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

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 en segundo articulo continuó la descripción del *Sistema modular* para aprendizaje (foto1) además del grabador ***MultiPIC***, que se emplearán durante la serie.

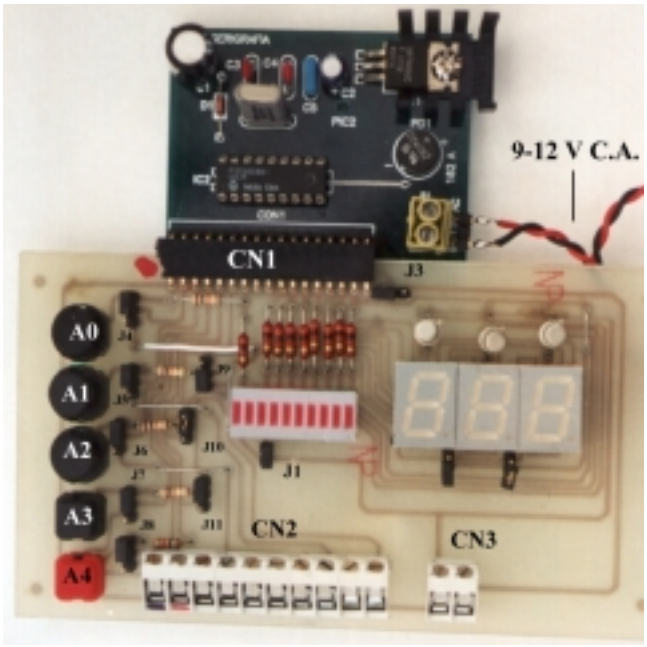

Foto 1. - ***Módulo-01*** *conectado al* ***Módulo de Aprendizaje*** *mediante el conector de 17 pines CN1.*

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 usando la versión MS-DOS del programa ***MultiPIC***. En esta tercera entrega realizará sus primeros programas en PBASIC empleando el material descrito con anterioridad compilará el programa y lo grabará en el microcontrolador. Se explicará el uso de las dos versiones Windows y DOS del programa ***MultiPIC***.

MÓDULO DE APRENDIZAJE.

Con el programador ***MultiPIC***, el ***Módulo-01*** del mes pasado, y ***Módulo de Aprendizaje*** que se presenta hoy, puede realizar el 90 % de las practicas incluidas el libro "*Microcontroladores PIC. La Solución en un CHIP*" indicado en la bibliografía. Estos dos módulos se conectan enfrentados a través del conector CN1, ya que el ***Módulo-01*** dispone de un conector CN1 macho mientras que el módulo de ***Aprendizaje*** dispone de un conector CN1 hembra.

Las características del ***Módulo de Aprendizaje*** son:

- *Dispone de ocho salidas digitales con diodos L.E.D.s (PB0 – PB7)..*
- *Presentación de la información mediante tres displays de 7 segmentos.*
- *Posee 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.*
- *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) figura 2, mientras que las ocho líneas del Puerto B (PB0 - PB7) están conectadas a diodos electroluminiscentes ***L.E.D.s*** de una barra de diez mediante resistencias limitadoras de corriente de 330 ohmios. En esta barra LEDs, el primero de ellos, está conectado directamente a la línea de +5 voltios a través de otra resistencia de 330 ohmios. Permite conocer en todo momento si se esta alimentando eléctricamente el módulo.

