LABORATORIO Nº2
DIRECCIONAMIENTO INDIRECTO, VISUALIZACION ESTROBOSCOPICA Y MANEJO DE TABLAS

I. OBJETIVOS

- ✓ Manejar el set de instrucciones de los microcontroladores PIC mediante diversos problemas propuestos.
- ✓ Emplear la técnica de visualización estroboscópica para displays 7 segmentos.
- ✓ Construir circuitos que se relacionen con las actividades cotidianas de los integrantes del grupo.
- ✓ Implementar aplicaciones sencillas y verificar su funcionamiento en condiciones reales.

II. MATERIAL DE APOYO

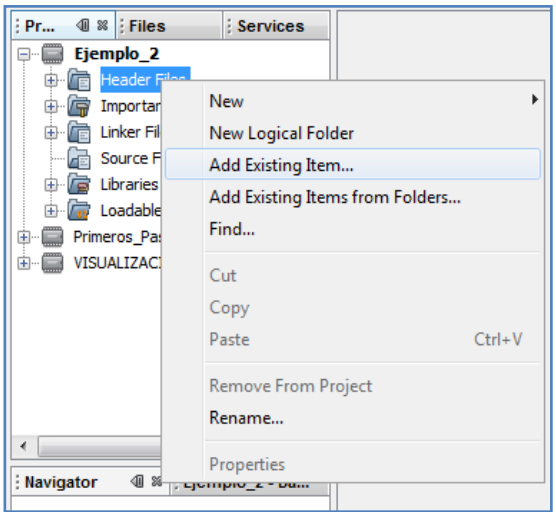
- ➢ Ordenador con las aplicaciones PROTEUS Y MPLAB X.
- ➢ Microcontrolador de la serie PIC16F88X.
- ➢ Programador de microcontroladores PIC.
- ➢ Protoboard
- ➢ Resistencias, Transistores, Switch, pulsadores, displays 7seg, etc.
- ➢ Fuente de alimentación 5V

III. EQUIPOS NECESARIOS

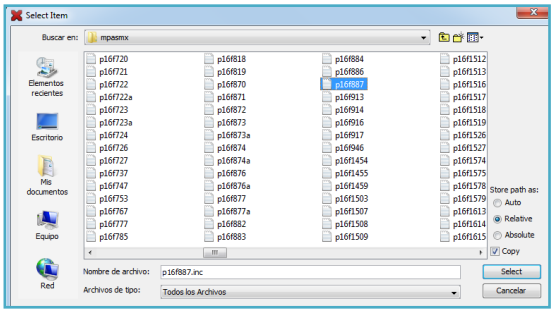
- ➢ PC y Software de simulación.

IV. EJEMPLO

Una vez creado el proyecto es necesario incluir la librería del pic a trabajar (PIC16F887) y de esta manera no definir por comandos las posiciones de los registros. Dando click derecho sobre *Header Files* se selecciona *Add Existing Item…*



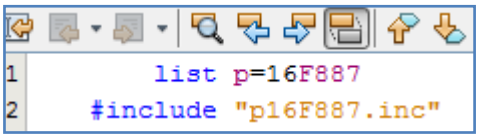
Las librerías de los microcontroladores se pueden encontrar accediendo a la carpeta donde quedó instalado el MPLAB, en este caso será *C:/ Archivos de programa/ Microchip/ MPLAB/ mpasmx/* al llegar a esta carpeta se busca la referencia del pic, en este caso ***p16f887.inc***



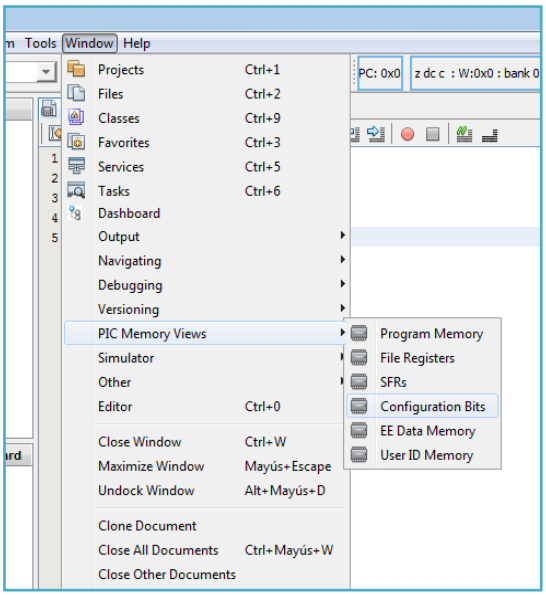
Se dejan seleccionadas las opciones *Relative* y *Copy* y se da click en *Select.*

