Curso de Aplicaciones con Microcontroladores PIC (VII).

Dr. Eugenio Martín Cuenca

Dpto. de Biología Animal
Facultad de Ciencias. Universidad de Granada.
E-mail:emartin@goliat.ugr.es
Web: http://www.ugr.es/~croneco/

Teoría (VII)

MEMORIA DE DATOS EEPROM EN LOS PIC 16C84 Y 16F84

Se ha explicado que los microcontroladores PIC16C84 y 16F84 disponen de una memoria de almacenamiento de datos **EEPROM** (*Electrically Erasable Programmable Read Only Memories*) de 64 bytes. Esta memoria para el almacenamiento de los datos es no volátil, no se necesita alimentación exterior para el mantenimiento de los mismos, por lo tanto se mantienen cuando el dispositivo se queda sin alimentación. La duración de la programación es de 10 ms por byte controlada automáticamente por un temporizador interno. Al escribir en una dirección, su contenido es borrado cuando se introduce el nuevo dato; es por esto por lo que no existe ningún comando específico de borrado. Esta memoria puede ser escrita o leída durante el funcionamiento normal en todo el rango de alimentación V_{DD}.

Una de las aplicaciones ideales para esta memoria se encuentra en los sistemas de seguridad, para poder modificar y almacenar un código de seguridad diferente. La programación de la EEPROM de datos dura aproximadamente unos 10 ms por byte y debe seguirse un procedimiento para evitar el que puedan producirse escrituras espúreas. La lectura de los datos solo dura tres ciclos de reloj.

El rango de direcciones de los 64 bytes que va desde $00 a $3F, no está mapeado directamente en el banco de registros por lo debe accederse de forma indirecta empleando los 4 registros especiales siguientes:

Registro	Dirección	Utilidad
EECON1	$88	Registro con 5 bits de control y definición del modo de funcionamiento.
EECON2	$89	Registro fisicamente inexistente. Se usa en el proceso de seguridad durante la escritura.
EEDATA	$08	Retiene el byte de datos para su lectura o escritura.
EEADR	$09	Retiene la dirección de la posición EEPROM a acceder.

R36

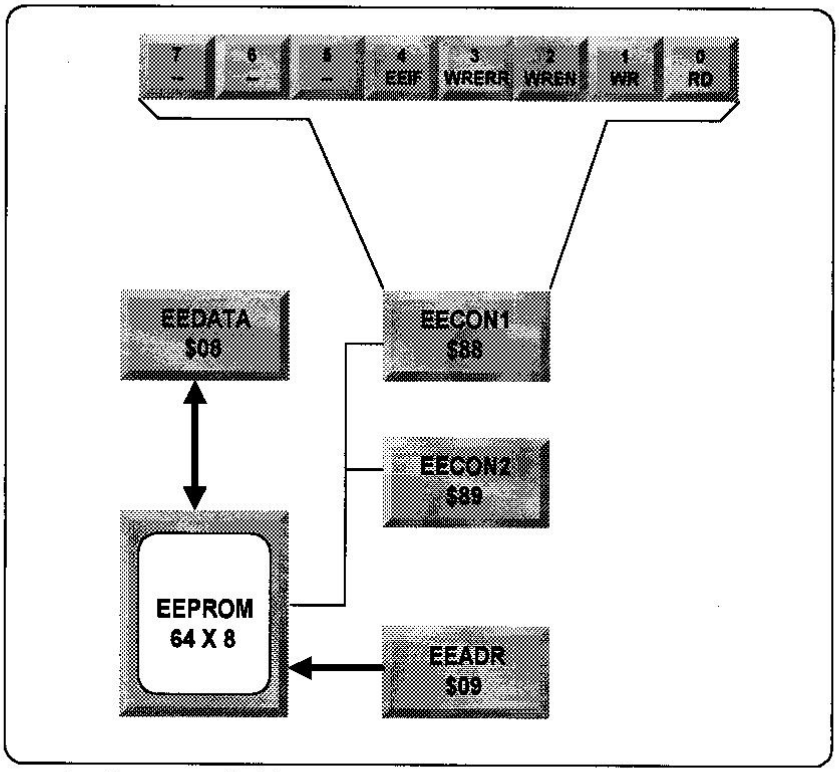

Figura 1.- *Diagrama de bloques relacionados con la EEPROM de datos de los PIC16F84 y 16C84.*

El registro **EEADR** puede direccionar hasta un máximo de 256 bytes, de los que solo se han implementado 64. Es por esto que los 2 bits de mayor peso deben permanecer a 0 para asegurar el no sobrepasar los 64 bytes.

