Cómo usar MultiPIC

Programación y diseño de dispositivos mediante microcontroladores PIC

Con el objetivo de refrescar la memoria, os recordamos que en el primer artículo de introducción a la serie se realizó la descripción de los diferentes compiladores Basic y de sus comandos. El mes pasado, un segundo artículo continuó con la descripción del sistema modular para aprendizaje además del grabador MultiPIC. También se escribió y compiló un programa ejemplo en ensamblador empleando el MPASM que distribuye gratuitamente Microchip. A continuación se detallaron los pasos para realizar la grabación del mismo con la versión MS-DOS del programa MultiPIC.

Módulo de aprendizaje

Con el programador MultiPIC, el Módulo-01 del mes pasado y el Módulo de aprendizaje que presentamos en este artículo se puede realizar el 90 % de las prácticas incluidas en el libro «Microcontroladores PIC. La Solución en un chip» indicado en la bibliografía. Estos dos módulos se conectan enfrentados a través del conector CN1 (Conector macho Módulo-01 y el conector hembra Módulo de aprendizaje).

Las características del Módulo de aprendizaje son:

—Dispone de ocho salidas digitales con diodos LEDs (PB0 - PB7).

—Presentación de la información mediante tres *displays* de 7 segmentos.

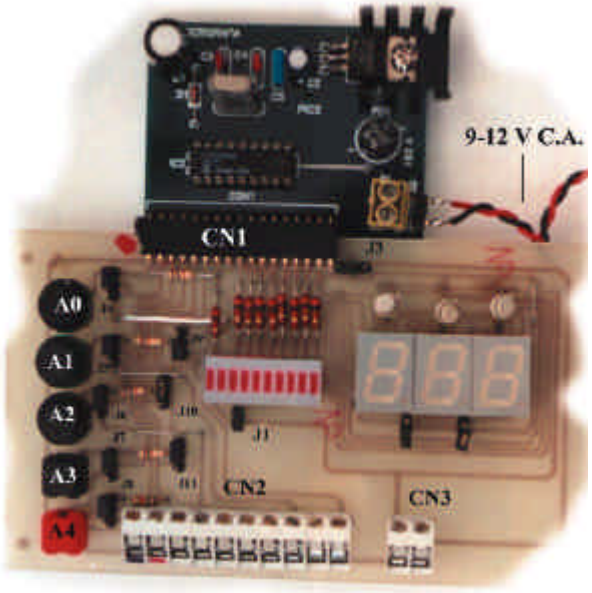
—Cinco entradas digitales de información mediante pulsadores (A0 - A4).

—Conexión exterior para las señales de entrada INT (interrupción externa) por el conector CN3.

—Conexión exterior para las señales T0CKI (entrada de reloj exterior) por el conector CN2.

—Todos los periféricos, tanto de entrada como de salida, pueden desconectarse usando los *jumpers* adecuados.

—Posibilidad de entradas externas E/S y A/D a través del conector CN2.



Módulo-01 enchufado al Módulo de Aprendizaje mediante el conector de 17 pines CN1.

En esta tercera entrega vamos a realizar nuestro primer programa en PBasic compilándolo y grabándolo en el microcontrolador. Además aprenderemos a usar el programa MultiPIC en sus versiones Windows y DOS.

—Conectores con la alimentación de +5 voltios (CN4) y el voltaje rectificado y filtrado del transformador (PWR).

El diagrama eléctrico del Módulo de aprendizaje se presenta en la «Figura 1». La configuración por defecto presenta los cinco pines del Puerto A conectados a los cinco pulsadores denominados (A0 - A4) de la «Figura 2», mientras que las ocho líneas del Puerto B (PB0 - PB7) están conectadas a una barra de diez diodos electroluminiscentes LEDs mediante resistencias limitadoras de corriente de 330 ohmios. En esta barra de LEDs el primero de ellos está conectado directamente a la línea de +5 voltios a través de otra resistencia de 330 ohmios. Esto permite conocer en todo momento si se está alimentando eléctricamente el módulo.