Se podrá ver toda la descripción de la librería seleccionada anteriormente clickeando en ***p16f887.inc***

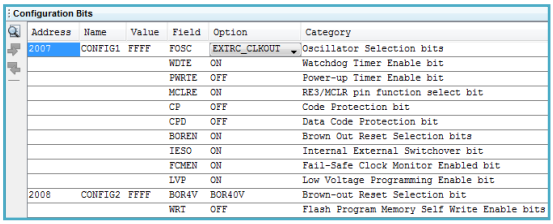
Es necesario añadir la librería del pic en el código de la siguiente manera:



Otra forma de hacer la configuración de bits de control se puede hacer vía *Window/ PIC memory views/ configuration bits*



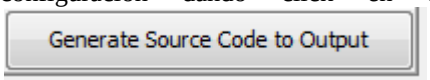
Se desplegará la siguiente ventana.



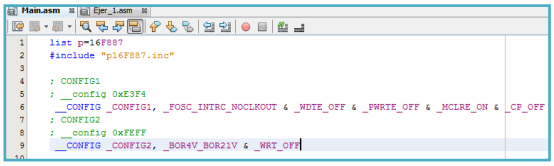
En ella se definirá el tipo de configuración que se requiere.
La configuración deberá quedar de la siguiente manera:

Name	Value	Field	Option
CONFIG1	E3F4	FOSC	INTRC_NOCLKOUT
		WDTE	OFF
		PWRTE	OFF
		MCLRE	ON
		CP	OFF
		CPD	OFF
		BOREN	ON
		IESO	OFF
		FCMEN	OFF
		LVP	OFF
CONFIG2	FEFF	BOR4V	BOR21V
		WRT	OFF

Ahora se genera el código de la palabra de configuración dando click en el botón



Este código deberá ser copiado en el archivo principal como se muestra a continuación.



- Direccionamiento indirecto.

El direccionamiento indirecto es posible gracias a la manipulación del registro *INDF* el cual no es un registro físico, y el *File Select Register* (*FSR)* el cual actúa como apuntador.

Este tipo de direccionamiento se usa por lo general para limpiar posiciones de memoria o para la asignación de datos de manera secuencial en RAM.

Un ejemplo de inicialización de variables por medio de direccionamiento indirecto se puede ver a continuación.

Primero se le asigna un valor de inicio a *FSR* el cual es de *0X20* (inicio de los registros de propósito general - variables). En el momento que se limpia el registro *INDF* se está limpiando la variable que está en la posición *0X20*; de

inmediato se incrementa el apuntador *FSR*. Esta tarea se repetirá nueve veces consecutivamente limpiando así las variables que están desde la posición *0X20* hasta *0X29.*

```
INI_REG:
    MOVLW   0X20    ;Asignacion de inicio
    MOVWF   FSR     ;al apuntador FSR
    CLRF    INDF    ;Limpieza del registro INDF
    INCF    FSR     ;Incremento del apuntador FSR
    MOVLW   0X29    ;Incrementara 0x9 posiciones de RAM
    XORWF   FSR,W   ;
    BTFSS   STATUS,Z;
    GOTO    $-5     ;
    RETURN
```

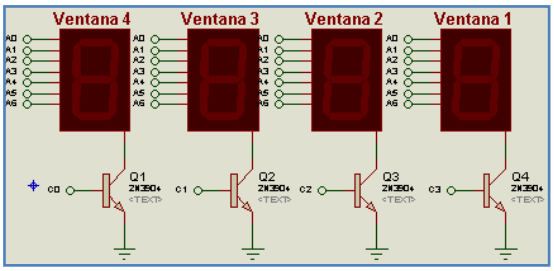
En resumen, el *FSR* sirve para determinar qué posición de memoria se quiere manipular y el *INDF* es el registro a modificar el cual es análogo a la posición de memoria apuntada por el *FSR*.

- Visualización estroboscopica.

La visualización estroboscopica se basa en hacer encender un led a una frecuencia muy rápida con el fin de que el ojo humano no identifique que hay un parpadeo, de esta manera se tiene el efecto del led encendido sin intermitencias.