El registro **EECON1**, es el de control de los 5 bits de menor peso que están implementados fisicamente con la siguiente estructura:

0 - RD (*Read Data*)	Poner a 1 para leer un dato. Después pasa automáticamente a 0
1- WR (*Write Data*)	Poner a 1 para escribir un dato. Después pasa automáticamente a 0
2 - WREN (*Write Enable*)	Poner a 1 para **autorizar** escritura. Con valor 0 impide la escritura.
3 - WRERR (*Write Error*)	Se pone a 1 cuando se produce un error de escritura. Por ejemplo un reset prematuro. Si se produce, no varían los contenidos de EEDATA y EEADR, para que la operación pueda repetirse correctamente.
4 - EEIF (*EEPROM Interupt Flag*)	1: Operación escritura completada. Hay que colocarlo a 0 mediante programa. 0 : Operación de escritura no completada o no comenzada.

Los bits de control **RD** y **WR** indican respectivamente lectura o escritura. No hay que ponerlos a 0, solo a 1. Se borran automáticamente cuando la operación de lectura o escritura ha sido completada. El bit WREN es él que habilita la escritura cuando es

puesto a 1. Por último el bit **WRERR** toma valor 1 cuando el proceso de escritura se interrumpe, ya sea mediante un reset a través del pin MCLR o por desbordamientos del " *Perro Guardián* " WDT. Si se produce esta situación, el usuario puede detectarla chequeando el bit WRERR y reescribir esta posición, ya que el dato y la dirección se mantienen sin alteración en los registros EEDATA y EEADR. La bandera de interrupción EEIF toma valor 1 cuando se completa la escritura y debe ser puesta a 0 mediante programa.

El registro **EECON2** no tiene existencia física, por lo que es imposible leerlo. Solo sirve para realizar la secuencia de seguridad en una operación de escritura. Esta seguridad consiste en escribir 55 seguido de AA en EECON2.

LECTURA DE UNA POSICIÓN DE LA EEPROM

Comprende los siguiente pasos:

1) Escritura de la dirección que hay que leer en el registro EEADR.
2) Poner a 1 el bit RD del registro EECON1.
3) Lectura del dato direccionado de esta forma en el registro EEDATA.

El dato está disponible en EEDATA después de colocar RD a 1, por lo que es posible leerlo con la siguiente instrucción.

El siguiente es un ejemplo de **lectura** de la memoria EEPROM de datos:

```
Lectura   MOVFW   Temp       ; Almacena dirección posición a leer
          MOVWF   EEADR      ; Dirección a leer
          BSF     STATUS,RP0 ; Selecciona página 1
          BSF     EECON1, RD ; Activa lectura
Espera    BTFSC   EECON1,RD  ; Espera final lectura
          GOTO    Espera     ;
          BCF     STATUS,RP0 ;Vuelve a página 0, los datos están
                             ; ahora en el registro EEDATA
```

ESCRITURA DE UNA POSICIÓN DE LA EEPROM

La escritura, que es en realidad una programación, es más compleja por razones de seguridad. Hay que realizar el siguiente proceso:

1) Escritura de la dirección en la que se desea escribir el registro EEADR.
2) Escritura del dato en el registro EEDATA.
3) Ejecutar la siguiente secuencia para iniciar la escritura de cada byte

```
MOVLW   55H        ;
MOVWF   EECON2     ; Escribe 55
MOVLW   AAH        ;
MOVWF   EECON2     ; Escribe AA
BSF     EECON1,WR  ; coloca a 1 el bit WR
```


R38

Esta última instrucción inicia el proceso de escritura propiamente dicho. Cuando se termina el bit EEIF está a 1 y, si ha sido activada la interrupción EEPROM haciendo uso del bit EEIE de INCONT, esta interrupción se genera. Mediante programa hay que poner a 0 el bit EEIF.