Los tres *displays* numéricos de 7 segmentos están controlados mediante las líneas del Puerto B (PB1-PB7) y las líneas A0, A1 y A2 del Puerto A. Esto permite realizar prácticas de presentación de información numérica.

Para emplear los presentadores de 7 segmentos se debe cambiar de posición los *jumpers* (puentes) J9 (A1), J10 (A2) y J11 (A3) que gobiernan el encendido de cada uno de estos *displays* a través de tres transistores del tipo BC547B. Conviene en este caso desconectar la barra de diodos LEDs retirando el *jumper* J1.

Cuando se haga uso de interrupciones externas (INT) habrá que cambiar la posición del *jumper* J3 para llevar la línea PB0/INT al conector CN3 donde queda disponible.

Para gobernar los tres *displays* numéricos se han elegido intencionadamente la líneas PB1-PB7, dejando así libre la línea PB0/INT. Esto permite trabajar con la entrada de interrupciones PB0/INT y con los presentadores de siete segmentos simultáneamente. Por este motivo, los puntos decimales de los presentadores se fijan empleando una resistencia a masa.

También se ha separado a propósito la línea A4, al tener ésta la característica adicional de ser la entrada de contador del Timer 0. Para el uso del TMR0/T0CKI a través de la entrada del pin A4, el *jumper* J8 permite llevar esta línea al conector CN2.

Por último, los *jumpers* (J4-J8) permiten llevar cualquiera de las líneas RA0-RA4 al conector CN2, lo que facilita su utilización en funciones de entrada y salida. Así, en el caso de usar un PIC16C71, pueden programarse indistintamente como entradas para el conversor A/D las líneas A0-A3. Este es un empleo diferente

al de su funcionamiento como pulsadores A0-A4 o al de E/S digitales.

Se dispone además de los conectores CN4 que proporciona un voltaje estabilizado en continua de +5 voltios y del CN5 (PWR) con el voltaje rectificado y filtrado del transformador.

Ejercicios con PBasic

Las explicaciones que se dan a continuación parten de la premisa de que se dispone del grabador MultiPIC con el compilador PBC incorporado. Si no es así, habrá que atenerse a las indicaciones del recuadro «Trabajo con la versión DOS».

Los pasos a seguir se presentan en la «Figura 3». Esta es una ampliación de la «Figura 7» del mes pasado. Como se puede comprobar comparándola con su homóloga, se ha añadido un apartado que ahora es el primero («Figura 3», paso I). Es decir, en primer lugar se ha de escribir el programa en lenguaje PICBasic y posteriormente ha de compilarse para obtener el archivo *.ASM.

El primer ejemplo consiste en un pequeño programa que enciende y apaga alternativamente dos diodos LEDs de la barra del Módulo de aprendizaje.

