Escuela Politécnica Superior
Universidad de Huelva

TERCER CURSO. INFORMÁTICA INDUSTRIAL II

Técnicas de programación de los microcontroladores PIC de Microchip

Manuel Sánchez Raya
Versión 0.9
24 de Febrero de 2002

ÍNDICE

BIBLIOGRAFÍA:

Microcontroladores de Bajo Costo. Familia PIC 15C5X
Antonio Bueno Juan. Editorial Ediatec

1.- Lectura de Tablas.

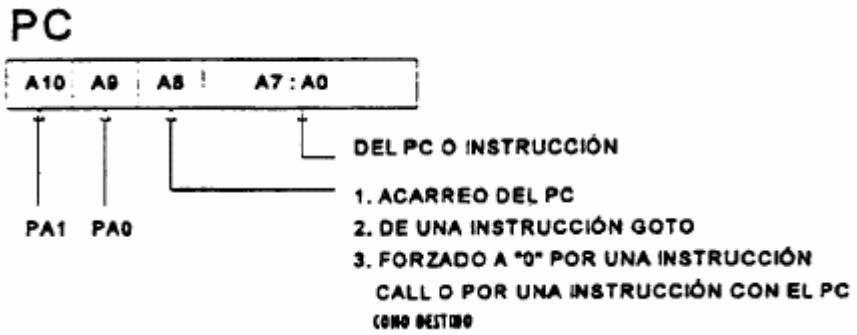
Un gran número de programas necesita almacenar en la memoria de programa tablas de datos que más tarde utilizan, por ejemplo, para trasformar números en BCD en el formato de siete segmentos. A continuación se estudia como se puede realizar esto en los microcontroladores de la familia PIC16C5X.

1.1.- Estructura del contador de programa.

En los PIC16C5X, el tamaño de la página de programa es de 512 palabras. Dependiendo del circuito, la memoria de programa puede tener hasta 2K palabras (en los circuitos PIC16C57 ó PIC16C58). El PC y el STACK tienen un ancho comprendido entre 9 y 11 bits dependiendo de la máxima dirección que deben almacenar. A continuación se puede ver una tabla que muestra el ancho del PC y el STACK.

Microcontrolador	Ancho (bits)		Memoria de Programa (Words)
	Contador de programa	Stack	
PIC 16C52	9	9	384
PIC 16C54/55	9	9	512
PIC 16C56	10	10	1 K
PIC 16C57/58	11	11	2K
PIC 16F84	13	13	1K
PIC 16F876	13	13	8K

Los 8 bits de menor peso son accesibles por el programa de usuario. Estos bits están contenidos en el registro PC. El bit A8 es forzado a 0 por una instrucción CALL o por una instrucción cuyo destino es el PC, por lo que la dirección de comienzo de las subrutinas debe estar en las primeras 256 palabras de cada página de memoria de programa. Los bits A9 y A10 son cargados con PA0 y PA1 respectivamente (contenidos en STATUS) cuando se ejecuta una instrucción CALL. La estructura del PC, así como las partes que lo componen, se muestran en la figura siguiente:



1.2.- Implementación de Tablas.

La implementación de tablas se realiza con ayuda de la instrucción RETLW. Esta instrucción al mismo tiempo que retorna de una subrutina, carga una constante de 8 bits en el registro W. Una vez obtenido el dato, el programa principal se encarga de recuperarlo del registro W y manipularlo como crea conveniente. El ejemplo siguiente muestra cómo obtener un byte de una tabla de datos almacenados en serie a partir de la dirección 000h en el PIC16C54.

```
            ORG   0           ;Almacén de la serie de números

            RETLW 0FFh
            RETLW 0FEh
            RETLW 0FDh
            RETLW 0FCh
            RETLW 0FBh
            RETLW 0FAh
            RETLW 0F9h
            RETLW 0Flh        ;Fin de la serie de números

main_prog   ORG   123h        ;Programa principal
            MOVLW byte_num    ;byte-num = 0
                              ;para el primer byte
            CALL  get_1byte

get_1byte   MOVWF PC          ;Escribe W en el PC
                              ;donde W = offset
                              ;W = 0 para el primer byte
                              ;Final de la subrutina.
END
```

El realizar este sistema de implementación de tablas imposibilita la implementación de más tablas. Para poder realizar más tablas, éstas se pueden implementar como se muestra en el listado siguiente, pero siempre deben estar situadas en las primeras 256 posiciones de cada página de memoria de programa. Si la tabla invade otra área de memoria, no se podrá acceder a los datos almacenados en estas posiciones de memoria, y por tanto la tabla no cumplirá el efecto deseado.

```
main_prog   ORG   123h        ;Programa principal
            MOVLW byte_num    ;byte-num = 0
                              ;para el primer byte
            CALL  get-1byte

            ORG   023h        ;Dentro de las primeras
                              ;256 palabras de la página
get_1byte   ADDWF PC          ;W = offset

            RETLW 0FFh
            RETLW 0FEh
            RETLW 0FDh
            RETLW 0FCh
            RETLW 0FBh
            RETLW 0FAh
            RETLW 0F9h
            RETLW 0Flh        ;Fin de la serie de números
END
```

2.- Gestión de la Pila por Software.

La familia de microcontroladores PIC16C5X dispone de tan sólo dos niveles de pila (dos posiciones de almacenamiento de la dirección de retorno). Esto tiene como resultado que sólo puedan realizarse dos llamadas concatenadas a subrutinas, o sea, sólo puede llamarse una subrutina cuando nos encontramos atendiendo a la llamada de una subrutina. Sin embargo en un buen número de programas es necesario realizar concatenación de llamadas a subrutinas con un número muy superior a dos. Esta aplicación se puede usar para implementar una pila por software que proporcione mayor número de niveles de concatenación en las subrutinas.

Puesto que el tamaño de memoria RAM de que dispone el microcontrolador utilizado tiene límite, hay que ser prudente y determinar el máximo número de niveles que se necesitan y definir la longitud adecuada de la pila.

2.1.- Implementación de la Pila.