Escritura de datos en la EEPROM:

```
Escritura   MOVF    TEMP1,W      ;Temp1 almacena la dirección
                                 ; RAM a escribir
            MOVWF   EEADR        ; Mueve el valor al registro
            MOVF    Temp2,W      ;Temp2 contiene el
                                 ;dato a almacenar
            MOVWF   EEDATA       ; Mueve el valor al registro
            BSF     STATUS,RP0   ; Página 1
            BSF     EECON1,WREN  ; WREN = 1  - Activa escritura
            MOVLW   55H          ;Deben emplearse la 4
                                 ;líneas siguientes
            MOVWF   EECON2       ; en la secuencia exacta
            MOVLW   AAH          ;
            MOVWF   EECON2       ;
            BSF     EECON1,W     ; Escribe el dato
Espera      BTFSC   EECON1,WR    ; Espera a que acabe la escritura
            GOTO    Espera       ;
            BCF     STATUS,RP0   ; Página 0
```

Microchip recomienda que se deshabiliten las interrupciones durante la secuencia, añadiendo las siguientes líneas al principio y final de la secuencia:

```
BCF   INTCON,GIE      ; Deshabilita Interrupciones
BSF   INTCON,GIE      ; Habilita Interrupciones
```

Es recomendable disponer de rutinas de escritura y lectura de datos por separado para evitar errores.

Adicionalmente el bit WREN de EECON1 debe ser puesto a 1 para habilitar escritura. Este mecanismo evita escrituras accidentales. El usuario debe controlar que el bit WREN esté siempre a 0 salvo en el momento de la escritura, ya que este bit no se borra automáticamente. Después de inicializada la secuencia de escritura, colocar a 0 el bit WREN no afecta al ciclo de escritura. Cuando el ciclo de escritura se ha completado, automáticamente el bit WR es puesto a 0 y el bit EEIF (*EE Write Complete Interrupt*) a 1. El usuario puede comprobar este bit si es necesario que debe ser puesto a 0 mediante programa.

VERIFICACIÓN DE LA ESCRITURA

Dependiendo de la aplicación, las buenas normas de programación dictan que se debe comprobar que los datos se están escribiendo correctamente, aunque esto no suele ser necesario en la mayoría de las ocasiones. El siguiente trozo de código muestra un ejemplo:

R39

```
        BCF   SATUS,RP0         ; Página 0
        MOVF EEDATA,W           ;
        BSF   STATUS,RP0        ; Página 1
Lectura
        BSF   EECON1,RD         ;
        BCF   STATUS,RP0        ;
;
; Si el valor se ha escrito en W y la lectura en EEDATA, son iguales ?
;
        SUBWF     EEDATA,W      ;
        BTFSS     STATUS,Z      ;
        GOTO      Error_Escritura ;
```

Por ultimo recordar que durante el proceso de programación, además de la grabación del programa puede escribirse el área de datos, para lo cual la información dirigida a la memoria de datos debe ser añadida al fichero *.HEX.

PRACTICAS (VII)

PRESENTACIÓN DE DATOS - II - DISPLAYS DE 7 SEGMENTOS

Como el lector ha podido comprobar, la solución que se dio a la presentación de los datos en 4 dispalys de 7 segmentos, sigue consumiendo gran numero de pines; en el caso del ***Módulo P-01***, basado en el CD 4511 emplea un total de 8 pines.

Cuando es necesario ahorrar el mayor número de pines, puede recurrirse a otra solución que es la que proponemos en este caso. Esta se basa el empleo de un controlador de los displays externo, que se encargue no solamente de la decodificación BCD, sino también del multiplexado de los presentadores. Existen mucho circuitos que realizan esta función, en este caso hemos elegido el **MC14499** de Motorola (a lo largo de la serie de presentará alguna solución mas) circuito no muy costoso. Además del ahorro de pines, el programa se simplifica en gran medida, ya que solo hay que enviar en modo serie al MC14499 las cifras a presentar y no hay que ocuparse de nada mas.

Este circuito puede gobernar hasta 4 presentadores de 7 segmentos consumiendo solo tres pines del PIC. Las líneas son las de DATOS, RELOJ y HABILITACIÓN.