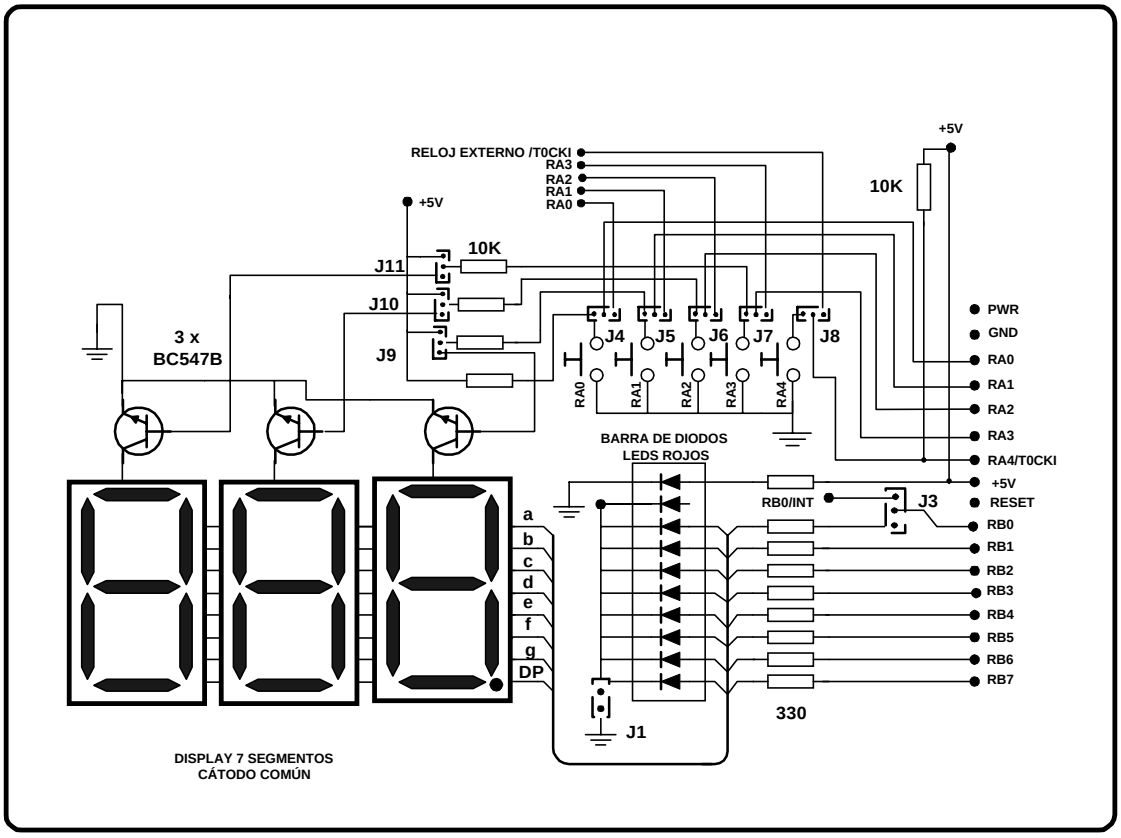

Figura 1. - *Diagrama eléctrico del Módulo de Aprendizaje.*

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 las practicas 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.

En los casos donde deba hacerse uso de interrupciones externas (***INT***), debe 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 exprofeso la línea A4, al tener esta, 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.