En esta aplicación se implementa una pila con 5 niveles, por lo que se pueden anidar hasta 5 llamadas de subrutina sin rebasar la pila. Se define la macro NCALL, la cual sustituye a la instrucción CALL cuando se llama a una subrutina. La rutina NCALL, almacena el valor de retorno del PC en la pila de RAM y ejecuta la subrutina llamada. Como finalización de la subrutina debe usarse la instrucción GOTO RETURN en lugar de la instrucción RETLW K. RETURN es una rutina que repone la dirección de retorno de la pila de RAM y continúa con el flujo normal del programa.

Como la pila diseñada utiliza el registro FSR y el direccionamiento indirecto, el usuario debe restaurar el valor "original" del FSR si se utiliza en cualquier otro lugar del programa. Las rutinas descritas en esta aplicación trabajan sólo si las rutinas llamadas se encuentran entre las primeras 256 palabras de la primera página. Si se desea saltar a otras páginas, como en el caso del PIC16C57, debe salvarse el byte de STATUS al mismo tiempo que el PC.

```
;*****************************************************************
;   PILA IMPLEMENTADA POR SOFTWARE
;*****************************************************************

LIST P=16C54, F=INHX8M

;*****************************************************************
;     pila.asm
;     Rutina de demostración de implementación de pila capaz de
albergar
;     más de dos llamadas a subrutina anidadas.
;     Esto es una demostración, por lo que se ha colocado la
instrucción NOP
;     donde debería aparecer el cuerpo de la subrutina.
;
;*****************************************************************

W           EQU   0     ;Almacena en W
INDI        EQU   0     ;Direccionamiento Indirecto
PC          EQU   2     ;Contador de programa
FSR         EQU   4     ;Puntero de la pila
STACK       EQU   8     ;Principio de la Pila

;*************************************************************
;     Las proximas 5 posiciones de RAM debe reservarse
;     para la pila. No se debe usar ninguna posición
```

```
;     entre la 8 hasta la 12 en decimal
;*****************************************************************

      ORG   01FF
      GOTO  INIT  ;Vector de inicio del programa

      ORG   0

INIT  MOVLW STACK ;Carga "STACK" en el puntero indirecto
      MOVWF FSR
      GOTO  START ;Comienzo del programa

;*****************************************************************
;     Define NCALL como una macro que se usa en lugar
;     de la instrucción CALL
;*****************************************************************

NCALL MACRO LABEL
      MOVF  PC,W  ;Almacena PC en la pila
      MOVWF INDI  ;Direccionamiento indirecto
      INCF  FSR   ;Incrementa el puntero de la pila
      GOTO  LABEL
ENDM

;*****************************************************************
;     Define el retorno de la subrutina llamada por NCALL
;*****************************************************************

RETURN      DECF  FSR         ;Apunta a la última posición ocupada de
                              ;la pila
            MOVLW 3           ;Obtiene la última dirección apuntada por
            ADDWF INDI,W      ;FSR y le suma 3
            MOVWF PC          ;Continua la ejecución en la dirección
                              ;anteriormente calculada

;*****************************************************************
;     PROGRAMA PRINCIPAL
;*****************************************************************

START       NOP                     ;Cuerpo del programa
            NOP
            NCALL JUAN
            MOVF        PC,W        ;Almacena PC en la pila
            MOVWF INDI              ;Direc. Indirecto
            INCF        FSR         ;Incrementa el puntero de la pila
            GOTO        JUAN        ;Salta a la subrutina
            NOP
            NOP
            SLEEP                   ;Pasa al estado de dormido

;*****************************************************************
;     SUBRUTINA JUAN
;*****************************************************************

JUAN        NOP                     ;Cuerpo de la subrutina JUAN
            NCALL LUIS
            MOVF        PC,W        ;Almacena PC en la pila
            MOVWF INDI              ;Direc. Indirecto
            INCF        FSR         ;Incrementa el puntero de la pila
            GOTO        LUIS        ;Salta a la subrutina
            NOP
```

```
            GOTO        RETURN

;*****************************************************************
;     SUBRUTINA LUIS
;*****************************************************************

LUIS        NOP                     ;Cuerpo de la sub. LUIS
            NCALL MARIA
            MOVF        PC,W        ;Almacena PC en la pila
            MOVWF INDI              ;Dir. Indirecto
            INCF        FSR         ;Incrementa el puntero de la pila
            GOTO        MARIA       ;Salta a la subrutina
            NOP
            GOTO        RETURN

;*****************************************************************
;     SUBRUTINA   MARIA
;*****************************************************************

MARIA       NOP                     ;Cuerpo de la sub. MARIA
            NOP
            GOTO        RETURN

END
```

3.- Llamadas de larga distancia.

Esta aplicación explica cómo implementar llamadas a subrutinas que se encuentran en cualquier parte de la memoria de programa, incluidas las 256 palabras de mayor peso de las páginas. Para poder realizar estas "llamadas de larga distancia" es necesario ubicar una pequeña parte del código (vectores) en las 256 palabras de menor peso de la página correspondiente.

Para comprender mejor el método, es conveniente repasar la estructura del contador de programa y recordar estos tres conceptos importantes:

1. Una instrucción CALL carga el contenido completo del PC en el STACK.
2. Una instrucción GOTO no modifica el STACK.
3. Una instrucción GOTO puede saltar a cualquier localización de una página de memoria.