El MC14499 posee una entrada de datos serie, de fácil manejo con un reloj asociado que debe generar el PIC. La transferencia de los datos puede por lo tanto realizarse a la velocidad que elija el usuario, incluso con un reloj irregular.

Si es necesario el empleo de más de 4 presentadores de 7 segmentos, es posible conectar varios MC14499 en cascada.

R40

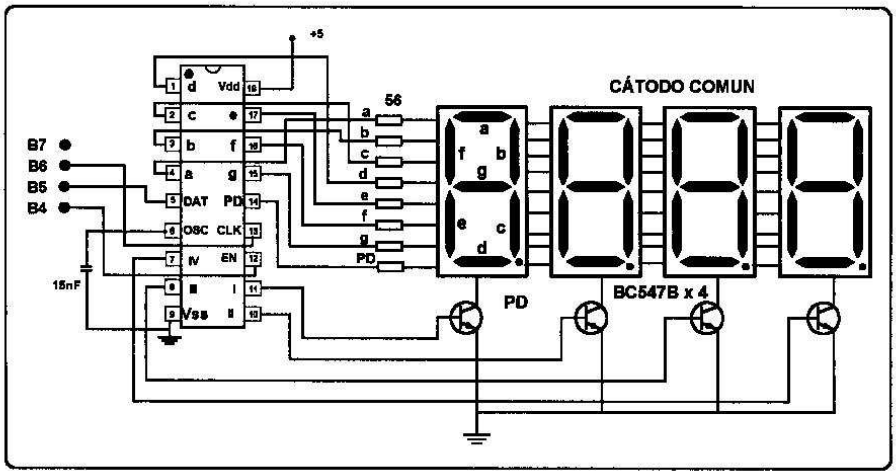

Figura 2.- *Esquela eléctrico del Módulo P-02.*

El dato presente en la línea DAT es memorizado en el circuito en el flanco descendente de la señal de reloj aplicada a CLOCK, ya este constituida esta señal por impulsos positivos o negativos.

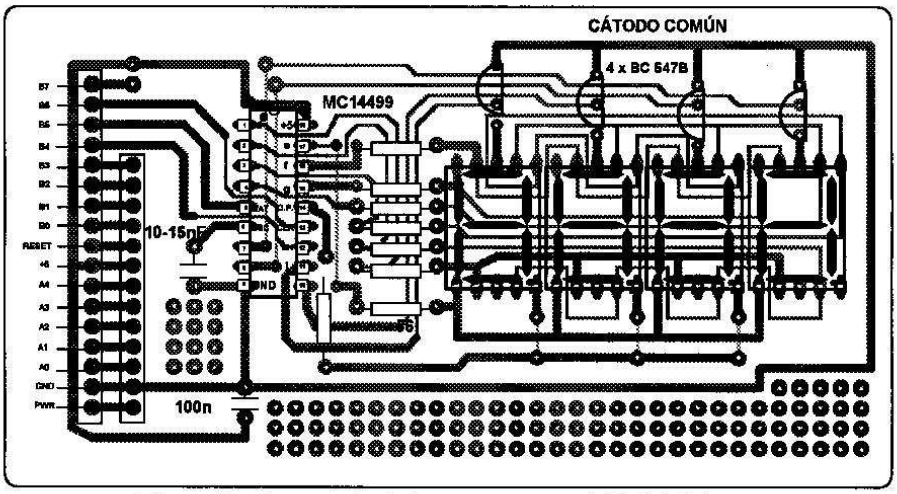

Figura 3.- *Disposición de los componentes del Módulo P-02.*

En la figura 2 se muestra el diseño eléctrico del ***Módulo P-02***, que controla datos codificados en BCD, y en la figura 3 se muestra la disposición de componentes del mismo.

La experiencia de este capítulo, consiste, además de la realización del montaje del ***Módulo P-02***, en un programa que muestra a modo de ejemplo como gobernar el mismo, sacando a cada uno de los cuatro presentadores, un número del 0 al 9.