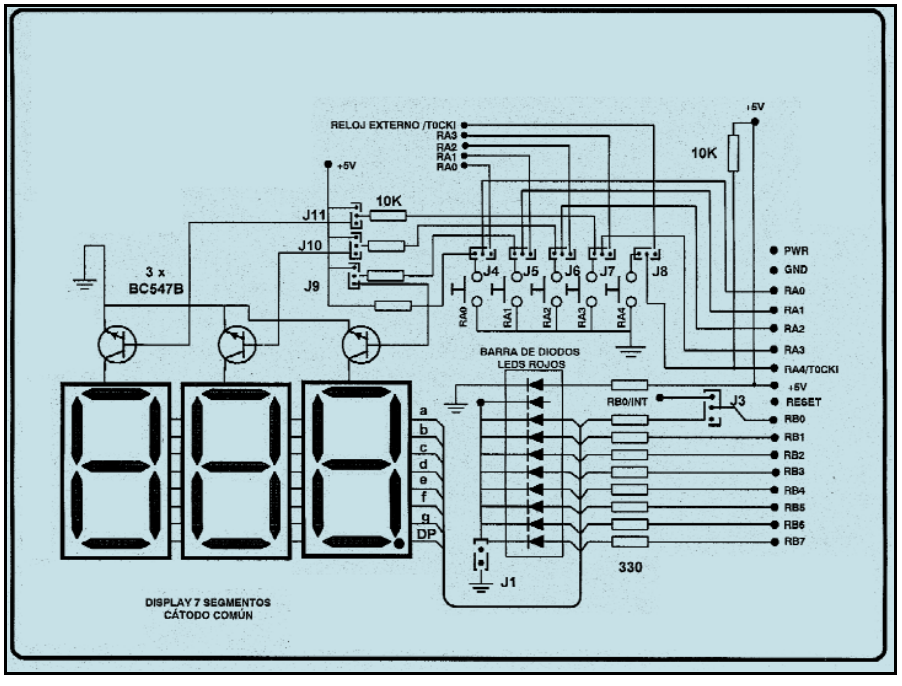
Una vez instalado el software siguiendo las indicaciones del *kit*, habrá que ejecutar el programa MultiPIC. Deberemos elegir del menú «Archivo» la opción «Nuevo Programa Fuente». En este punto se abre un editor ASCII que nos permitirá escribir el programa tal y como se muestra en la «Figura 4». Escribe tu primer programa en PBC como muestra el «Listado 1» y grábalo con el nombre ART001.BAS mediante la opción «Guardar Como» del menú «Archivo».

A continuación se pasa a compilar el programa ART001.BAS eligiendo la opción «Compilar». El programa MultiPIC automáticamente llama al compilador PicBasic Compiler (PBC) y al PICmicro Macro Assembler (PM), que son dos programas que funcionan bajo DOS.

El programa PBC compilará él fuente Basic, fichero ART001.BAS, a código fuente ensamblador, al que llamará ART001.ASM («Figura 3», paso II).

Si la compilación ha sido correcta y no ha ocurrido ningún error, se ejecutará el programa PM.EXE convirtiendo el fichero fuente ensamblador ART001.ASM a código máquina. Se creará un nuevo fichero que denominaremos ART001.HEX. Esta es la misma función que la realizada por el MPASM el mes pasado: convertir el fichero ART001.ASM en código máquina. El fichero de código máquina ART001.HEX ya puede ser manejado por el grabador y será el que se envíe al microcontrolador («Figura 3», paso III). Terminada esta operación, deberemos cerrar la ventana DOS.

Ahora solo falta grabar el microcontrolador. Lo primero será elegir la opción Programar PIC («Figura 3», paso IV). Al pulsar «Inicia Programación» («Figura 5») con la alimentación del grabador conectada se apagan los diodos LEDs en éste y en la pantalla aparece el mensaje de «Colocar PIC en zócalo de programación». Coloca por tanto el microcontrolador en el zócalo adecuado del grabador y pulsa «Ok». El programa tomará el



«Figura 1». Diagrama eléctrico del Módulo de aprendizaje.

Trabajo con la versión DOS

Los programa para el compilador PICBasic se escriben con un procesador de textos como el edit del DOS.

Usando cualquier editor de textos, escribir el programa y luego grabarlo en formato texto ASCII con el nombre de ART001.BAS.

Salir ahora del procesador de textos y pasar al subdirectorio donde hemos grabado el fichero de PICBasic. Si lo hemos llamado previamente desde el subdirectorio PBASIC, escribir desde el prompt del DOS:

CD C:\PBASIC

Si además hemos grabado en este subdirectorio el fichero ART001.BAS, escribir:

PBC ART001

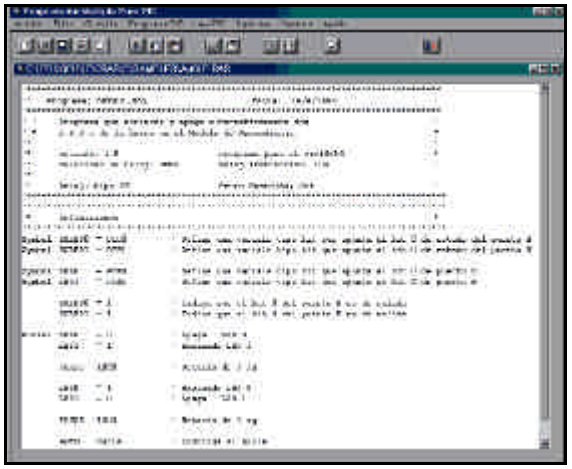
No necesitamos indicar la extensión *.BAS, ya que el programa PBC.EXE la asume por defecto. Sin embargo, si no la hemos usado en el nombre de su fichero, deberemos indicar la extensión que hayamos empleado. El programa PBC compilará en este momento su fuente Basic a código fuente ensamblador, al que llamará ART001.ASM. Si la compilación ha sido correcta y no ha ocurrido ningún error, a continuación se ejecutará automáticamente el programa PM.EXE para convertir el fichero fuente ensamblador a código máquina creando un fichero que denominará ART001.HEX. Este es el fichero en código máquina que se volcará al microcontrolador.

Si todas las operaciones han sido correctas, en la pantalla de nuestro ordenador aparecerán los siguientes mensajes:

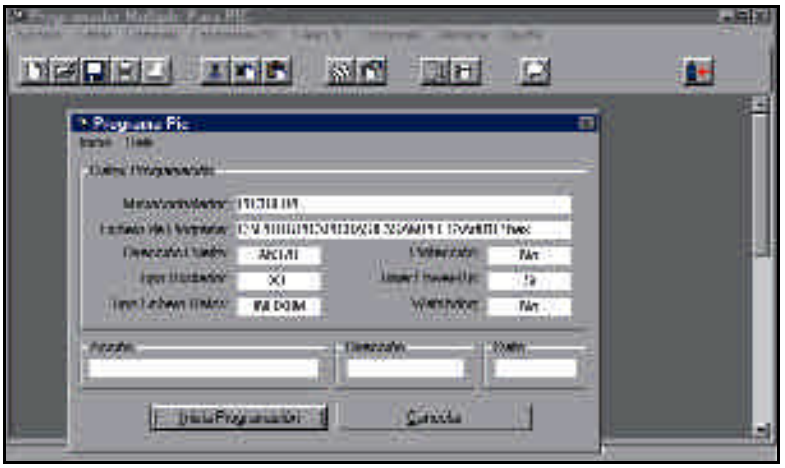
PicBasic Compiler Version X.YZ Copyright (c) 1995, 1999 microEngineering Labs

PIC Macro Assembler Version A.BC Copyright 1999, microEngineering Labs

Si sólo aparecen los mensajes anteriores, todo se habrá compilado y realizado correctamente y el fichero de código máquina habrá sido almacenado en el mismo subdirectorio como los otros programas de PICBasic.



«Figura 4». Programa MultiPIC con el ejercicio numero 1.



«Figura 5». Pantalla del programa MultiPIC al elegir la opción «Programa PIC».

fichero ART001.HEX generado y lo enviará al microcontrolador realizando así la grabación del mismo («Figura 3», paso IV).

Terminada la grabación, tal y como se aprecia en la «Figura 6», el programa muestra el fichero ART001.HEX grabado en el microcontrolador. Además, dispondremos ahora de los ficheros ART001.ASM y ART001.HEX en el mismo directorio donde se encontraba el fichero ART001.BAS.

A continuación retira el microcontrolador del grabador e introdúcelo en el Módulo-01. Conecta éste con el Módulo de aprendizaje y alimenta eléctricamente el montaje. Tu primer programa en PICBasic comenzará a funcionar.

Aunque el programa ejemplo es muy fácil de entender y se encuentra ampliamente comentado, haremos en este punto un repaso a los comandos PBasic empleados y otras peculiaridades del mismo.

En primer lugar, se encuentra la cabecera de comentarios que describe las operaciones que realiza, así como las opciones de funcionamiento y programación del microcontrolador.

Le siguen las definiciones mediante el comando *Symbol* y a continuación el cuerpo principal del programa. Este es un bucle continuo comprendido entre la etiqueta de línea *Bucle:* hasta el comando *GOTO Bucle*.

