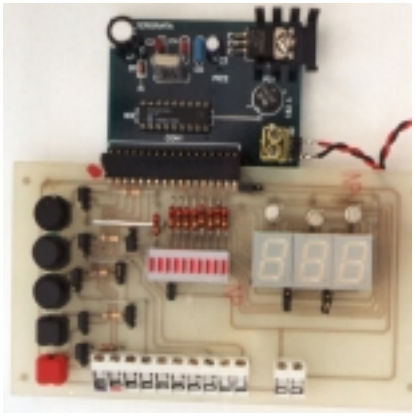


Foto 1. - ***Módulo-01*** *conectado al* ***Módulo de Aprendizaje*** *mediante el conector de 17 pines CN1 ejecutando el ejemplo número 4.*

Pero como lo prometido es deuda, el mes que viene comenzaremos la descripción del robot hexápodo movido por tres servo motores (foto 2).

Foto 2.- *Componentes del KIT hexápodo y robot terminado.*

Seguimos animando a los lectores a que nos escriban, y estamos tratando de dar respuesta a sus consultas a través de la página Web (http://curtis.ugr.es) dentro de nuestras posibilidades de tiempo.

EJERCICIOS CON EL COMPILADOR BASIC PBC.

Las explicaciones que se dan a continuación, parten de la premisa de que usted dispone del grabador ***MultiPIC*** u otro programador para el PIC16F84 y del compilador PBC.

En este primer ejercicio vamos a tratar de aclarar algunos conceptos y a comprobar la existencia de distintos caminos para realizar la misma función, utilizando para ello, diferentes comandos del compilador PBC.

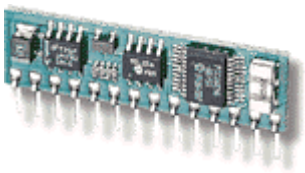


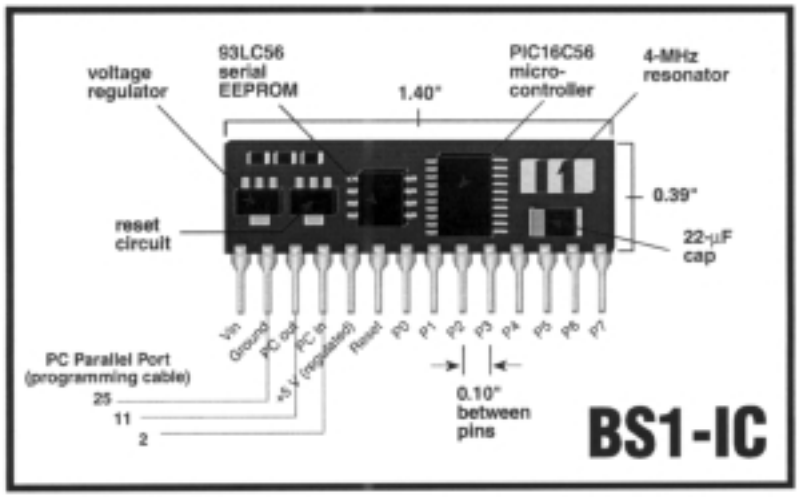
Figura 3. – *Módulo BS1-IC y descripción del patillaje.*

Como ya se explicó en el primer artículo (Julio/Agosto 1999), el compilador PBC deriva del interprete PBASIC diseñado para el módulo BS1-IC de la empresa *Parallax* (Figura 3 - foto 3).

Por lo anteriormente dicho, el compilador PBC ha heredado alguna idiosincracia al respecto. Aunque este permite programar una gran variedad de tipos de microcontroladores PIC que incorporan puertos de entrada y salida diferentes al PortB (PortA, PortC, PortD), sin embargo conserva las denominaciones del módulo BS1-IC.

El módulo BS1-IC (figura 3) tiene el Puerto B numerado como P0 a P7. Para el interprete PBASIC, Pin0 significa PORT B0, Pin1 PORT B1 y así sucesivamente. Además las instrucciones del interprete PBASIC hacen referencia al número del pin en los comandos HIGH y LOW, con solo indicar el numero del mismo que va de 0 a 7. Así, por ejemplo HIGH 3 significa poner al nivel lógico 1 el Pin 3, (es equivalente a HIGH Pin3, aunque si se escribe la instrucción de esta forma se produciría un error). Sin embargo al carecer el BS1-IC de otros puertos que no sean el Puerto B, los demás no se tuvieron en cuenta en el diseño del interprete PBASIC.