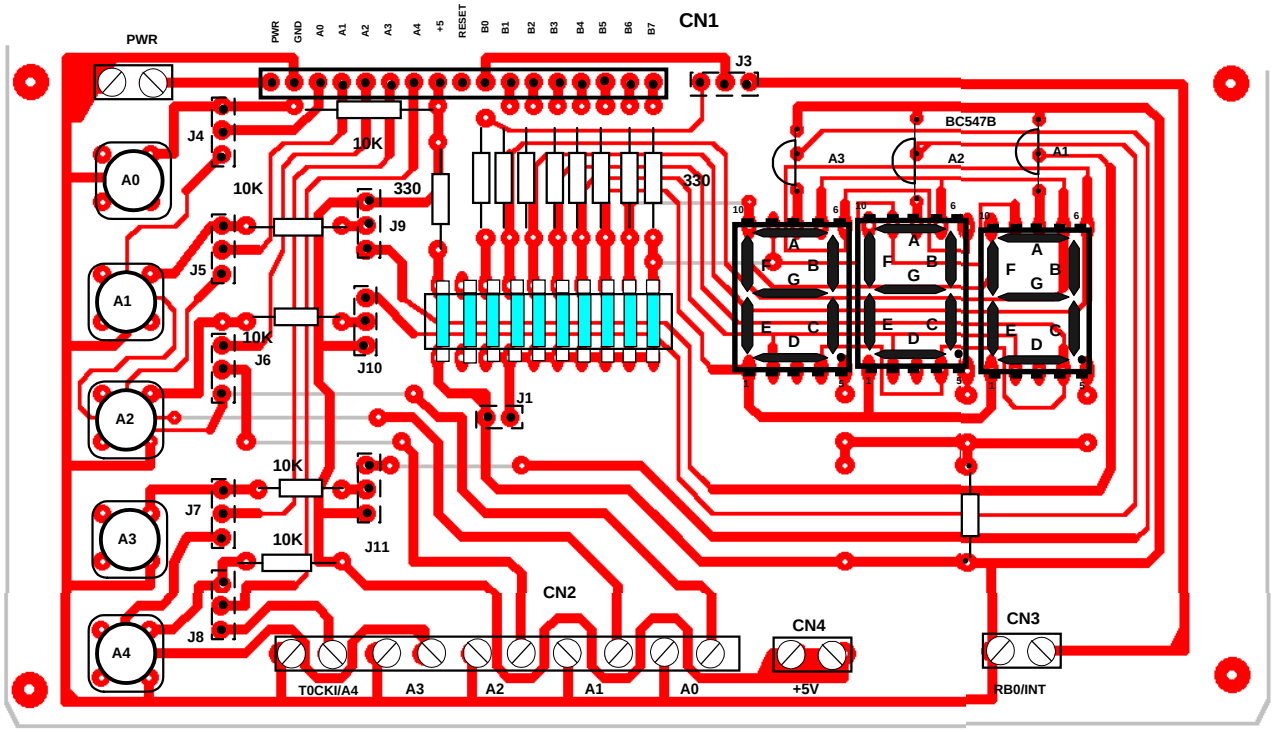

Figura 2. - *Disposición de componentes del Módulo de Aprendizaje.*

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 usted dispone del grabador ***MultiPIC*** con el compilador PBC incorporado. Si no es así aténgase a las indicaciones del apéndice A.

Los pasos a seguir se presentan en la figura 3, que es una ampliación de lo ya expuesto el mes pasado. Como puede comprobar comparándola con su homóloga del mes anterior, se ha añadido un apartado, que ahora es el primero. Es decir, en primer lugar se ha de escribir el programa

en lenguaje PICBASIC y posteriormente ha de compilarse para obtener el archivo *.ASM, que enlaza la secuencia de pasos con lo explicado el mes pasado.