R4

```
;*****************************************************************************
;Programa: RESIS006.ASM                    Fecha: 23/01/1998                 *
;*****************************************************************************
;  Ejemplo de utilización del integrado MC14499                               *
;                                                                             *
; Revisión: 1.0              Programa para el PIC(16C84/16F84)               *
; Velocidad de reloj: 4Mhz       Reloj instrucción: 1us                      *
;                                                                             *
; Reloj: tipo XT             Perro Guardián: Off                             *
;*****************************************************************************
;*****************************************************************************
;Definiciones                                                                 *
;*****************************************************************************
;Direcciones de los registros de uso específico del PIC 16C84
PORTA       EQU   05h       ;Dirección del puerto A
PORTB       EQU   06h       ;Dirección del puerto B
STATUS      EQU   03h       ;Dirección del registro de estado
TRISA       EQU   05h       ;Dirección del registro de estado para A
TRISB       EQU   06h       ;Dirección del registro de estado para B
;bits del registro STATUS
Z           EQU   02h       ;Bit de cero
C           EQU   00h       ;Bit de acarreo
RP0         EQU   05h       ;Selecciona el segundo banco de registros
;Definiciones del programa
DATOS       EQU   05h       ;Bit por donde se mandan los datos al
RELOJ       EQU   06h       ;Bit utilizado para el reloj del MC14499
ENABLE      EQU   04h       ;Bit utilizado para seleccionar el MC1449
;*****************************************************************************
;Posiciones de memoria utilizadas por el programa                             *
;*****************************************************************************
        ORG   0Ch
Contador1   res   1         ;Utilizada por la rutina de retardo
Contador2   res   1         ;Utilizada por la rutina de retardo
Temp0       RES   1         ;Utilizada para almacenamiento temporal
Temp1       RES   1         ;Utilizada para almacenamiento temporal
DIG12       RES   1         ;Almacena los dígitos 1 y 2
DIG34       RES   1         ;Almacena los dígitos 3 y 4
DEC         RES   1         ;Almacena la configuración para el punto
Semilla     RES   1         ;Número generado para mostrarlo
;*****************************************************************************
;Primera instrucción tras salir del estado de reset
        ORG   00h
        GOTO  INICIO
;*****************************************************************************
;Código correspondiente a los procedimientos utilizados
;
        ORG   05h
;*****************************************************************************
;Produce un retardo
;entrada          nada
;salida           nada
;variables utilizadas: Contador1 y Contador2
```


R42

```
;*************************************************************************
;
RETARDO   CLRF    Contador1    ;Pone a cero el contenido de Contador1
          MOVLW   0fah         ;Almacena en W el valor 0FAh
          MOVWF   Contador2    ;Guarda el valor de W en Contador2
ret       DECFSZ  Contador1,1  ;Decrementa Contador1 y salta si es cero
          GOTO    ret          ;Cerramos el primer bucle
          DECFSZ  Contador2,1  ;Decrementa Contador2 y salta si es cero
          GOTO    ret          ;Cerramos el segundo bucle
          RETURN               ;Salimos de la subrutina
;*************************************************************************************
;
;  Escribe en el MC14499 los 4 dígitos que puede mostrar y los puntos          *
; decimales.                                                                   *
;Entrada:  DIG12, DIG34 -> los pasa al MC14499                                 *
;salida:   nada                                                               *
;variables utilizadas: Temp0                                                  *
;*************************************************************************************
;
MANDA14499  BCF     PORTB,ENABLE   ;Habilitamos al MC14499
            MOVF    DEC,0          ;Pasa a W byte del punto decimal
            MOVWF   Temp0          ;W -> Temp0
            MOVLW   04h            ;Carga 4 en W
            CALL    BYTE14499      ;Escribe un byte en el MC14499
            MOVF    DIG12,0        ;Pasa a W los dígitos 1 y 2
            MOVWF   Temp0          ;W -> Temp0
            MOVLW   08h            ;Carga 8 en W
            CALL    BYTE14499      ;Escribe un byte en el MC14499
            MOVF    DIG34,0        ;Pasa a W los dígitos 3 y 4
            MOVWF   Temp0          ;W -> Temp0
            MOVLW   08h            ;Carga 8 en W
            CALL    BYTE14499      ;Escribe un byte en el MC14499
            BSF     PORTB,ENABLE   ;Deshabilitamos al MC14499
            RETURN
;*************************************************************************************
;
;  Escribe en el MC14499 un byte, genera las señales de reloj y de datos       *
;                                                                              *
;Entrada:  Temp0 -> Byte a transmitir al 14499                                 *
;          W     -> Tamaño del byte (4 ó 8)                                    *
;salida:   nada                                                               *
;variables utilizadas: Temp1                                                  *
;*************************************************************************************
;
BYTE14499  MOVWF   Temp1          ;Guarda contenido de W en Temp0
By14499    BCF     PORTB,DATOS    ;Pone a cero el bit de datos
           RLF     Temp0,1        ;Pasa a C el bit de datos actual
           BTFSC   STATUS,C       ;Mira si C es uno ó cero
           BSF     PORTB,DATOS    ;Pone a uno el bit de datos actual
           BSF     PORTB,RELOJ    ;Genera el ciclo de reloj
           NOP                    ;para el MC14499 y así escribir
           BCF     PORTB,RELOJ    ;el bit en el MC14499
           DECFSZ  Temp1,1        ;Lo hacemos tantas veces como
                                  ; bits a transmitir
           GOTO    By14499        ;Cierra el bucle
           RETURN                 ;Regreso de la subrutina
```