Para permitir al ojo visualizar el encendido y apagado de los LEDs se introducen dos retardos de 1 segundo mediante la instrucción *PAUSE 1000*

```
'*  Programa: ART001.BAS          Fecha:  19/4/1998      *
'*****************************************************************
' *  Programa que enciende y apaga alternativamente dos
'*
' *  L.E.D.s de la barra en el Módulo de Aprendizaje.      *
'*                                                        *
'*  Revisión: 1.0          Programa para el PIC16F84      *
'*  Velocidad de reloj: 4Mhz     Reloj instrucción: 1us   *
'*
'*  Reloj: tipo XT          Perro Guardián: Off           *
'*****************************************************************
'*****************************************************************
'*  Definiciones                                          *
'*****************************************************************
Symbol ESLED0= DIR0          'Define variable tipo bit que apunta
al bit 0 de estado del Port B
Symbol    ESLED1= DIR1       'Define variable tipo bit que apunta
al bit 1 de estado del Port B
Symbol    LED0     = PIN0          ' Define una variable tipo bit
que apunta al bit 0 de puerto B
Symbol    LED1     = PIN1          ' Define una variable tipo bit
que apunta al bit 1 de puerto B

      ESLED0   = 1          ' Indica que el bit 0 del puerto B es de
salida
      ESLED1   = 1          ' Indica que el bit 1 del puerto B es de
salida

Bucle:    LED0     = 0          ' Apaga  LED 0
      LED1     = 1          ' Enciende LED 1

      Pause    1000         ' Retardo de 1 sg

      LED0     = 1          ' Enciende LED 0
      LED1     = 0          ' Apaga  LED 1

      PAUSE    1000         ' Retardo de 1 sg


GOTO      Bucle             ' Continua el bucle
```

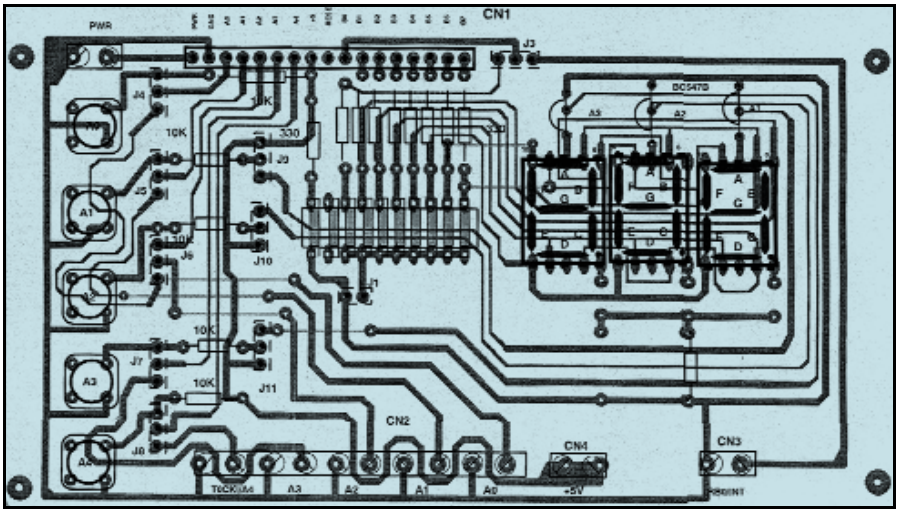
Variables

La pequeña cantidad de memoria RAM del PIC (ver «Arquitectura interna» en el artículo del mes pasado) se emplea para el almacenamiento de las variables. Debido a los limitados recursos del PIC, el PICBasic sólo proporciona 14 bytes de RAM para variables de usuario si se trabaja con el PIC16C84 y 25 bytes en el caso del PIC16F84. Esta memoria puede ser usada como variables tipo byte o variables tipo palabra (word). Los dos primeros bytes también pueden emplearse como variables tipo bit. Por lo tanto, el PICBasic dispone de tres tipos de variables: bit, byte y word.