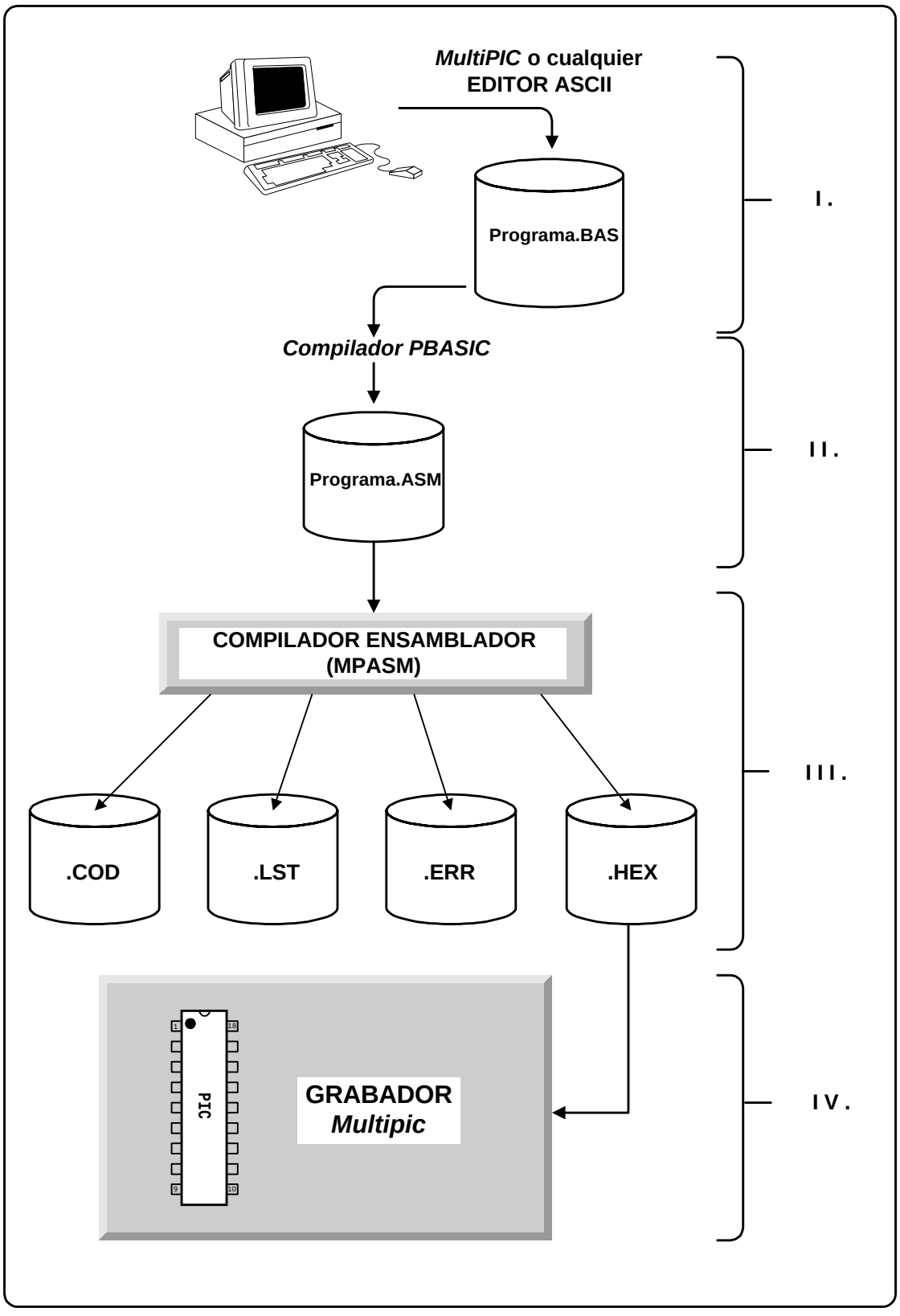

Figura 3. *Esquema de la secuencia de pasos para la grabación del microcontrolador.*

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, ejecute el programa MultiPIC. Elija del menú ***Archivo*** la opción ***Nuevo Programa Fuente***. En este punto se abre un editor ASCII que le permitirá escribir el programa tal y como se muestra en la figura 4. Escriba su primer programa en PBC como muestra el listado 1 y una vez hecho esto grábelo con el nombre **ART001.BAS** mediante la opción del menú ***Archivo*** de ***Guardar Como***.