El compilador PBC si ha tenido en cuenta los demás puertos que puede poseer el microcontrolador PIC con



el que se esté trabajando. Para ello, se han ampliado los comandos del PBC incorporando las instrucciones PEEK y POKE, pero a su vez ha conservado la compatibilidad con el interprete PBASIC, por lo que cuando se indique Pin1 solo se refiere al PORT B1 y no a ningún otro puerto.

El siguiente ejemplo muestra lo que se acaba de exponer.

```
Comienzo:   LOW   0      ' configura el Pin0 como salida y lo pone a nivel 0
            HIGH  1      ' configura el Pin1 como salida y lo pone a nivel 1
'
Bucle:      PAUSE 500    ' espera 0.5 segundos
            TOGGLE 0     ' cambia el estado del Pin0. Si está a nivel bajo lo
                         ' pasa a nivel alto y viceversa
            TOGGLE 1     ' Idem con Pin1
            GOTO  Bucle  ' vuelve a bucle y repite de forma continua
```

Como en nuestros ejemplos, ya sea con un montaje realizado por el lector, ya sea mediante el Módulo de Aprendizaje, el Puerto B se encuentra conectado a diodos electroluminiscentes LEDs, se puede modificar el código anterior para una mejor comprensión por parte del programador, empleando el comando Symbol. Así, al pin 0 del PortB conectado al primer LED, se le asigna la denominación simbólica de LED1 y al pin 1 del PortB conectado al segundo LED la de LED2.

El siguiente programa realiza la misma función que el del mes pasado pero con diferentes comandos.

```
' ****************************************************************************
'  Programa : ART002.BAS              Fecha : 15 / 09 / 1999                 *
' ****************************************************************************
' *       Enciende y apaga alternativamente dos diodos LEDs                  *
' *      Al igual que el programaART001.BAS, se hace parpadear dos diodos LEDs *
' *       pero se utilizan otras instrucciones del compilador.               *
' *       Revisión : 1.0                Programa para PIC16F84               *
' *       Velocidad de reloj : 4 MHz    Reloj de instrucción : 1µs           *
' *       Reloj : Tipo XT               Perro Guardián : Off                 *

' ****************************************************************************
Symbol LED1=0                          ' Nombre del pin PB0 = LED1
Symbol LED2=1                          ' Nombre del pin PB1 = LED2

'
Comienzo:   LOW   LED1                 ' configura el pin como salida y apaga el LED1
            HIGH  LED2                 ' configura el pin como salida y enciende el LED2

'
Bucle:      PAUSE 500                  ' espera 0.5 segundos
            TOGGLE LED1                ' cambia el estado del LED1. Si está a nivel bajo lo
                                       ' pasa a nivel alto y viceversa
            TOGGLE LED2                ' Idem con LED2
            GOTO  Bucle                '  vuelve a bucle y repite de forma continua
```

HIGH Pin

La instrucción HIGH programa automáticamente el pin especificado como salida y lo coloca además a nivel alto.
Pin es una variable o una constante que especifica el pin de E/S, cuyo valor está comprendido entre (0 – 7) .

Puede imaginar esta instrucción como la equivalente de:

```
OUTPUT 3          ' Programa Pin 3
                  'como salida
LET PIN 3 = 1     ' Coloca el Pin 3
                  'a nivel alto
```

Observe en este pequeño ejemplo que el comando OUTPUT acepta el número de pin (3), mientras que el comando LET necesita el nombre de la variable *pin3*. Con el uso del comando HIGH se necesita, como ha comprobado, una instrucción menos.

Mencionamos aquí, un error muy común de los programadores en el empleo del comando HIGH. Normalmente estos sustituyen el nombre del pin, como pin3 por el número de pin. Recuerde que los nombres de los pines son realmente variables tipo bit. Como bits, estas solo pueden almacenar los valores 0 y 1. El comando HIGH pin3 es una instrucción BASIC válida, pero recuerde " Coge el estado de pin3. Si pin3 es 0, establece *pin 0* como salida a nivel alto. Si pin3 es 1, programa *pin 1* como salida y lo coloca nivel alto."

LOW Pin

La instrucción LOW programa automáticamente el pin especificado como salida y además lo coloca a nivel bajo.
Pin es una variable o una constante que especifica el pin de E/S, cuyo valor está comprendido entre (0 – 7) .

Puede imaginar esta instrucción como la equivalente de :

```
OUTPUT 3          ' Programa Pin 3
                  ' como salida
LET PIN 3 = 0     ' Coloca el Pin 3
                  ' a nivel bajo
```

TOGGLE Pin

Programa el pin referenciado como salida y conmuta su estado.
Pin es una variable o una constante que especifica el pin de E/S, cuyo valor está comprendido entre (0 – 7).