Con la intención de proporcionar al programador el máximo uso de esta memoria, todas las variables de PICBasic tienen nombres y posiciones de memoria predefinidos.

Las variables tipo bit sólo pueden almacenar los números 0 o 1. Las variables tipo byte sólo pueden almacenar números de 0 a 255. También son llamadas variables de 8-bits. B3 es un ejemplo de variable tipo byte.

Las variables tipo palabra (word) pueden almacenar números comprendidos entre 0 y 65.535. Se llaman variables 16-



«Figura 2». Disposición de componentes del Módulo de aprendizaje.

bits o 2-bytes. W2 es un ejemplo de una variable tipo palabra.

Tened cuidado al emplear los diferentes tipos de variables. Si tratamos de almacenar un número demasiado grande en una variable, podemos sufrir una pérdida de información pues el número es truncado si se introduce en un espacio menor.

Adicionalmente se han definido los siguientes nombres de variables: PORT, Dirs y Pins. Al igual que W0 están son *overlays* y pueden ser referenciadas como bits. La variable PORT da nombre al Puerto B del PIC, y la variable Pins hace referencia a cada bit de un puerto.

Cuando se conecta la alimentación o se produce un *reset*, Dirs es colocada a $00 (todos los pines como entradas) al igual que las demás de las variables. Los valores de las variables carecen de signo. Los bits tienen los valores 0 o 1, los bytes de 0 a 255 y las palabras de 0 a 65.535.

Puede parecer que el uso de nombres prefijados sea una limitación. El comando *Symbol* permite dar a dichas variables nombres más apropiados e inteligibles para el programador. Esto lo podéis ver en la tabla que se incluye en esta página.

Si se emplea el PIC16F84 las variables llegan hasta B51 y W25. Es decir, dependiendo de la cantidad de RAM del micro así será el número de variables disponibles.

Variable tipo Palabra (Word)	Variable tipo Byte	Variable tipo Bit	Comentario
W0	B0	Bit0,Bit1,...,Bit7	Cada bit direccionable individualmente. Bit 0 y Bit 1 se usan junto con PA3 y PA4
	B1	Bit8, Bit9, ..., Bit15	Cada bit direccionable individualmente
W1	B2		
	B3		
W2	B4		
	B5		
W3	B6		
	B7		
W4	B8		
	B9		
W5	B10		
	B11		
W6	B12		
	B13		
W7	B14		No disponible si se usa opción -C
	B15		No disponible si se usa opción -C
W8	B16		No disponible si se usa opción -C, además son usados temporalmente
	B17		
W9	B18		
	B19		
W10	B20		
	B21		
Port	Pins	**Pin0, Pin1, ..., Pin7**	Salida PIN 0 = LOW, 1 = HIGH
	Dirs	**Dir0, Dir1,... , Dir7**	Estado I/O 0 = Entrada, 1 = Salida

Comentarios

Los programas bien escritos contienen los comentarios adecuados. En PICBasic un comentario comienza con la instrucción *REM* o con comillas simples (*'*). Los caracteres que le siguen en esa línea son ignorados.

A diferencia de otros Basic, *REM* es una instrucción única y es la abreviatura de *REMark*. Aquellos nombres de variables que comiencen por *REM* incluyendo esta misma no son válidos.
Ejemplo:

```
'*****************************************************************
'Programa: ART002.BAS          Fecha: 16/06/1999   *
'*****************************************************************
' Programa que enciende y apaga un led de la barra      *
' de LEDs dependiendo si se ha pulsado el botón A0       *
'                                                       *
' Revisión: 1.0          Programa para el PIC16F84      *
' Velocidad de reloj: 4Mhz      Reloj instrucción: 1us   *
'                                                       *
' Reloj: tipo XT          Perro Guardián: Off           *
'*****************************************************************
```