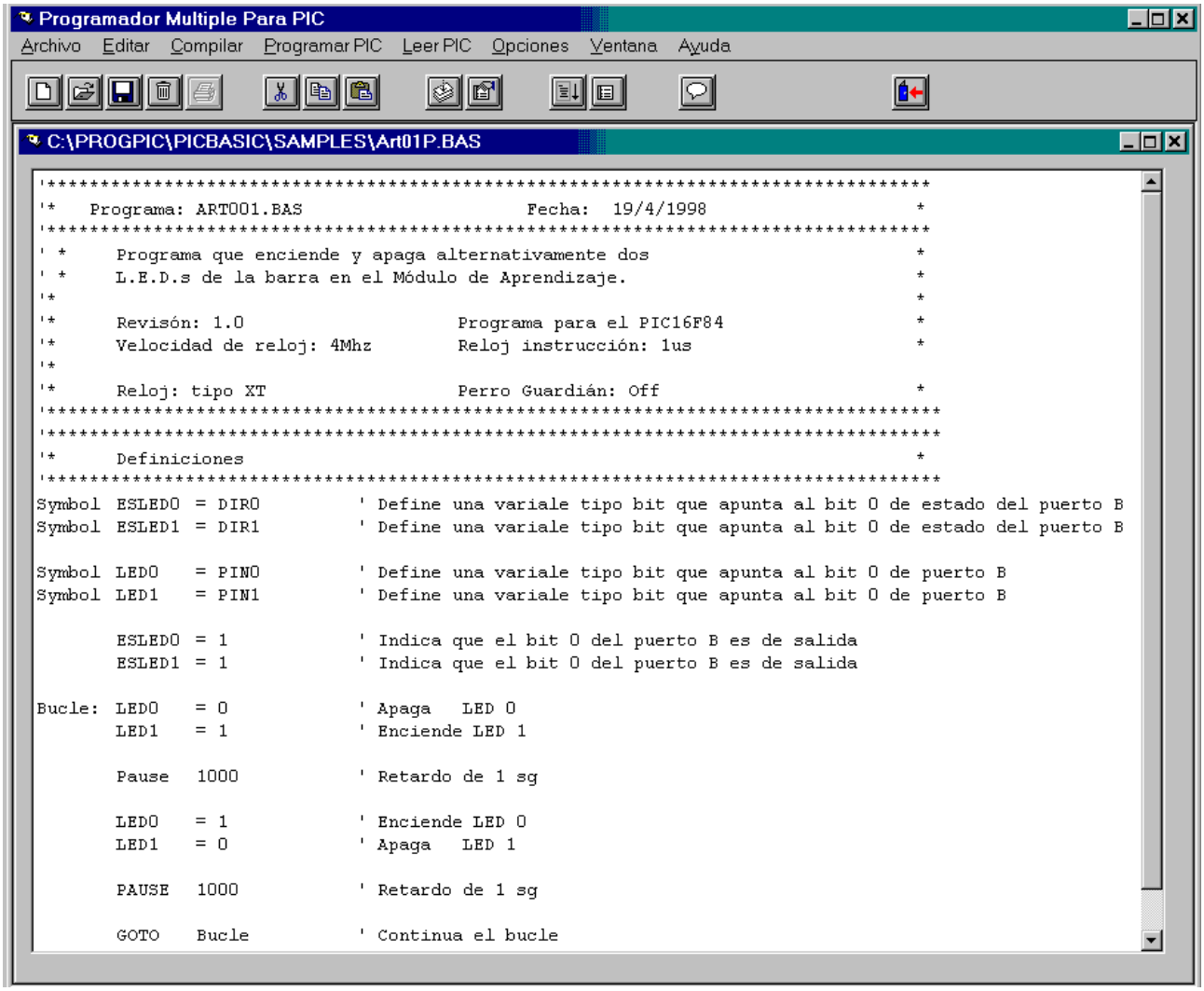

Figura 4. – *Pantalla del programa MultiPIC con el ejercicio numero 1.*

En este momento se pasa a compilar el programa ART001.BAS eligiendo la opción ***Compilar*** del menú. Automáticamente el programa ***MultiPIC*** llama al compilador *PicBasic Compiler* (PBC) y a continuación al *PICmicro Macro Assembler* (PM) que son dos programas que funcionan bajo DOS. El primero convierte el fichero ART001.BAS en otro ART001.ASM (figura 3, paso II), mientras que el segundo programa realiza la misma función que el MPASM ya mencionado el mes pasado, convierte el fichero ART001.ASM en un fichero que puede manejar el grabador el ART001.HEX (figura 3, paso III). Una vez terminada esta operación, solo debe cerrar esta ventana DOS. Ahora solo falta grabar el microcontrolador. Elija 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 este y en la pantalla aparece el mensaje de "*Colocar PIC en zócalo de programación*". Coloque por tanto el microcontrolador en el zócalo adecuado del grabador y pulse "*OK*". En este momento el programa toma el fichero ART001.HEX generado y lo envía a microcontrolador realizando así la grabación del mismo.

Al finalizar la grabación, el programa le mostrará el fichero ART001.HEX que ha sido grabado en el microcontrolador, tal y como se aprecia en la figura 6.

Además, ahora dispone de los ficheros ART001.ASM y ART001.HEX, en el mismo directorio donde se encontraba el fichero ART001.BAS.