OPERACIONES MATEMÁTICAS

Con PBC pueden realizarse las siguientes operaciones matemáticas:

Estas operaciones pueden realizarse con constantes numéricas (números) o variables tipo byte o word. El compilador PBC utiliza matemática de enteros y sólo con enteros positivos. No puede emplear números negativos o números con punto decimal. Estas limitaciones sin embargo, no constituyen un serio problema y es posible trabajar con operaciones matemáticas completas redondeándolas.

Cuando trabaje con variables, recuerde los limites del tipo de variable que esté empleando. Por ejemplo el mayor número que una variable tipo byte (B0, B1, etc) puede almacenar es 255 mientras que en las variables tipo word es 65535 (W0, W1, ..).

```
+          Suma.
b1 = 2 + 4          ' Suma dos números ( la suma debe ser menor o igual a 255)
b 2 = b4 + b6       ' Suma dos variables tipo byte ( la suma debe ser ≤ a 255)
w3 = w4 + b3        ' Suma una variable tipo word y otra byte ( suma de ser ≤ 65535 )
w3 = w5 + w7        ' Suma dos variables tipo word (suma de ser ≤ 65535)
-          Resta.
*          Multiplica y devuelve los 2 bytes menos significativos de un resultados de 4 bytes.
           B4 = 12 * 10      ' El resultado (120) es almacenado en B4 (variable de 1 byte)
           W4 = 30 * 30      ' El resultado (900) es almacenado en W4 (variable de 2 bytes)
           W4 = 500 * 200 ' Los dos bytes menos significativos del resultado (100.000) son
                             'almacenados en W4 (variable de 2 bytes). En este caso el resultado
                             '100.000 ( número de 4 bytes) cuyo valor de los dos bytes LSB es 34464
                             '( el resto de dividir 100.000 / 65536).
**         Multiplica y devuelve los 2 bytes más significativos MSB de un resultado de 4 bytes.
           B4 = 12 ** 10     ' El resultado (120) es almacenado en B4 (variable de 1 byte)
           W4 = 30 ** 30     ' El resultado (900) es almacenado en W4 (variable de 2 bytes)
           W4 = 500 ** 200' Los dos bytes más significativos del resultado (100.000) son almacenados
                             'en W4 (variable de 2 bytes). En este caso el resultado 100.000
                             ' ( número de 4 bytes) cuyo valor de los dos bytes más significativos es 1
                             ' ( el cociente de dividir 100.000 / 65536).
/          Divide y devuelve el cociente.
           B4 = 9 / 2        ' El cociente (4) es almacenado en B4
//         Divide y devuelve el resto.
           B4 = 9 / 2        ' El resto (1) es almacenado en B4
MIN        Guarda el valor seleccionado en la variable si el resultado es menor o igual que este.
           B4 = B5 – B6 min 10    ' Realiza la resta y almacena el resultado en B4, pero si el resultado
                                  ' es menor de 10, entonces almacena 10 en B4
MAX        Guarda el valor seleccionado en la variable si el resultado es mayor o igual que este.
           B4 = B4 max 10         ' Si B4 es menor de 10, es almacenado su valor,  pero si es mayor
                                  ' de 10, B4 toma el valor 10
&          Realiza una operación lógica AND bit a bit.
           B4 = %10111001 &  %11110011      ' Realiza una operación lógica Y (AND) y almacena
                                            ' el resultado en B4
|          Realiza una operación lógica OR bit a bit.
           B4 = %10111001 |  %11110011      ' Realiza una operación lógica O (OR) y almacena
                                            ' el resultado en B4
^          Realiza una operación lógica XOR bit a bit.
           B4 = %10111001 ^  %11110011      ' Realiza una operación lógica  XOR y almacena
                                            ' el resultado en B4
&/         Realiza una operación lógica NAND (NO AND) bit a bit.
           B4 = %10111001 &/  %11110011     ' Realiza una operación lógica  NAND (no AND)
                                            'y almacena el resultado en B4
|/         Realiza una operación lógica NOR bit a bit.
           B4 = %10111001 |/  %11110011     ' Realiza una operación lógica  NOR (no OR)
                                            'y almacena el resultado en B4
^/         Realiza una operación lógica NXOR bit a bit.
           B4 = %10111001 ^/  %11110011     ' Realiza una operación lógica  NXOR (no OR)
                                            'y almacena el resultado en B4
```

TERCER PROGRAMA EJEMPLO

Hasta el momento solo se han realizado operaciones de salida, encendiendo o haciendo parpadear un diodo LED. En este tercer programa ejemplo, se realizaran operaciones de entrada de información (pulsado de un botón), análisis de la misma y respuesta (salida) encendiendo diodos LEDs.