También debe recordarse que para seleccionar una página determinada, los bits PAl y PA0 (STATUS <6:5>) deben ser programados de acuerdo con la página deseada. Estos bits no serán cargados en los bits del PC A10:A9 hasta que ocurra uno de los sucesos siguientes:

1. Una instrucción CALL.
2. Una instrucción GOTO.
3. Una instrucción que modifique el PC (PC<7:0>), como ADDW PC,F.

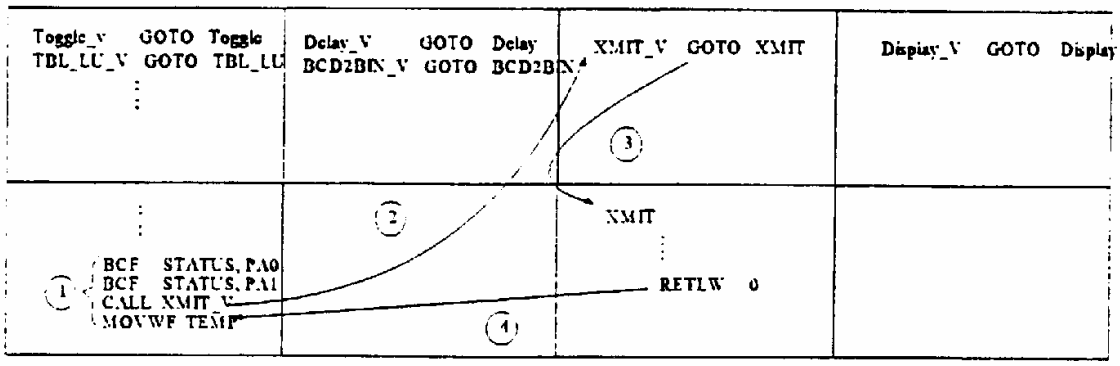
Así que un instrucción CALL seguida por una instrucción GOTO, la cual siempre se mantiene en la misma página, se comporta como la llamada que se desea realizar,

3.1.- Implementación.

Esto se consigue colocando "vectores de llamada" en las primeras 256 palabras de cada página. De manera que la llamada de la subrutina accede hasta el vector, el cual se encarga de ejecutar la subrutina (GOTO) que se puede encontrar en cualquier parte de la página. La instrucción de

retorno de subrutina (RETLW), pone el contenido de la pila en el contador de programa y continúa con la instrucción siguiente a la que realizó la llamada a la subrutina.

A continuación puede verse una representación gráfica de cómo se realiza una llamada de larga distancia.



La secuencia ocurrida en la figura anterior es la siguiente:

1. Selecciona la página de memoria de programa de la subrutina que desea y ejecuta la llamada a la subrutina.

2. El programa carga la pila con la dirección PC+ 1, y salta a la página seleccionada y dirección especificada del "vector de llamada" (debe estar entre las primeras 256 posiciones de la página).

3. Ejecuta la instrucción GOTO para poder acceder a toda la página de memoria. Entonces ejecuta la subrutina.

4. Ejecuta la instrucción REXW, la cual pone el valor que tiene la pila en el PC. Esto produce que el programa continúe con la ejecución de la instrucción posterior a CALL.

El uso de "llamadas de larga distancia" se puede usar para colocar todas las subrutinas en la(s) páginas seleccionada(s), pudiendo contener toda la página las subrutinas (no estando restringido su uso a la primera mitad de la página). La colocación de las subrutinas en menor número de páginas de programa puede reducir el encabezamiento que por especificación requieren las páginas, por lo que se cambiarán menos frecuentemente.

Con el uso del ensamblador MPASM se puede realizar fácilmente la verificación de que los vectores y las subrutinas se encuentran en la misma página de programa. A continuación se muestra el uso de las directivas para imprimir avisos definidos por el usuario o mensajes de error en el fichero lista. Éstos se muestran como estamentos condicionales sombreados. Estos mensajes son impresos sólo en el fichero lista, y no se muestra ninguna indicación de este mensaje en el momento del ensamblado.

```
P1-TOP      EQU   0x0000      ; Primera dirección en la Página 0
P2-TOP      EQU   0x0200      ; Primera dirección en la Página 1
P3-TOP      EQU   0x0400      ; Primera dirección en la Página 2
P4-TOP      EQU   0x0600      ; Primera dirección en la Página 3
RESET-V     EQU   0x07FF      ; Vector de reset.

            ORG   P1TOP

            ;
```

```
            ;

            ORG   P3TOP
            ;
My_sub_v    GOTO  My_sub      ; Vector de la subrutina My_sub.

            ;
            ;

My_sub                        ; Subrutina My_sub.
            ;
if ( (My_sub_v & 0x600) # (My_sub & 0x600))
MESSG “ERROR - VECTOR y SUBRUTINA no se encuentran en la misma página”
endif


My_sub_FIN  RETURN
            ;
if ( (My-sub-v & 0x600) # (My-sub-FIN & 0x600))
MESSG “AVISO - La subrutina cambia de página’’
endif
            ;
            ;
            ORG   RESET-V     ; Dirección de memoria de programa
                              ; para el vector de reset.

            GOTO  START       ; Salta al comienzo del programa.
```

El uso de “llamadas de larga distancia” puede facilitar el desarrollo de programas de aplicación. Con muy poca utilización del inicio de la página, se puede ejecutar subrutinas ubicadas en cualquier lugar de la memoria del programa, y regresar a la posición deseada. Esto facilita el desarrollo de programas de aplicación, por eliminar el mapeado de las subrutinas en las primeras 256 palabras de cada página de memoria de programa. El uso de “llamadas de larga distancia” es posible en cualquiera de los PIC16C5X, pero es más utilizado en los circuitos de más de una página. El los circuitos de más de una página de memoria de programa, las directivas del ensamblador pueden utilizarse para verificar que las subrutinas están en la página de memoria de programa adecuada.

4.- Técnicas de Interrupción por software.

Esta aplicación describe el único método posible para implementar interrupciones por software en la familia de microcontroladores PIC16C5X. Este método aprovecha las ventajas de la arquitectura de los PIC16C5X, la cual permite cambiar el contador de programa mediante software. Pueden implementarse más de ocho líneas de interrupción, pero en la práctica el límite es de seis líneas de interrupción, o 64 condiciones de interrupción. El tiempo de detección de la interrupción se controla por software y las líneas de entrada/salida del los PIC se utilizan como líneas de interrupción.

4.1.- Muestreo por software de las líneas de E/S.

La condición de interrupción se determina detectando el cambio de las líneas IIO que han sido seleccionadas como líneas de interrupción. Estos cambios se usan para crear una tabla de saltos, que permite realizar las diferentes respuestas a cada condición de interrupción. El tiempo de respuesta de la interrupción está bajo el control del software y puede ser tan corto como 10 o 20 milisegundos, dependiendo del programa principal y de la longitud de la subrutina de interrupción.