❖ Ejemplo de visión estroboscopica con cuatro displays 7 segmentos.

Los displays tienen que estar conectados al mismo puerto de salida del pic. En este caso el dato que pasara por los displays tendrá salida por el puerto A y el puerto de control será el puerto C.



Se envia un dato y dependiendo en que display se quiere visualizar se manda el dato al puerto de control (*PORT C*).

Luego de configurar los bits y llamar la librería del procesador a usar, nos disponemos a empezar con el nuevo proyecto.

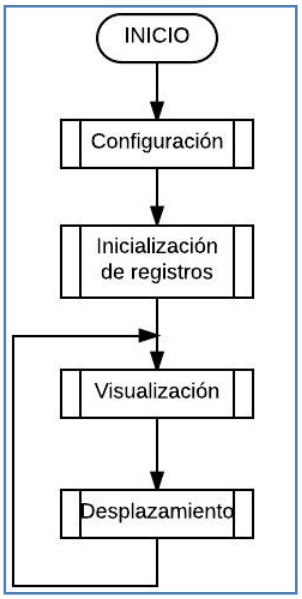
Dado que tenemos dos displays se crearán cuatro variables que los representarán.

```
CBLOCK 0X20
    VENTANA : 4
ENDC
```

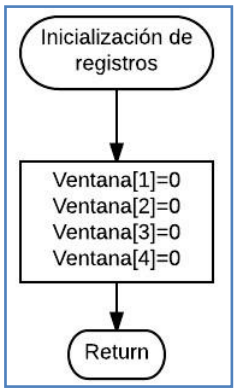
Se hace la respectiva configuración de puertos de entrada/ salida y se selecciona la frecuencia de oscilador que se usará, en este caso será:

```
CONFIGURACION:
    BANKSEL OSCCON
    MOVLW   0XF1        ;Fosc=8Mhz
    MOVWF   OSCCON

    BANKSEL TRISA
    CLRF    TRISA       ;SELECCION DE DISPLAY
    CLRF    TRISC       ;PUERTO CONTROL
    BANKSEL ANSEL
    CLRF    ANSEL

    BANKSEL PORTA
    CLRF    PORTC
    CLRF    PORTA

    RETURN
```

Para tener una idea del funcionamiento del programa a desarrollar se presenta el siguiente diagrama de flujo:



Para la inicialización de registros se recomienda usar direccionamiento indirecto.



Para este ejemplo se clarearon las primeras cuatro variables las cuales son las encargadas de representar el número de displays.

```
INI_REG:
    MOVLW   VENTANA
    MOVWF   FSR
    CLRF    INDF
    INCF    FSR
    MOVLW   0X25
    XORWF   FSR,W
    BTFSS   STATUS,Z
    GOTO    $-5
    RETURN
```

Para la rutina de visualización se tiene en cuenta el orden del diagrama de flujo que la describe. En éste se quiere tener un tiempo de visión de 5ms en cada display antes de desplazar el mensaje; y se visualizara 25 veces el mensaje en la misma posición. De esta manera se verá el desplazamiento del mensaje cada 0,5s.

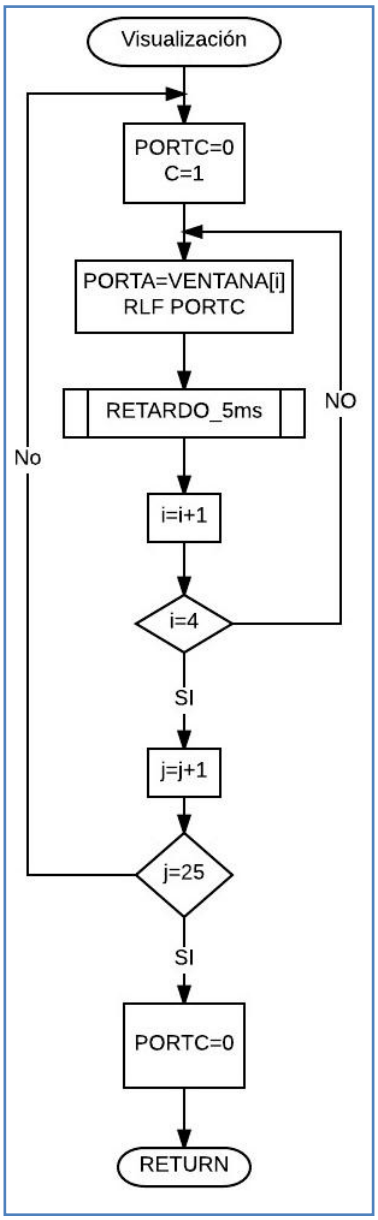
$$T_{desplazamiento} = (0.005)(i)(j)$$

$$T_{desplazamiento} = (0.005)(4)(25)$$

$$T_{desplazamiento} = 0.5s$$

Donde *i* es el número de displays.