POKE Dirección, Valor

Este comando escribe *Valor* en el registro del PIC especificado en *Dirección*. Esta instrucción es propia del compilador PBC y no se encuentra en el interprete PBASIC. Como se ha mencionado con anterioridad, el interprete sólo tiene en cuenta el Puerto B. La instrucción POKE permite acceder a los demás puertos de E/S, así como a características especiales que posea el PIC con el que se esté trabajando como pueden ser el TMR0, el conversor A/D, USART, etc..

```
'*****************************************************************************
' Programa: ART003.BAS                Fecha:  16/06/1999        *
'*****************************************************************************
'  Programa que enciende y apaga un led de la barra             *
' de Led`s, dependiendo si se ha pulsado el botón A0            *
'                                                               *
'  Revisón: 1.0              Programa para el PIC16F84          *
'  Velocidad de reloj: 4Mhz      Reloj instrucción: 1µs          *
'                                                               *
'  Reloj: tipo XT            Perro Guardián: Off                *
'*****************************************************************************
'*****************************************************************************
'                          Definiciones                         *
'*****************************************************************************
Symbol PortA   = $005               ' dirección del puerto A
Symbol TrisA   = $085               ' dirección del byte de control para el puerto A

Symbol ESBIT0= DIR0          ' Define una variable tipo bit que apunta al bit 0 de estado del
puerto B
Symbol Led     = PIN0        ' Define una variable tipo bit que apunta al bit 0 de puerto B
Symbol ValPuertoA = b0              ' Define una variable de tipo byte
Symbol Boton   = bit8               ' Define una variable de tipo bit

        POKE  TrisA,255             ' Todo el puerto A como entrada
        ESBIT0= 1                   ' Indica que el bit 0 del puerto B es de salida
        Led   = 0                   ' El led está apagado inicialmente

Bucle:
        PAUSE         100           ' Retardo de 100 ms, para eliminar rebotes
        PEEK PortA, ValPuertoA      ' Obtenemos el valor de los bits del puerto A
        Boton  = ValorPuertoA & $001 ' Obtenemos el valor del bit 0 del puerto A

        IF Boton = 1 THEN  LedOn    ' Si toma el valor 1 el led se enciende
        IF Boton = 0 THEN  LedOff   ' Si toma el valor 0 el led se apaga

        GOTO  Bucle                 ' Continua el bucle

LedOn: Led    = 1                   ' Pone a 1 el bit 0 del puerto B
        GOTO  Bucle                 ' regresa al bucle

LedOff: Led   = 0                   ' Pone a 0 el bit 0 del puerto B
        GOTO  Bucle                 ' regresa al bucle
```

POKE $85,0 ' Escribe 0 en el registro hexadecimal 85 (Programa el Puerto A como salida)

PEEK Dirección, Valor

Este comando lee el registro del PIC especificado *Dirección* y lo almacena en *Valor*. Al igual que POKE esta instrucción es propia del compilador PBC y no se encuentra en el interprete PBASIC. Como se ha dicho anteriormente el interprete solo tiene en cuenta el Puerto B. La instrucción POKE permite acceder a los demás puertos de E/S, así como a características especiales que posea el PIC que estemos empleando como pueden ser TMR0, conversor A/D, etc.

Las instrucciones *PEEK* y *POKE* permiten el acceso directo a todos los registro del PIC, incluidos los puertos A, B, C, D y E y a sus registros asociados de dirección (TRIS). Estas instrucciones operan en todos los bits, por ejemplo el byte completo de un registro particular a la vez. Cuando uno hace un POKE al Puerto A, es modificado todo el registro no solo un bit individual.