4.2.- Creación de la Tabla de saltos a rutinas de interrupción.

Cada condición de E/S debe tener su única subrutina para responder a los cambios producidos en las líneas de interrupción. Directamente se puede acceder a estas rutinas usando la habilidad de los PIC16C5X de cambiar el contador de programa bajo el control del software. A continuación puede verse un ejemplo de cómo se encuestan dos líneas:

```
MOVF        CONDTN,W    ;Carga condición de E/S en W.
ANDLW       3           ;Enmascara 6 bits de mayor peso.
ADDWF       2,1         ;Suma el número al PC para saltar
                        ;si se ha producido interrupción.
GOTO        MAIN        ;Si no hay cambio va a MAIN.
GOTO        INT1        ;Si hay cambio en bit 0 va a INTl.
GOTO        INT2        ;Si hay cambio en bit 1 va a INT2.
GOTO        INT3        ;Si bit 0 y en bit 1 va a INT3.
```

El cambio en las líneas de E/S ha sido usado para crear un número de dos bits que es sumado al contador de programa. El GOTO que se ejecutará depende de la dirección que hay ahora en el contador de programa.

4.3.- Creación de una constante de tiempo de muestreo.

En un gran número de aplicaciones que requieren interrupciones, es muy importante realizar el muestreo de las líneas de interrupción a intervalos fijos de tiempo, normalmente a unos pocos microsegundos de tiempo. Existen dos técnicas que pueden ser empleadas para realizar esto con la familia PIC16C5X y son:

- Dividiendo el programa principal en varias secciones.
- Implementando un contador de intervalo de tiempo.

Ambas utilizan el mismo concepto de tabla de salto que se ha descrito en el apartado anterior.
En la primera, el programa principal se divide en varias secciones basadas en el tiempo de encuesta de las líneas de E/S. Cuando se llama a MAIN se suma un registro de salto al contador

de programa. Éste determina qué sección de MAIN debe ser ejecutada a continuación. Al final de la ejecución de la sección, el registro de salto debe ser decrementado para que la próxima sección de código pueda ser ejecutada después de la próxima muestra. Si el registro de salto está a cero, entonces el número de secciones de código se suma al registro de salto para comenzar el programa de nuevo.

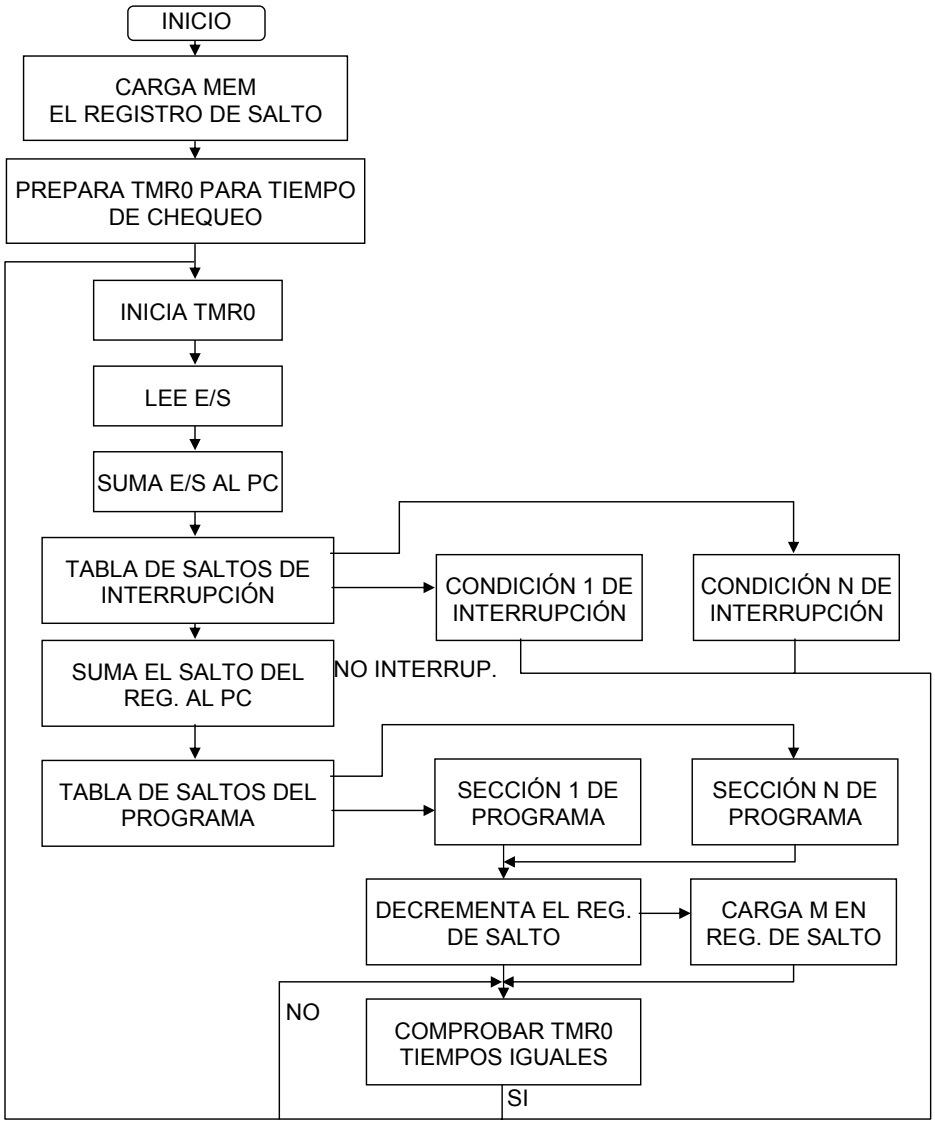
La segunda puede ser implementada usando el contador TMR0. En el comienzo de la encuesta de E/S, el registro TMR0 es inicializado. Entonces comienza la cuenta de ciclos de instrucción. Después de que la subsección del programa principal ha sido ejecutada, el registro TMR0 se resta del tiempo de encuesta deseado. Éste determina cuántas instrucciones necesitan ser ejecutadas antes de la próxima encuesta. Más abajo puede verse un ejemplo. Este ejemplo asume que se necesitan, además, de 0 a 15 ciclos de instrucción. El número necesario debe calcularse para cada aplicación.

```
MOVLW POLL        ;POLL = ciclos hasta la encuesta - 15
SUBWF TMR0,W      ;Determina tiempo a esperar.
ADDWF 2,1         ;Suma el tiempo de espera al PC
NOP               ;15 ciclos de espera adicional.
NOP               ;14 ciclos de espera adicional.
NOP               ;13 ciclos de espera adicional.
NOP               ;12 ciclos de espera adicional.
NOP               ;11 ciclos de espera adicional.
NOP               ;10 ciclos de espera adicional.
NOP               ;9 ciclos de espera adicional.
NOP               ;8 ciclos de espera adicional.
NOP               ;7 ciclos de espera adicional.
NOP               ;6 ciclos de espera adicional.
NOP               ;5 ciclos de espera adicional.
NOP               ;4 ciclos de espera adicional.
NOP               ;3 ciclos de espera adicional.
NOP               ;2 ciclos de espera adicional.
NOP               ;1 ciclo de espera adicional.
GOTO  START       ;0 ciclos de espera adicional.
```