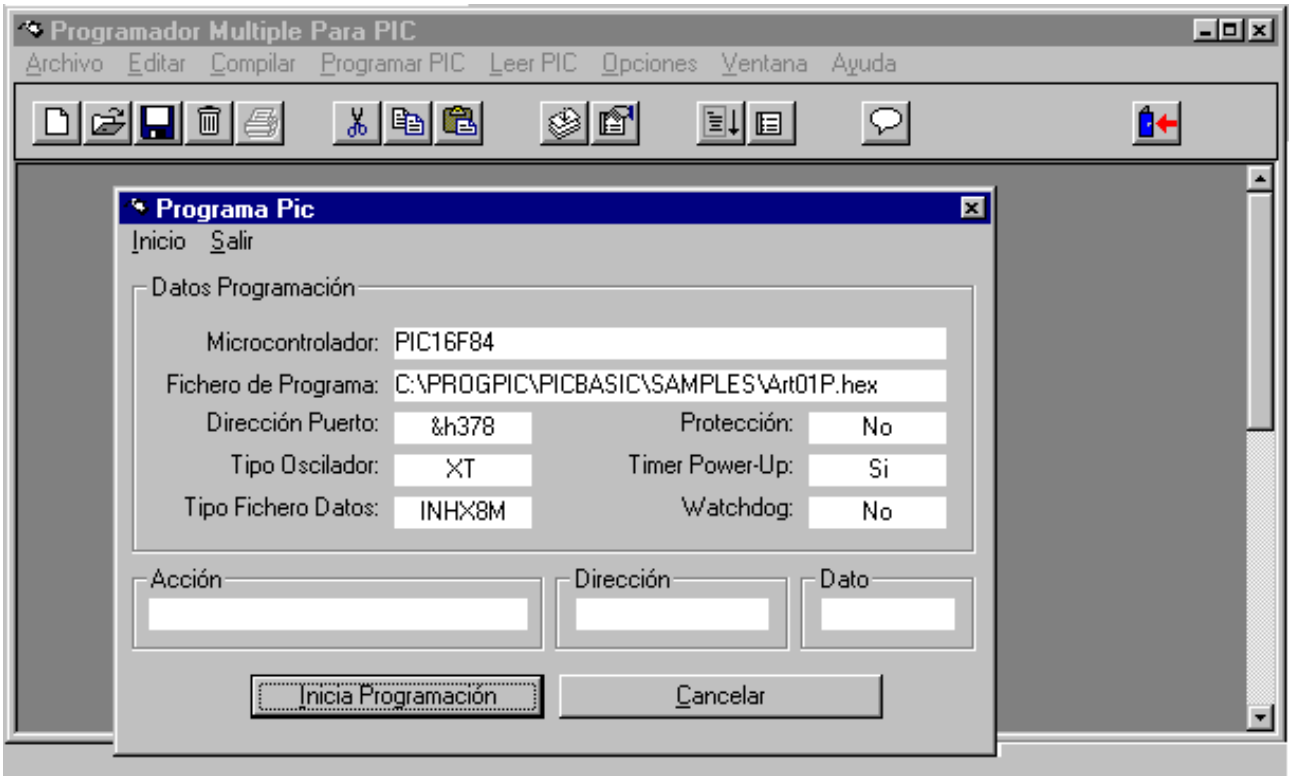

Figura 5. – *Pantalla del programa MultiPIC al elegir la opción "Programa PIC".*

Ahora solo tiene que retirar el microcontrolador del grabador, introducirlo en el ***Módulo-01***, conectar este con el ***Módulo de aprendizaje*** y alimentar eléctricamente el montaje. Su 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 el programa 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. Esta es un bucle continuo comprendido entre la etiqueta de línea Bucle: hasta el comando GOTO Bucle.

Para permitir visualizar al ojo 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.                    *
'*                                                                           *
'*       Revisón: 1.0              Programa para el PIC16F84                 *
'*       Velocidad de reloj: 4Mhz      Reloj instrucción: 1us                *
'*
'*       Reloj: tipo XT            Perro Guardián: Off                       *
'*****************************************************************************
'*****************************************************************************
```