Si se necesita una sustancial manipulación de bits individuales, es recomendable que esa función sea

```
' Lee los bits del Port A empleando como variable intermedia B0
Symbol PortA = 5                    ' dirección del puerto A
Symbol TrisA = $85                  ' dirección del byte de control para el puerto A
               POKE TrisA,255       ' Todo el puerto A como entrada

Bucle:         PEEK PortA,B0        'Almacena en B0 el estado del puerto A
               IF Bit0 = 1 Then Etiqueta1   ' Salta a Etiqueta1 si Bit0(RA0) vale 1
               IF Bit1 = 0 Then Etiqueta2   ' Salta a Etiqueta2 si Bit1(RA1) vale 0
               GOTO Bucle           '

Etiqueta1:     HIGH 0               ' Coloca Pin0 (RB0) a 1
               GOTO Bucle

Etiqueta2:     LOW 1                ' Coloca Pin1 (RB1) a 0
               GOTO Bucle

               END
```

```
' Escribe los bits del Port A empleando como variable intermedia B0
Symbol PortA = 5                    ' dirección del puerto A
Symbol TrisA = $85                  ' dirección del byte de control para el puerto A
               POKE TrisA,0         ' Todo el puerto A como salidas

               PEEK PortA,B0        'Almacena en B0 el estado del puerto A
               Bit1 = 1             ' Coloca Bit1 de B0 a 1 es decir RA1
               Bit3 = 0             ' Coloca Bit3 en B0 a 0 es decir RA3
                                    ' el resto de los bits del PortA 2 y 4 quedan
inalterados
               POKE PortA, B0       ' Envía el nuevo byte al PortA para completar el
cambio
               END
```

asignada a un Pin del puerto B. De forma alternativa, la variable B0 permite la manipulación de bits individuales. Para ello debe ser activada en primer lugar y posteriormente usar POKE con una manipulación de bits cómoda. El siguiente es un ejemplo de dicha técnica que puede emplearse con cualquier registro del PIC.

IF ... THEN

IF *Comp* { AND / OR *Comp*}
THEN *Etiqueta*

Realiza una o más comparaciones entre variables y / o números y salta a la *Etiqueta* de línea si la comparación es verdadera, si no lo es, se ejecuta la siguiente instrucción del programa. El THEN en esta instrucción es esencialmente un GOTO a la *Etiqueta* de línea. El objeto de la izquierda en la comparación siempre debe ser una variable, es decir seguir la siguiente estructura.

Variable Relación Variable o Número

Comparaciones lógicas:

=	Igual que
<>	Diferente de
>	Mayor que
>=	Mayor o igual que
>	Menor que
<=	Menor o igual que

El operando a la derecha debe ser un número positivo o una variable con un valor positivo, ya que el PBC no soporta números negativos. La siguiente expresión es incorrecta b3 > -3. Se pueden combinar comparaciones de forma lógica con AND y OR. por ejemplo, b2 = 4 AND b3 = 5.

CUARTO PROGRAMA EJEMPLO

En este último programa, realizaremos funciones de entradas y salidas manejando dos de los displays de 7 segmentos (ver página siguiente).

CALL

Salta y ejecuta una subrutina en lenguaje ensamblador y posteriormente al concluir retorna al programa PBC .

CALL *Etiqueta*

Etiqueta : Es la etiqueta de la dirección de la subrutina en lenguaje ensamblador que debe ser ejecutada.

El compilador PBC tiene la habilidad de ejecutar subrutinas en lenguaje ensamblador que no son partes propiamente dichas del programa PICBASIC. Éste puede incluir en su interior subrutinas como son AD0, AD1 y AD que leen el conversor Analógico digital del PICStic3 o las rutinas CLOCKSET y CLOCKGET que se emplean para programar y leer el reloj de tiempo real del PICStic2.

El PBC puede usar también rutinas en ensamblador que se hayan escrito y hacerlas parte del programa compilado como son las necesarias para el control de los pins PA3 y PA4.

Después de terminar la rutina de lenguaje ensamblador, la ejecución continua por la siguiente instrucción PBC que sigue al comando CALL. Ejemplo:

```
' Programa para demostrar el uso de CALL para ejecutar subrutinas
' de lenguaje ensamblador incluidas en su interior
Loop:  CALL AD1                     ' Salta a subrutina ensamblador
                                    ' lee canal 1 A/D, almacena el
       W1 = W10                     ' resultado en W1
       CALL AD0                     ' Salta a subrutina ensamblador
                                    ' lee canal 0 del A / D
       SEROUT 0,2400, (#W1,#W10)    ' Envía ambas lecturas a puerta
                                    ' serie en pin 0
       IF W1 > W10 THEN Tone        ' Si lectura AD1 > AD0 beep
       GOTO Loop                    ' Vuelve y realiza otra lectura
Tone : SOUND 1, (100,150)           ' Altavoz en pin 1 beep
       GOTO Loop                    ' Vuelve y realiza otra lectura
```

Constantes Cadenas de Caracteres

PBC no proporciona capacidades para el manejo de cadenas de caracteres, pero las cadenas pueden emplearse con algunos comandos. Una cadena contiene uno o más caracteres y está delimitada por dobles comillas. No están soportadas secuencias de escape con caracteres no-ASCII (sin embargo muchos comandos PICBASIC contienen la posibilidad de este manejo).
Ejemplos:

" Hello" ' Cadena (En lugar de "H","e","l","l","o")

SEORUT0,N2400,("GATO")

es equivalente a

SEROUT0,2400,("G")
SEROUT0,2400,("A")
SEROUT0,2400,("T")
SEROUT0,2400,("O")

```
'*************************************************************************
'Programa: ART004.BAS                Fecha:  16/06/1999          *
'*************************************************************************
'  Pulsando el botón A0  se incrementa en una unidad el valor    *
'  mostrado por dos displays de 7 segmentos, hasta llegar a 99   *.
'  volviendo a comenzar la cuenta desde 0.                       *
'  Revisón: 1.0              Programa para el PIC16F84           *
'  Velocidad de reloj: 4Mhz      Reloj instrucción: 1µs          *
'                                                                *
'  Reloj: tipo XT            Perro Guardián: Off                 *
'*************************************************************************
'*************************************************************************
'Definiciones
'*************************************************************************
Symbol PortA          = $005            ' dirección del puerto A
Symbol PortB          = $006            ' dirección del puerto B
Symbol TrisA          = $085            ' dirección del byte de control para el puerto A
Symbol TrisB          = $086            ' dirección del byte de control para el puerto B

Symbol CsDigito1      = $002            ' bit del puerto A que activa al dígito 1
Symbol CsDigito2      = $004            ' bit del puerto A que activa al dígito 2
Symbol Cero           = $0be            ' Representación del cero
Symbol uno            = $00c            ' Representación del uno
Symbol dos            = $076            ' Representación del dos
Symbol tres           = $05e            ' Representación del tres
Symbol cuatro         = $0cc            ' Representación del cuatro
Symbol cinco          = $0da            ' Representación del cinco
Symbol seis           = $0f8            ' Representación del seis
Symbol siete          = $00e            ' Representación del siete
Symbol ocho           = $0fe            ' Representación del ocho
Symbol nueve          = $0ce            ' Representación del nueve

'Variables tipo bit
Symbol Boton          = bit2            ' Almacena el estado del botón A0

'Variables tipo byte
Symbol Digito1        = b2              ' Almacena el valor del dígito 1
Symbol Digito2        = b3              ' Almacena el valor del dígito 2
Symbol ValorPuertoA   = b4              ' Guarda el estado de todos los bits del puerto A
Symbol ValorDigito    = b5              ' Para almacenamiento temporal


              POKE  TrisA,%001          ' bit 0 entrada el resto salida
              POKE  TrisB,0             ' puerto B como salida

Inicio:       Digito1 = 0               'El dígito 1 toma el valor 0 inicialmente
              Digito2 = 0               'El dígito 1 toma el valor 0 inicialmente

Bucle:
              PAUSE         10          ' Retardo de 10 ms, para mostrar el dígito
              POKE  PortA,CsDigito1     ' Seleccionamos el dígito 1
              ValorDigito = Digito1
              CALL  MuestraDigito       ' Muestra el dígito 1

              PAUSE         10          ' Retardo de 10 ms, para mostrar el dígito
              POKE  PortA,CsDigito2     ' Seleccionamos el dígito 2
              ValorDigito = Digito2
```

Las cadenas son tratadas normalmente como una lista de valores de caracteres individuales.

COMPILADOR BASIC GRATUITO LET BASIC

Pensando en aquellos lectores que no deseen o no puedan adquirir los compiladores de BASIC para PIC en los que estamos basando el curso, se ha localizado un compilador gratuito, el LET BASIC, pero que desgraciadamente es incompatible con los compiladores mencionados hasta el momento.

Presentaremos una versión de algunos de los ejemplos, realizados con este compilador LET BASIC y haremos una somera exposición de sus comandos (Tabla adjunta).

El compilador LET BASIC (de la empresa *Leading Edge Technology* de Malta) produce ficheros *.ASM que pueden ser compilados con el MPASM. Para esto, sólo hay que ejecutar el fichero GO.BAT. Puede producir código para los PIC 16C54, 16C55, 16C56, 16C57, 16C71 y 16C84.

Sólo se permite un comando por línea de código, en las que no es necesario especificar el número de línea. Aunque si son necesarias etiquetas de línea para la instrucciones GOTO, GOSUB o IF THEN. Debe colocarse un espacio o TAB entre la etiqueta y el comando BASIC, así como el comando END al final del programa BASIC.