Por ejemplo, si se desea un tiempo de instrucción de 50 ciclos y la subsección que estamos ejecutando consume un total de 40 ciclos de instrucción (incluyendo todos los ciclos de cabecera) el valor de TMR0(40) - POLL(50-15(35)) = 5 se sumará al contador de programa. El programa saltará al sexto NOP Los 9 NOP siguientes serán ejecutados con un total de 10 ciclos de instrucción más. EL GOTO final tiene 2 ciclos de instrucción, y éstos deben ser incluidos en la cabecera del programa.

4.4.- Ejemplo de la técnica del contador de intervalo de tiempo.

Este ejemplo es el corazón del programa de la técnica de interrupción de software con contador de intervalo de tiempo. Este programa asume cuatro condiciones de interrupción, cuatro secciones del programa principal y un intervalo de tiempo adicional de ocho instrucciones. El diagrama de flujo es el siguiente:




```
;*****************************************************************
;     APLICACIÓN DE INTERRUPCIONES POR SOFTWARE
;*****************************************************************
LIST  P=16C54

;*****************************************************************
;     BRANCH es el registro del programa principal
;*****************************************************************

BRANCH      EQU   8
CNDTN       EQU   9
ES          EQU   0A
TEMP        EQU   0B

SETUP       CLRF        CNDTN
            MOVLW       4
            MOVWF       BRANCH      ;Cuatro seccines del programa ppal
            MOVLW       8
            OPTION                  ;Configura el TMR0 como contador
                                    ;de un ciclo por instrucción
START       CLRF        1           ;Inicia el registro TMR0
            MOVF        6,W
            MOVWF       ES
```

```
            IORWF       CNDTN,W     ;Esta sección de código
            MOVWF       TEMP        ;Calcula salto a la tabla
            MOVF        CNDTN,W     ;Cualquier cambio de cero a uno
            SUBWF       TEMP,1      ;es considerado una interrupción
            MOVF        ES,W        ;La ecuación es
            MOVWF       CNDTN       ;(ES+CNDTN)-CNDTN=INT
            MOVF        TEMP,W      ;Donde ES es la entrada actual y
                                    ;CNDTN es la entrada anterior.
            ANDLW       3           ;Máscara que elimina los 6 bits
                                    ;de mayor peso
            ADDWF       2,1         ;Suma la entrada al PC para saltar
                                    ;a la tabla.
            GOTO        MAIN        ;Si INPUT=0
            GOTO        INT1        ;Si INPUT=1
            GOTO        INT2        ;Si INPUT=2
            GOTO        INT3        ;Si INPUT=3
;
INT1        NOP                     ; Código de la interrupción 1
            GOTO        START
INT2        NOP                     ; Código de la interrupción 2
            GOTO        START
INT3        NOP                     ; Código de la interrupción 3
            GOTO        START

MAIN        MOVF        BRANCH,W
            ADDWF       2,1         ;Suma salto a PC y salta a la tabla
            NOP
            GOTO        MAIN4       ;Tabla de saltos, última sección
            GOTO        MAIN3
            GOTO        MAIN2
            GOTO        MAIN1

MAIN1       NOP                     ;Programa ppal sección 1
            GOTO        BRNCHK
MAIN2       NOP                     ;Programa ppal sección 2
            GOTO        BRNCHK
MAIN3       NOP                     ;Programa ppal sección 3
            GOTO        BRNCHK
MAIN4       NOP                     ;Programa ppal sección 4
            GOTO        BRNCHK

BRNCHK      DECFSZ      BRANCH,1    ;Decrementa el registro de
                                    ;salto y comprueba si es 0
            GOTO        TIMCHK
            MOVLW       4
            MOVWF       BRANCH      ;Carga de nuevo BRANCH con 4
                                    ;al final de MAIN
TIMCHK      MOVLW       D'41'       ;Comprueba si TMR0 ha alcanzado
                                    ; 50(50-7)
            SUBWF       1,W         ;Determina el tiempo de espera
            ADDWF       2,1         ;Suma el tiempo de espera al PC
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            GOTO        START
END
```

5.- Almacén y restauración del status en una interrupción.

Esta aplicación se basa en la posibilidad de crear una pila por software en la que poder almacenar tanto la dirección de retorno como el contenido del status o cualquier otro registro que se considere necesario. Puesto que el tamaño de memoria RAM de que dispone el microcontrolador utilizado tiene límite, hay que ser prudente y determinar el mínimo número de niveles que se necesitan y definir la longitud adecuada de la pila.

5.1.- Implementación.

En esta aplicación se implementa una pila con 5 niveles donde se almacena el STATUS o cualquier otro registro cuando se entra en una subrutina de interrupción o de otro de tipo, con la macro PUSH, y se repone el valor del STATUS cuando se va a salir de ella por medio de la macro POP, Por lo que sólo se pueden almacenar 5 registros sin rebasar la pila. La macro PUSH debe llamarse en el inicio de la subrutina con el nombre del registro que se desea almacenar, y la macro POP debe llamarse antes de abandonar la subrutina con el nombre del registro que se almacenó. En una subrutina pueden almacenarse más de un registro, en el momento de reponer estos registros se realizará en el orden inverso al de llamada.