Dado que el PORTC es el puerto de control, cuyos bits a usar serán los 4 menos significativos o el nibble bajo, es necesario clarearlo. Inmediatamente se pone en alto el estado del carry (C), con el fin de permitir la visualización del mensaje en un display a la vez; usando la instrucción de rotación *RLF*.



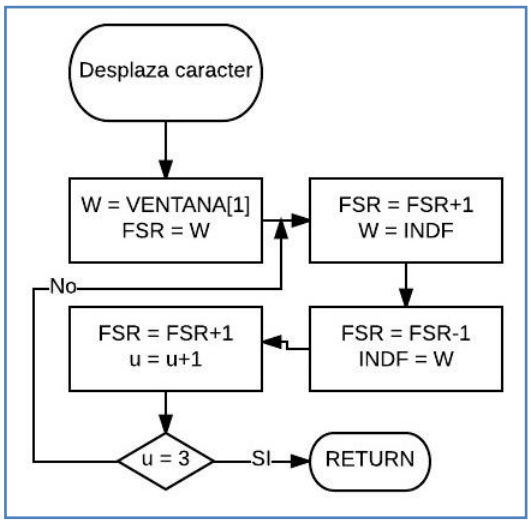
A continuación se puede ver el código desarrollado para la rutina de visualización. En este se hace la implementación de las variable I y J las cuales son las encargadas de los tiempo de en cada display.

```
VISUALIZAR:
    CLRF    PORTC       ;PORTC = 0
    BSF     STATUS,C    ;Carry = 1
    MOVLW   VENTANA     ;Carga de la direccion
    MOVWF   FSR         ;de la variable VENTANA
    MOVF    INDF,W      ;Carga del mensaje
    MOVWF   PORTA       ;para que salga por PORTA
    RLF     PORTC       ;Rotacion del carry
    CALL    RETARDO     ;Retardo 5ms
    INCF    FSR         ;Cambio de VENTANA (display)
    INCF    I           ;I ha incrementado 4 veces?
    MOVLW   0X04        ;
    XORWF   I,W         ;
    BTFSS   STATUS,Z    ;
    GOTO    $-9         ;
    CLRF    I           ;
    INCF    J           ;
    MOVLW   .25         ;Repeticion de visualizaci?n
    XORWF   J,W         ;para que se vea 0.5 seg
    BTFSS   STATUS,Z    ;el mensaje sin desplazamiento
    GOTO    VISUALIZAR  ;alguno
    CLRF    J
    CLRF    PORTC       ;
    RETURN
```

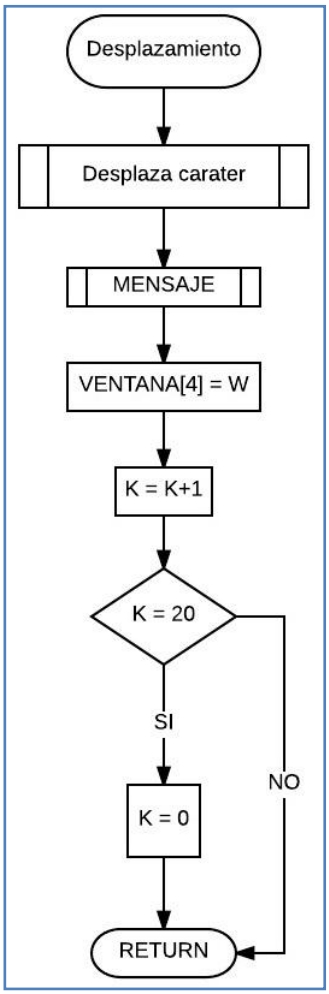
Ahora para la rutina de desplazamiento se seguirá la lógica del diagrama de flujo inmediatamente siguiente.