R43

```
;*******************************************************************************
;
;Incrementa Semilla una unidad.
;entrada: Nada
;salida:  En Semilla se encuentra un número del 1 al 99 (En BCD)
;variables utilizadas: Semilla
;*******************************************************************************
;
IncSemilla    INCF      Semilla,1     ;Incrementa en una unidad Semilla
              MOVLW     0fh           ;Carga en W 0fh
              ANDWF     Semilla,0     ;Nos quedamos con los 4bits bajos
              SUBLW     0Ah           ;Vemos si es un número de 0 a 10
              BTFSC     STATUS,Z      ;Si es 10 se pone acero los 4bits bajos
              GOTO      IncSemiAc     ;y se incrementan los 4bits altos
              GOTO      IncSemiTo     ;de lo contrario vemos si ha llegado al
                                      ; número máximo
IncSemiAc     MOVLW     010h          ;Carga 010h en W
              ADDWF     Semilla,1     ;Suma a los 4bits altos de Semilla 1
              MOVLW     0F0h          ;Carga 010h en W
              ANDWF     Semilla,1     ;Ponemos los 4bits bajos a cero
IncSemiTo     MOVF      Semilla,0     ;Carga W con el valor de Semilla
              SUBLW     099h          ;Resta W el número máximo
              BTFSS     STATUS,Z      ;Testamos el bit de cero
              RETURN                  ;Si no es máximo sale de la
                                      ; subrutina (Z=0)
              MOVLW     01h           ;Si es el máximo Semilla toma el valor 1
              MOVWF     Semilla
              RETURN                  ;Salimos de la subrutina
;*******************************************************************************
;Código correspondiente al programa principal                                   *
;*******************************************************************************
;
INICIO        BSF       STATUS,RP0    ;Seleccionamos el segundo banco
              CLRF      TRISB         ;El puerto B como salida
              BCF       STATUS,RP0    ;Seleccionamos el primer banco
              BSF       PORTB,ENABLE  ;Deshabilitamos al MC14499
              BCF       PORTB,RELOJ   ;Deshabilitamos la línea de reloj
              CLRF      DIG12             ;Ponemos a 0 los dígitos 1,2,3 y 4
              CLRF      DIG34
              CLRF      DEC           ;Los puntos decimales a cero

CicloNuevo    SWAPF     DEC,1         ;Intercambia los 4 bits del byte
              CALL      MANDA14499    ;Actualiza valor a mostrar en el MC14499
              CALL      IncSemilla    ;Incrementa una unidad semilla
              MOVF      Semilla,0     ;Carga en W el contenido de semilla
              MOVWF     DIG12         ;Carga en DIG12 el contenido de W
              MOVWF     DIG34         ;Carga en DIG34 el contenido de W
              SWAPF     DEC,1         ;Intercambia los 4 bits del byte
              INCF      DEC,1         ;Incrementa en 1 el registro DEC
              CALL      RETARDO       ;Produce un retardo
              CALL      RETARDO       ;Produce un retardo
              GOTO      CicloNuevo    ;Inicia un nuevo ciclo

        END
```


R44