Como la pila diseñada utiliza el registro FSR y el direccionamiento indirecto, el usuario debe restaurar el valor "original" del FSR si se utiliza en cualquier otro lugar del programa. Las rutinas descritas en esta aplicación trabajan sólo si las rutinas llamadas se encuentran entre las primeras 256 palabras de la primera página. Si se desea saltar a otras páginas como en el caso del PIC16C57, debe salvarse el byte de STATUS al mismo tiempo que el PC.

```
;*****************************************
; PILA IMPLEMENTADA POR SOFTWARE
;*****************************************
;
LIST P=16C54, F=INHX8M
;
;*****************************************
; status.asm
; Rutina de demostración de cómo implementar
; una pila capaz de albergar los registros
; que se deseen.
; Nota: Esto es una demostración, por lo que
; se ha colocado la instrucción NOP donde
; debería aparecer el cuerpo de la subrutina
;
;*****************************************

;
W           EQU   0           ;Almacena en W
INDI        EQU   0           ;Direc. Indirecto.
PC          EQU   2           ;Contador de programa.
FSR         EQU   4           ;Puntero de la pila.
STACK       EQU   8           ;Principio de la pila.
STATUS      EQU   3           ;Registro de estado.
DATO        EQU   0x0F        ;Registro de datos.

;*************************************************
;     Nota: Las próximas 5 posiciones de RAM
;     deben ser reservadas para la PILA. No se
;     debe usar ninguna posición entre la 8
;     y la 12 (en decimal).
```

```
;*****************************************************

            ORG   01FF
            GOTO  INIT        ;Vector de inicio del prog.

            ORG   0
INIT        MOVLW STACK       ;Carga “stack” en el
            MOVWF FSR         ;puntero indirecto.
            GOTO  START       ;Comienzo del programa.

;*****************************************************
;
;     Define PUSH como una macro que se usa para
;     almacenar un registro en la pila
;*****************************************************

PUSH        MACRO LABEL
            MOVF  LABEL,W     ;Almacena LABEL en el stack
            MOVWF INDI        ;Dir indirecto
            INCF  FSR         ;Incrementa el puntero de pila
ENDM

;*****************************************************
;
;     Define POP como una macro que restaura un registro
;     tomado de la pila
;*****************************************************

POP         MACRO LABEL
            DECF  FSR         ;Apunta a la última posición
                              ;ocupada de la pila.
            MOVF  INDI,W      ;Obtiene el último registro
                              ;de la pila.
            MOVVF LABEL       ;Repone el valor del registro
ENDM
;*****************************************************
;
;     PROGRAMA PRINCIPAL
;*****************************************************
START
            NOP               ;código encargado de gestionar
            NOP               ;las interrupciones por software
            NOP               ;y código del programa principal
            NOP
            GOTO  START
;*****************************************************
;
;     RUTINA DE INTERRUPCIÓN
;*****************************************************
Rutina_interrup
            PUSH  STATUS      ;Guarda en la pila el status
            MOVF  STATUS,W    ;Almacena STATUS en la pila
            MOVWF INDI        ;Dir indirecto
            INCF  FSR         ;Incrementa el puntero de la pila.
            PUSH  DATO        ;Guarda en el registro DATA
            MOVF  DAT0,W      ;Almacena DATO en la pila
            MOVWF INDI        ;Dir indirecto
            INCF  FSR         ;Incrementa el puntero de pila
            NOP
            NOP
            NOP
            POP   DATO        ;Restablece el registro DATA
            DECF  FSR         ;Apunta a la última posición
            MOVF  INDI,W      ;Obtiene el último registro
            MOVWF DATO        ;Repone el valor de1 registro
```

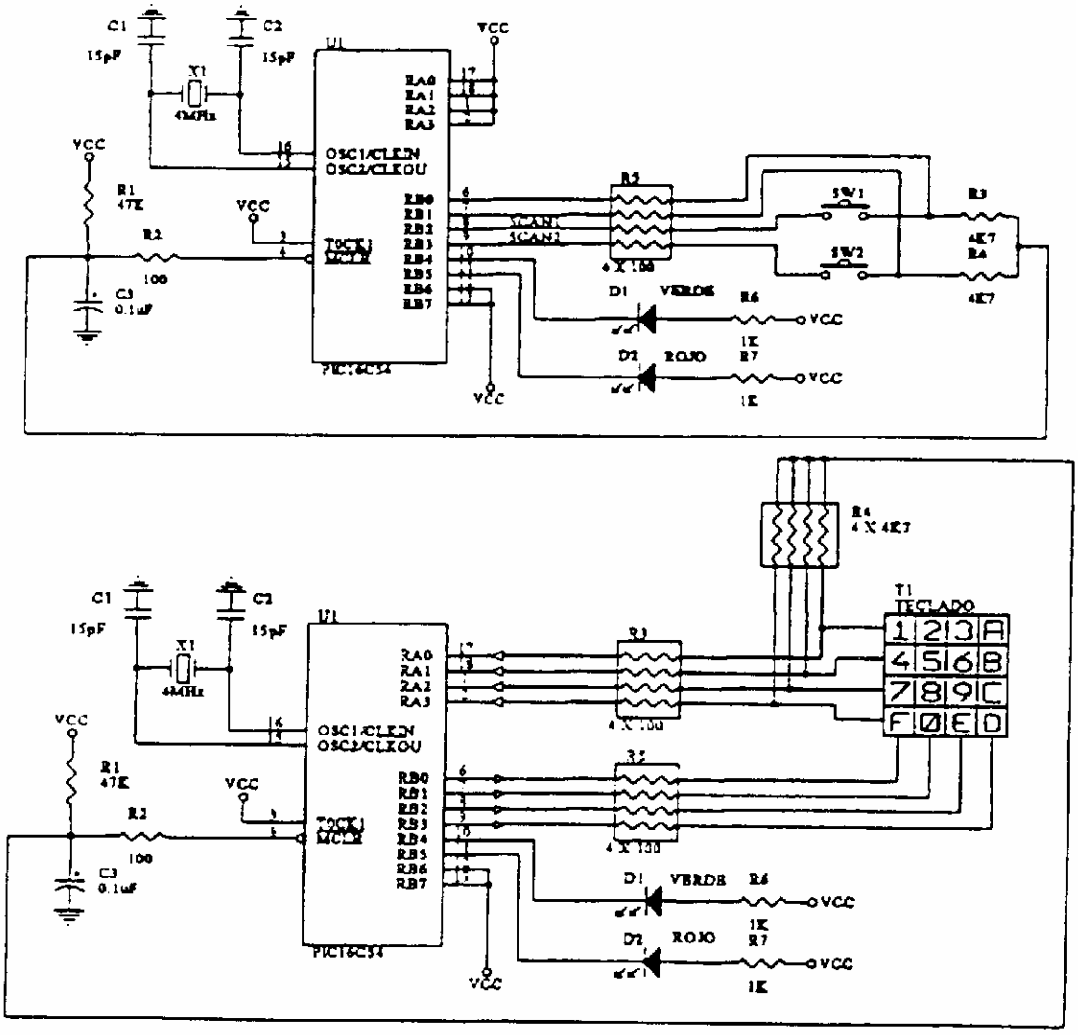
```
            POP   STATUS      ;Restablece el status
            DECF  FSR         ;Apunta a la última posición
            MOVF  INDI,W      ;Obtiene el último registro
            MOVWF STATUS      ;Repone el valor del registro
            GOTO  START       ;Continua con el programa.
END
```