Dado que el desplazamiento en un publick es de derecha a izquierda, en este caso de la ventana 1 a la ventana 4, el caracter del mensaje se cargará en la ventana 4 y se irá desplazando hacia la ventana 1 y gracias al puerto de control se tendrá el efecto de desplazamiento que se quiere.

De esta manera es necesario cargar lo que está en la ventana 4 a la ventana 3, lo que está en la ventana 3 en la ventana 2 y lo que esta en la ventana 2 en la ventana 1, **de allí sale la variable u**, tres movimientos de ventana a ventana.



Teniendo en cuenta el tamaño del mensaje, en este caso de 20 caracteres, se crea una variable K, la cual desplazara el puntero dentro de una tabla en que se encuentra el mensaje obteniendo así cada posición de éste.



A continuación se puede ver el código desarrollado siguiendo el diagrama de flujo inmediatamente anterior.

```
DESPLAZAR:
    CALL    DESPLAZA_CARACTER
    MOVF    K,W         ;llamado a tabla de mensaje
    CALL    MENSAJE     ;en la posicion K
    MOVWF   INDF        ;
    MOVLW   .20         ;Tamaño del mensaje
    XORWF   K,W         ;
    BTFSC   STATUS,Z    ;
    GOTO    $+3
    INCF    K
    RETURN
    CLRF    K
    RETURN
```

```
DESPLAZA_CARACTER:
    MOVLW   VENTANA      ;Carga de direccion
    MOVWF   FSR          ;de VENTANA en FSR
    INCF    FSR          ;FSR + 1
    MOVF    INDF,W       ;w = INDF
    DECF    FSR          ;FSR - 1
    MOVWF   INDF         ;INDF = W
    INCF    FSR          ;FSR + 1
    INCF    U            ;U ha incrementado 3 veces?
    MOVLW   0X03         ;
    XORWF   U,W          ;
    BTFSS   STATUS,Z
    GOTO    DESPLAZA_CARACTER+2
    CLRF    U
    RETURN
```

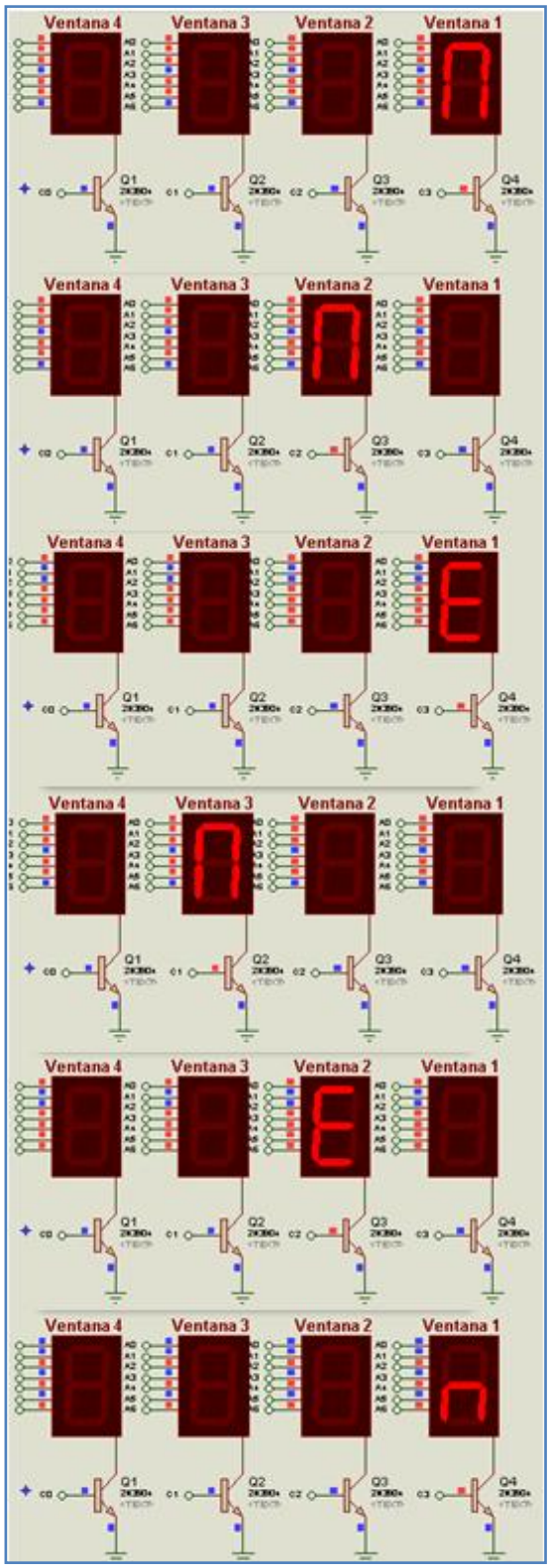
En la rutina de MENSAJE solo se tiene una tabla con 20 números que representan un carácter. El mensaje a visualizar es "MENSAJE_PRUEBA_1234_".

```
MENSAJE:
    ADDWF   PCL,F
    DT      0X37, 0X79, 0X54, 0X6D, 0X77, 0X1F, 0X79, 0X08, 0X73, 0X50
    DT      0X1C, 0X79, 0X7C, 0X77, 0X08, 0X06, 0X5B, 0X4F, 0X66, 0X08
```