```
'*         Definiciones                                                        *
'*************************************************************************************
Symbol ESLED0         = DIR0          ' Define una variable tipo bit que apunta al bit 0 de estado del Port B
Symbol ESLED1         = DIR1          ' Define una 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
```

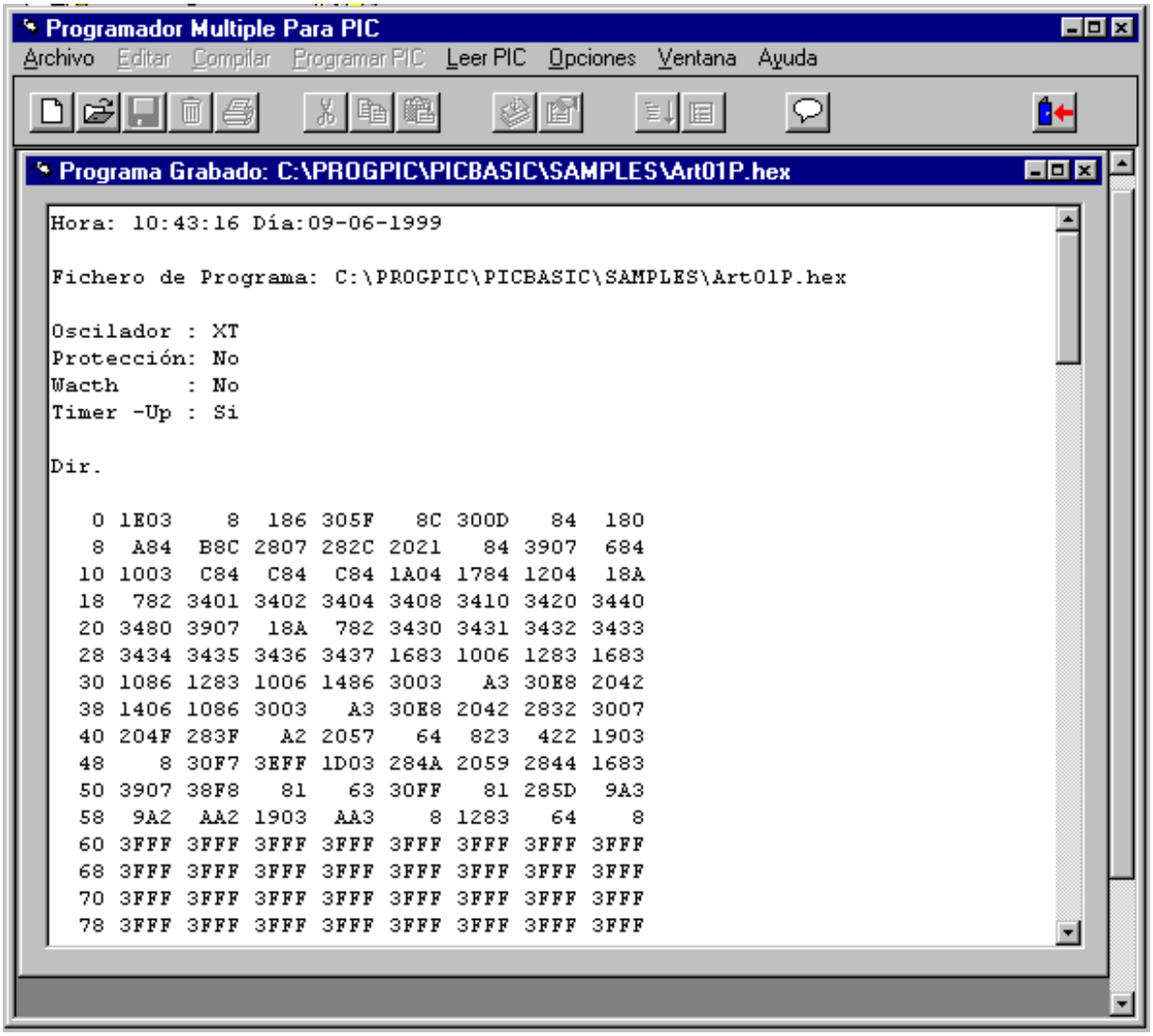

Figura 6. – *Pantalla del programa MultiPIC al finalizar la grabación.*

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 ser usados como variables tipo bit. Es decir 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 solo pueden almacenar números 0 ó 1. Las variables tipo byte solo 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) solo pueden almacenar número entre 0 y 65535. También son llamadas variables 16-bits o 2-bytes. **W2** es un ejemplo de una variable tipo palabra.

Debe tener cuidado en el empleo de los diferentes tipos de variables. Si trata de almacenar un número demasiado grande en una variable, puede sufrir una pérdida de información, pues el número es truncado al introducirlo en un espacio menor.

Se han definido adicionalmente 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.

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,
	B17		además son usados temporalmente
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

Cuando se conecta la alimentación o se hace un reset, **Dirs** es colocada a $00 (todos los pines como entradas) al igual que el resto de las variables. Todos los valores de las variables carecen de signo. Los bits tienen los valores 0 ó 1, los bytes de 0 a 255, y las palabras de 0 a 65535.

Puede parecer que el uso de nombres fijos limiten al programador, pero pueden emplearse nombres más apropiados e inteligibles para el programador a dichas variables mediante el comando ***SYMBOL***.

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 posibles.

COMENTARIOS

Los programas bien escritos contienen comentarios adecuados. Un comentario en PICBASIC 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 la abreviatura de REMark. Aquellos nombres de variables que comiencen por REM, incluyendo esta misma no son validos.

Ejemplo:

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

SYMBOL

Con la intención de hacer los programas más comprensibles, PICBASIC permite al usuario definir sus propios símbolos. Estos símbolos pueden ser usados para representar constantes, variables, etc. No pueden emplearse sin embargo como etiquetas de las líneas. Por cada comando ***SYMBOL*** solo puede realizarse una definición.

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 ser definidas 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 $. Los valores decimales son los usados por defecto y no necesitan ningún prefijo.