6.- Despertar cuando se pulsa una tecla.

En un gran número de aplicaciones el PIC16C5X sólo debe actuar cuando se pulsa una tecla, por ejemplo en mandos a distancia. En tales aplicaciones, la vida de la batería puede alargase poniendo el PICIGCSX en el modo sleep durante el estado inactivo, y cuando se pulsa la tecla, despierta, realiza la tarea y vuelve a dormirse (regresa al modo SLEEP).

6.1.- Implementación.

El circuito de la figura muestra una aplicación con 2 teclas. El PIC16C5X normalmente se encuentra dormido (modo SLEEP) consumiendo una corriente muy pequeña. Si se pulsa alguna de las teclas, el PIC16C5X despierta, muestrea la tecla pulsada y enciende el led correspondiente. Cuando se pulsa SW1 se enciende el led verde, y cuando se pulsa SW2 se enciende el led rojo; cuando deja de pulsarse la tecla los leds son apagados y el microcontrolador regresa al modo SLEEP. Los leds se usan como demostración. En la vida real la pulsación de la tecla generaría una transmisión completa por el led antes de regresar al modo SLEEP Este ejemplo puede ser extendido a más de dos teclas como indica la figura adicional.



En el modo sleep, las salidas de escaneo (SCANl y SCAN2) son puestas a nivel bajo. En este estado, el condensador C1 está totalmente cargado y hay un nivel alto en el pin MCLR del PIC16CSX. Cuando se pulsa una tecla C1 se descarga a través de R3 o R4 (dependiendo de que se pulse SW1 o SW2) y la tensión en C1 baja rápidamente (aproximadamente 1ms), causando un nivel bajo en el pin MCLR, el cual produce un reset y que se despierte el microcontrolador. En el estado de reset, las salidas SCAN1 y SCAN2 por defecto en alta impedancia, hace que la vía de descarga del condensador C1 se bloquee y se cargue a nivel alto a través de la resistencia R1.

Los valores de RC se han elegido para que los tiempos de Los ciclos de carga y descarga sean menores que el tiempo de reset, (aproximadamente 18 ms), y ciertamente mucho menor que la mínima duración una pulsación de la tecla (aproximadamente 50-100 ms).

Después de que el ciclo de reset se complete, la ejecución del código hace que tomen por un momento nivel bajo las salidas SCAN1 y SCAN2 como orden de muestrear la pulsación de una tecla. Esto no causa la descarga del condensador debido a que la duración del nivel bajo es del orden de 10 μsegundos.

Una vez que la función keystroke (pulsación de tecla) ha sido ejecutada, el programa espera hasta que la tecla deje de pulsarse, poniendo las salidas SCAN1 y SCAN2 en nivel bajo y regresando al estado sleep (dormido). Las resistencias de 100 ohmios no son necesarias para el funcionamiento, pero son recomendables porque proporcionan protección frente a descargas electrostáticas (ESD). Cuando se presionan los pulsadores SW1 y SW2, Suele pasar frecuentemente ESD al PICI6C5X.

```
;*************************************************************
;     DESPERTAR CUANDO SE PULSA UNA TECLA
;*************************************************************

TITLE “Despertar cuando se pulsa una tecla”
LIST  P=16C54,    F=INHX8M
;*************************************************************
;     Este programa demuestra como despierta el PIC16C5X
;     del estado sep, cuando se pulsa una tecla.
;     El programa consta de dos teclas, pero puede ser
;     modificado a más teclas.
;     Cuando se pulsa SW1 se enciende el led VERDE.
;     Cuando se pulsa SW2 se enciende el led ROJO.
;*************************************************************

; Tabla de definiciones

PC          EQU   2
PORT_B      EQU   6
SCAN1       EQU   2
SCAN2       EQU   3
SW1         EQU   0
SW2         EQU   1
GRN_LED     EQU   4
RED_LED     EQU   5
MSEC_20     EQU   D’40’
DB1         EQU   8
GP          EQU   8
DB2         EQU   9
;
; Asignación del puerto B.
;
```