De esta manera ya podemos armar nuestro programa con cada rutina creada anteriormente.

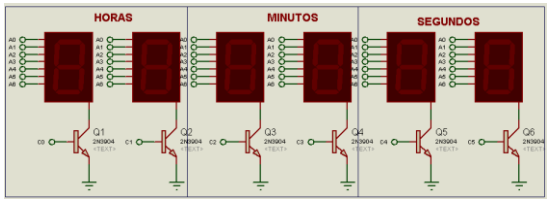
```
    ORG .0
    CALL    CONFIGURACION
    CALL    INI_REG
INICIO:

    CALL    VISUALIZAR
    CALL    DESPLAZAR
    GOTO    INICIO


    #INCLUDE"CONFIGURACION.INC"
    #INCLUDE"INI_REG.INC"
    #INCLUDE"VISUALIZAR.INC"
    #INCLUDE"DESPLAZAR.INC"
    #INCLUDE"MENSAJE.INC"
    #INCLUDE"RETARDOS.INC"
    END
```

Para la visualización en la simulación se recomienda que el valor de J sea igual a 2 o 3, dado que el simulador presenta problemas con este tipo de visualización a alta frecuencia.



V. DESCRIPCIÓN DEL LABORATORIO

LABORATORIO N°2: Temporizador. Se deben tener 5 tiempos de temporización fijos los cuales serán escogidos por medio de un dipswitch; estos tiempos tienen que ser diferentes entre sí, mayores a 50 segundos y menores o iguales a 5 minutos. Una vez cada conteo llegue a cero, éste no debe detenerse, es decir, seguirá contando negativamente (-1,-2,-3); esto se hará hasta que el "usurario" desactive mediante un pulsador dicho conteo. Cada vez que el temporizador sea desactivado deberá salir un aviso tipo publik (mínimo 30 caracteres). Se debe tener un mensaje para cada tiempo de temporizador. En el momento que la temporización y su mensaje terminen, el sistema debe entrar en un estado de espera, se deberá visualizar el mensaje "ESPERA" en los displays hasta que sea cargado otro valor a temporizar. Usar 6 displays 7 segmentos, direccionamiento indirecto y tablas.



VI. BIBLIOGRAFIA

[1]. Microcontroladores dsPIC Diseño práctico de aplicaciones. Tercera Edición. J. Mª. Angulo Usategui y I. Angulo Martínez. Editorial McGraw Hill, 2007

[2]. Microcontroladores PIC Diseño práctico de aplicaciones. Tercera Edición. J. Mª. Angulo Usategui y I. Angulo Martínez. Editorial McGraw Hill, 1999

[3]. Microcontroladores PIC. La clave del diseño. E. Martín Cuenca , J. Mª. Angulo Usategui y I. Angulo Martínez. Editorial Thomson

[4]. Microcontroladores PIC, la solución en un chip J. Mª. Angulo Usategui, E. Martín Cuenca y I. Angulo Martínez. Editorial Paraninfo, 2000

[5]. Microcontrolador PIC16F84. Desarrollo de proyectos. PALACIOS, E.- REMIRO, F. y LÓPEZ, L.J. Febrero 2004. Rústica y CD-ROM, 648 Págs..

[6]. Designing Embedded Systems with PIC Microcontrollers Principles and applications. Tim Wilmshurst. 2007. Elsevier