Los comandos del LET BASIC son los siguientes :

En esta versión sólo se permite una línea de DATA. Emplee RESTORE como siguiente comando después del de DATA.

Como el compilador produce código según la estructura del programa, es posible quedarse sin memoria fácilmente si no se tiene cuidado. Por ejemplo, un retardo producido con el comando DELAY produce cinco palabras de código máquina cada vez que se emplea, por lo tanto si usted escribe :

```
DELAY 255
DELAY 255
DELAY 255
DELAY 255
```

tratando de producir un retardo mayor, ha usado 20 palabras de memoria. Si necesita retardos grandes es mejor escribir una subrutina que produzca retardos grandes RETARDOM de la siguiente manera:

```
RETARDOM    for t=0 to 100
            delay 255
            next t
            return
```

EJEMPLOS CON LET BASIC

Existe una versión mayor, la 2.0, no gratuita que se vende por unas 10 libras, y soporta además los siguientes comandos:

LCD comandos: Move Cursor, CLS, Print, Home, GotoXY. LCD Alphanumeric Displays LM015, LM020, LM070 LM038, LM058, LM041, LM044L etc Todos los displays de 16 x 1 líneas a 20 x 4 líneas. El código se inserta solamente en el fichero .ASM después de una instrucción INCLUDE.

KEYPAD matriz de 4 X 4 (o menores)

READ A/D conversión analógico / digital en el PIC16C71.

READ/WRITE Lee y escribe la memoria EEPROM del Pic16c84.

MULTIPLE comandos por línea:- a=j*c : high led : print a etc

COMANDOS EXTRAS

INCLUDE <*dispositivo*>,<*portx*> .Donde *dispositivo* = LCD, Keypad y Portx = PORTB o PORTC

Comando	Función	Ejemplo

ASM {.....}	Coloca código ensamblador en el fichero BASIC.	Asm { movlw 12
		movwf temp1
		addwf temp2,w
		}

DATA	Tabla de valores de datos. Use READ y RESTORE para acceder a ella.	DATA 1,3,5,34,23,177,243
DEFINE	Coloca los PORTS como IN (entrada) o OUT (salida).	(En un reset todos los puertos se programan como entradas). DEFINE porta in,in,out,in Set-up como A.0=in, A.1=Out , A.2=IN , A.3=in DEFINE porta ins Programa PORTa = todo entradas. DEFINE portb Outs Programa PORTb = todo salidas.
DELAY	Retardo num	Retardo de num * 1 micro-segundos (4Mhz).
DIM	Todas las variables deben ser declaradas al principio del programa.	Dim a,ans,i,k,temp1 etc
END	Debe colocarse al final del programa.	
FOR var=<num> to <num>	For i=12 to 24	No se permite la instrucción step
GOSUB <etiqueta>	Salta a la línea con <etiqueta> y continua hasta encontrar RETURN.	
GOTO <etiqueta>	Salta a la línea con <etiqueta>.	
HIGH <sym>	Coloca a nivel alto el pin indicado con SYM (ver SYM)	- high led
IF <var><condición> Then <etiqueta>	Si la condición es verdadera salta a la línea con esa etiqueta.	= si igual (a=4) <> si no igual es decir diferente de (a<>4) <= menor o igual (a<=4) >= mayor o igual (a>=4)
LET<opcional>		Let a=a*6 OR a=a*6 Los operadores son: Suma +, Resta -, Multiplicación *, División /,And &, y OR # << Desplaza a la izquierda a=a<<3. Bit Cero pasa a 0. >> Desplaza a la derecha a=a>>1 Bit Siete pasa 0. INP a=inp(port) PEEK a=peek(dirección)
LOW <sym>	Coloca el pin en estado bajo igualado por SYM	LOW LED (LED equates como PORTA.4)
OUT <port,num>	Saca una variable o un numero al Port.	out(porta,12) out(portb,j)
PEEK(addr,var)	Usar con el comando LET.	a=peek(20)
POKE addr,<num><var>	Poke a FileNumber(addr) un valor Num or Var	poke 29,5 poke j,5 poke 20,j

INIT *<dispositivo>* Donde *dispositivo* = LCD, Keypad, A2D

CLS Borra la pantalla LCD.

CURSOR *<cmd>* Donde cmd = LEFT, RIGHT, HOME o X,Y goto X across, Y down.

PRINT *<var,num,$,">*

Ejemplo :PRINT A
PRINT "HELLO"
PRINT $A
PRINT "Line",a,"Data",j

INKEY Espera a que se pulse una tecla y luego devuelve el valor de la misma. Ejemplo : a=inkey

ADIN(num) Recoge una valor de la línea especificada por NUM del A/D.