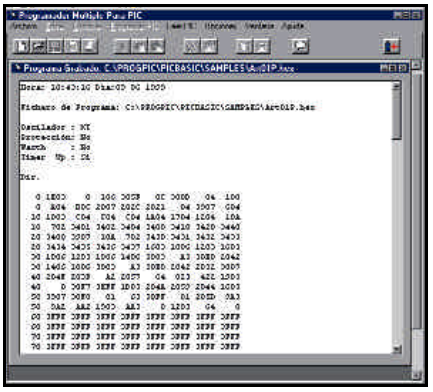
Symbol: Con la intención de hacer los programas más comprensibles, PICBasic permite al usuario definir sus propios símbolos. Estos símbolos pueden utilizarse para representar constantes, variables, etc. pero no como etiquetas de las líneas. Sólo esta permitida una definición por cada comando *Symbol*.
Ejemplos :

```
Symbol   PortA     = $005     ' dirección del Puerto A
Symbol   TrisA     = $085     ' dirección del byte de control
Puerto A
Symbol   Diez = 10            ' Constante simbólica
Symbol   Contador= W3         ' Nombre de variable tipo word
Symbol   BitVar    = BIT0     ' Nombre variable tipo Bit
```

Constantes numéricas: PICBasic permite que las constantes numéricas puedan definirse en tres bases: decimal (BASE 10), binario (BASE 2) y hexadecimal (BASE 16). Los valores binarios son definidos usando el prefijo % y los hexadecimales usando el prefijo $. Por defecto, se entiende que son valores decimales.
Ejemplos:
100: Valor Decimal 100
%100: Valor Binario del Decimal 4
$100: Valor Hexadecimal del Decimal 256

Constantes como caracteres: Las constantes pueden expresarse también como caracteres ASCII individuales usando las dobles comillas (“) en lugar del carácter. Pero si es una constante ASCII, sólo un carácter cada vez.

Para una programación más fácil, los caracteres individuales son convertidos en su equivalente ASCII:
“A”: Valor ASCII del Decimal 65
“d”: Valor ASCII del Decimal 100



«Figura 6». Pantalla del programa MultiPIC al finalizar la grabación.

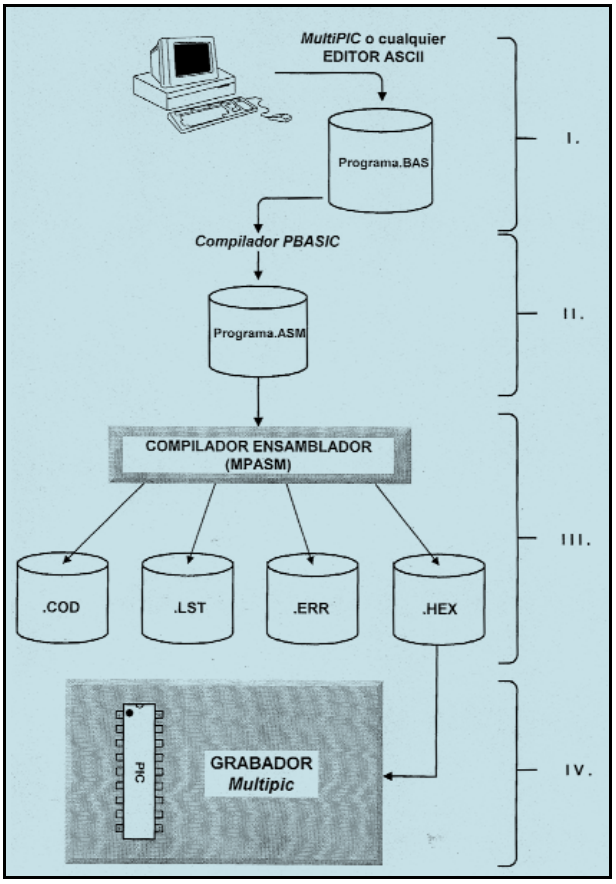
Ejemplo: *SEROUT 0,2400,("A")* es equivalente a *SEROUT 0,2400,(65)*

Identificadores: Símbolos identificadores pueden ser cualquier combinación de letras, números y subrayados (_). El primer carácter no puede ser un número. El símbolo identificador debe ser continuo sin espacios entre las secciones. Su longitud máxima es 32 caracteres, aunque las técnicas de la buena programación recomiendan que para la mayor legibilidad del programa sean cortos.