```
100     ‘ Valor Decimal 100
%100    ‘ Valor Binario del Decimal 4
$100    ‘ Valor Hexadecimal del Decimal 256
```

CONSTANTES COMO CARACTERES

Las constantes pueden ser expresadas también como caracteres ASCII individuales usando las dobles comillas “ en lugar del carácter. Pero solo puede ser usado un carácter cada vez si es una constante ASCII.

Para un a programación más fácil, caracteres individuales son convertidos en su equivalente ASCII. Carácter constante puede ser indicado usando dobles comillas y puede contener un solo carácter (si no serian constantes de cadena).

```
“A”     ‘ Valor ASCII del Decimal 65
“d”     ‘ Valor ASCII del Decimal 100
```

Ejemplo:

```
SEROUT  0,2400,(“A”)   es equivalente a   SEROUT   0,2400,(65)
```

IDENTIFICADORES

Símbolos identificadores pueden ser cualquier combinación de letras, números y subrayados(_), pero 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, pero las técnicas de la buena programación recomiendan que sean cortos para la mayor legibilidad del programa.

ETIQUETAS DE LÍNEA

El compilador PICBASIC no emplea números de línea. Cuando se necesita referirse a una línea determinada, ésta debe poseer una etiqueta de línea.

Cuando se define una etiqueta de línea, puede comenzar en cualquier columna, pero debe terminar con dos puntos (:). Cuando la etiqueta es definida inicialmente, los dos puntos no se usan cuando se la referencia posteriormente en otra parte del programa.

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, al igual que no pueden existir dos etiquetas de línea iguales.

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 practica común hacer la etiqueta descriptiva y que dé 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 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

```
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 continua la ejecución del programa desde ese punto, es decir produce un salto a otro punto del programa.

Ejemplo

```
‘ 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
```

PAUSE *n*

Detiene la ejecución del programa durante ***n*** milisegundos. ***n*** debe ser un número o una variable cuyo valor debe estar comprendido entre 0 y 65535. 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 crear retardos de tiempo muy precisos, mucho más que lo 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.

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).

****

TRABAJO CON LA VERSIÓN DOS

Los programa para el compilador PIC BASIC se escriben con un procesador de textos como el EDIT del DOS.

Use cualquier editor de textos y escriba el siguiente programa y luego grábelo en formato texto ASCII con el nombre de ART001.BAS.

COMPILAR Y ENSAMBLAR EL PROGRAMA

Salga ahora salga del procesador de textos, y pase al subdirectorio donde se ha salvado el fichero de PICBASIC. Si lo ha llamado previamente desde el subdirectorio PBASIC, escriba desde el prompt del DOS

```
CD      C:\PBASIC
```

Si además ha grabado en este subdirectorio el fichero ART001.BAS, escriba

```
PBC     ART001
```

No es necesario indicar la extensión .BAS, ya que el programa PBC.EXE la asume por defecto. Sin embargo si no ha usado para el nombre de su fichero la extensión .BAS, debe indicar la extensión que haya empleado. El programa PBC compilará en este momento su fuente BASIC en 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 en 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 su computadora aparecerán los siguientes mensajes:

```
PicBasic Compiler Version X.YZ Copyright © 1995, 1999 microEngineering Labs
PIC Macro Assembler Version A.BC Copyright 1999, microEngineering Labs
```

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