Ejemplo : a=adin(2)

EEDATA *<address,var>* Recoge dato de la EEPROM del PIC16C84's. Donde la dirección es 0-63, debe emplearse una variable.

Ejemplo : a=eedata(32)
a=eedata(j)

STORE *<num,var>,<address,var>* coloca un número o variable en la EEprom. Ejemplo:

store 12,32 ;coloca 12 en la posición 32
store a,j ;coloca el contenido de 'a' en la dirección 'j'

NOTAS

Deseamos agradecer al Profesor Carmelo Ruiz Rejón del Dpto. de Genética de la Universidad de Granada, la realización de las fotografías que aparecen en esta serie de artículos, así mismo queremos dar las gracias a la empresa ***Sagitrón*** (http://www.sagitron.es) distribuidora en España de los productos Microchip, en especial a los señores José de Bustamante y Eduardo Bonilla, que nos han proporcionado desinteresadamente muestras de los nuevos microcontroladores flash PIC16F876 y PIC16F877, con las que hemos podido actualizar el software del programador MultiPIC.

BIBLIOGRAFÍA

- Martín Cuenca, E., Angulo J.M., y Angulo, I. (1998). ***"Microcontroladores PIC. La solución en un CHIP"***. 2ª Edición. Paraninfo-ITP.
- Martín Cuenca, E. y Moreno Balboa, J.M. ***"Diseño y Realización de Aplicaciones Industriales con Microcontroladores PIC"***. (en preparación).

```
REM
        REM Programa: ART001.BAS                Fecha:  2/9/1999
        REM
        REM   Programa que enciende y apaga alternativamente dos led de la barra
        REM  de Led`s.
        REM
        REM   Revisión: 1.0                Programa para el PIC16F84
        REM   Velocidad de reloj: 4Mhz      Reloj instrucción: 1us
        REM
        REM   Reloj: tipo XT               Perro Guardián: Off
        REM
        REM
        REM Definiciones
        REM
        REM  Definimos todos los bits del puerto B como salidas

                REM Definimos una variable de un byte
                DIM     iContador
                REM Todos los bits del puerto B como salida
                DEFINE          portb outs

                REM Creamos un sinónimo del bit 1 del puerto B
                SYM   LED1 = portb.1
                REM Creamos un sinónimo del bit 2 del puerto B
                SYM   LED2 = portb.2

                REM  Encendemos el LED0
lblBucleHIGH    LED1
                REM  Apagamos   el LED1
                LOW    LED2

                REM  Retardo de 200 * 255uS
                GOSUB           subRetardo

                REM  Apagamos   el LED0
                LOW    LED1
                REM  Encendemos el LED0
                HIGH   LED2

                REM  Retardo de 200 * 255uS
                GOSUB           subRetardo

                REM  Iniciamos el ciclo
                GOTO  lblBucle

subRetardo      FOR     iContador = 0 TO 150
                DELAY 255
                NEXT   iContador
                RETURN

                END
```

```
REM Programa: ART002.BAS                Fecha:  16/06/1999
        REM
        REM Programa que enciende y apaga un led de la barra
        REM de Led`s, dependiendo si se ha pulsado el botón A1
        REM
        REM Revisión: 1.0               Programa para el PIC16F84
        REM Velocidad de reloj: 4Mhz       Reloj instrucción: 1us
        REM
        REM  Reloj: tipo XT             Perro Guardián: Off
        REM

                REM Definimos una variable de un byte
                DIM     iContador, iValorBoton

                REM Todos los bits del puerto B como salida
                DEFINE          portb outs

                REM Todos los bits del puerto A como entrada
                DEFINE porta ins

                REM Creamos un sinónimo del bit 1 del puerto B
                SYM   LED1 = portb.1

                REM Leemos el puerto A
lblBucleLET     iValorBoton = INP(porta)

                REM Seleccionamos el bit 1 del puerto A
                LET     iValorBoton = iValorBoton&2

                REM Si el botón se pulsa, apagamos el led
                IF iValorBoton = 0 THEN lblApagaLed

                REM  Retardo de 50 * 255uS, para eliminar rebotes
                GOSUB           subRetardo

                REM  Encendemos el LED0
                HIGH    LED1

                REM  Iniciamos el ciclo
                GOTO  lblBucle

                REM  Apagamos   el LED0
lblApagaLed     LOW     LED1
                GOTO  lblBucle

subRetardo      FOR     iContador = 0 TO 50
                DELAY 255
                NEXT   iContador
                RETURN

                END
```