Etiquetas de línea: El compilador PICBasic no emplea números de línea. Cuando necesitemos referirnos a una línea determinada, ésta deberá tener una etiqueta de línea.

Al definir una etiqueta de línea podemos comenzar en cualquier columna, pero debe terminar con dos puntos (:). Si la etiqueta se define inicialmente, al referenciarla con posterioridad en otra parte del programa no se usarán los dos puntos.
Ejemplo:
Bucle: LED = 0 ; Apaga LED 0

No es necesario haber definido una etiqueta de línea antes de su uso en el programa, pero debe estar definida en algún lugar del mismo. Tampoco pueden existir dos etiquetas de línea iguales.



«Figura 3». Esquema de la secuencia de pasos para la grabación del microcontrolador.

Las etiquetas pueden ser una combinación de letras, números y subrayado, pero el primer carácter no puede ser un número. La etiqueta debe ser continua sin ningún espacio entre sus secciones. La longitud máxima de caracteres es de 32, pero es una buena técnica de programación usar etiquetas cortas para ayudar a la legibilidad del programa. Es una práctica habitual construir etiquetas descriptivas que den alguna indicación de la funcionalidad del código o de su propósito.

No pueden usarse como etiquetas palabras asignadas al PICBasic como son las instrucciones y los nombres de las variables.

Líneas multicomandos: Con la idea de permitir programas más compactos y el agrupamiento lógico de comandos relacionados, PICBasic soporta el uso de dos puntos (:) para separar los comandos colocados en la misma línea. Los siguientes ejemplos son equivalentes

Las variables tipo palabra (word) pueden almacenar números comprendidos entre 0 y 65.535. Se llaman variables 16-bits o 2-bytes

```
' Bucle continuo empleando GOTO
Parpadeo:  HIGH 3          ' Coloca PB3 a nivel alto
           PAUSE 500       ' Retardo de medio segundo
           LOW 3           ' Coloca PB3 a nivel bajo
           PAUSE 500       ' Retardo de medio segundo
           GOTO  Parpadeo  ' Salta a  la etiqueta Parpadeo
```

Bibliografía

Microcontroladores PIC. La solución en un chip. Martín Cuenca, E.; Angulo J.M., y Angulo, I. (1998). 2ª Edición. Paraninfo-ITP.

Diseño y realización de aplicaciones industriales con microcontroladores PIC (en preparación). Martín Cuenca, E. y Moreno Balboa, J.M.

Fundamentos de electrónica moderna. Teoría y diseño de circuitos. Martín Cuenca, E. y Moreno Balboa, J.M. (1998).

W2 = W0
W0 = W1 equivalen a W2 = W0 : W0 = W1 : W1 = W2
W1 = W2

GOTO Etiqueta_de_Línea

Esta instrucción salta a la etiqueta de línea y continúa la ejecución del programa desde ese punto, es decir, produce un salto a otro punto del programa.
Ejemplo: (ver Bucle continuo empleando GOTO).

PAUSE n: Detiene la ejecución del programa durante «n» milisegundos. «n» debe ser un número o una variable cuyo valor esté comprendido entre 0 y 65.535. Por tanto el máximo retardo generado por la instrucción *PAUSE* son 65.535 milisegundos, algo menos de un minuto.

Ejemplos:

PAUSE 500: Detiene la ejecución del programa medio segundo

PAUSE B4: Detiene la ejecución del programa según el valor de B4

El comando *PAUSE* puede generar retardos muy precisos, mucho más que los creados con la instrucción NAP, pero con la diferencia que la instrucción *PAUSE* no coloca el microcontrolador en modo de bajo consumo *sleep*. Durante el retardo producido por *PAUSE* todas las salidas se mantienen en su estado previo (alto o bajo) y continúan gobernando cualquier periférico externo.

Dr. Eugenio Martín Cuenca (emartin@goliat.ugr.es)
Ing. José María Moreno Balboa