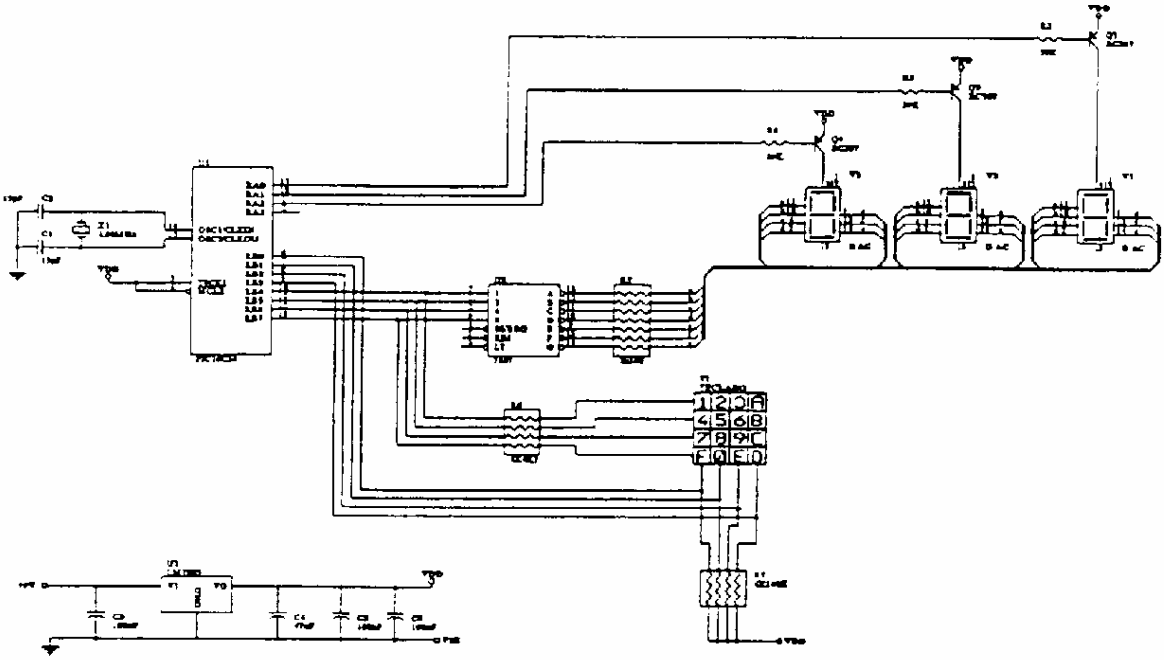
```
; 0 -> SW1        ENTRADA
; 1 -> SW2        ENTRADA
; 2 -> SCANl      SALIDA
; 3 -> SCAN2      SALIDA
; 4 -> GRN_LED    SALIDA
; 5 -> RED_LED    SALIDA
;6 y 7 -> ASIGNADOS COMO FALSAS SALIDAS.
;*************************************************************
;
; Comienzo del programa.
;*************************************************************

            ORG   0
START
            CALL  INIT_PORT_B       ;Inicializa el PORT_B
            CALL  DELAY             ;Retardo de 20 ms.
            CALL  SCAN_KEYS         ;Toma el valor de la tecla.
            MOVWF GP                ;La almacena en RAM.
            BTFSC GP,SW1            ;Salta si no se ha pulsado SW1.
            CALL  TURN_GREEN_ON     ;En otro caso hace la rutina.
            BTFSC GP,SW2            ;Salta si no se ha pulsado SW2.
            CALL  TURN_RED_ON       ;En otro caso hace la rutina.

CHK-FOR-KEY                         ;Espera a que deje de pulsarse.
            CALL  DELAY             ;Retardo de 20 ms.
            CALL  SCAN-KEYS         ;Toma el valor de las teclas.
            XORLW 0                 ;Or exclusiva con 0.
            BNZ   CHK-FOR-KEY       ;Si está pulsada una tecla regresa

NO-KEY-PRESSED
            BCF   PORT_B,SCAN1      ;Pone las líneas de escaneo
            BCF   PORT_B,SCAN2      ;a nivel bajo.
            CALL  TURN_GREEN_OFF    ;Apaga el led verde.
            CALL  TURN_RED_OFF      ;Apaga el led rojo.
            SLEEP                   ;Deja dormido el microcontrolador.

INIT_PORT_B
            MOVLW B'00000011'       ;configura RBO y RB1 como entradas
            TRIS  PORT_B            ;y RB2-7 como salidas.
            MOVLW 0FFh
            MOVWF PORT_B            ;Valor de defecto para el puerto B.
            RETLW 0                 ;Regreso de la subrutina.
;*************************************************************
;     Esta rutina escanea dos teclas y devuelve lo siguiente:
;     0           Si no se ha pulsado ninguna tecla
;     1           Si se ha pulsado SW1.
;     2           Si se ha pulsado SW2.
;     3           Si se han pulsado SW1 y SW2.
;*************************************************************
SCAN-KEYS
            BCF   PORT_B,SCAN1      ;Habilita SW1.
            BCF   PORT_B,SCAN2      ;Habilita SW2.
            MOVLW B'00000011'       ;Carga la máscara en W.
            ANDWF PORT_B,0          ;AND con el puerto.
            BSF   PORT_B,SCAN1      ;Deshabilita el escaneo.
            BSF   PORTB,SCAN2       ; /
            ADDWF PC,1              ;Calcula el valor a retornar.
            RETLW 3                 ;Pulsados SW1 y SW2.
            RETLW 2                 ;pulsado SW2.
            RETLW 1                 ;Pulsado SW1.
            RETLW 0                 ;No se ha pulsado ninguna teda.
```

```
;*************************************************************
Retardo de tiempo de ADROX 20.4ms, para un reloj de 4Mhz.
;*************************************************************
DELAY
            MOVLW       MSEC_20
            MOVWF       DB1
DLYl
            CLRF        DB2
            DECFSZ      DB1
            GOTO        DLY2
            RETLW       0
DLY2
            DECFSZ      DB2         ;Bucle de 0.512 ms.
            GOTO        DLY2
            GOTO        DLY1

TURN_GREEN_ON
            BCF         PORT_B,GRN_LED
            RETLW       0

TURN_RED_ON
            BCF         PORT_B,RED_LED
            RETLW       0

TURN_GREEN_OFF
            BSF         PORT_B,GRN_LED
            RETLW       0

TURN_RED_OFF
            BSF         PORT_B,RED_LED
            RETLW       0

            ORG         1FF
            GOTO        START

END
```

7.- Teclado matricial y displays multiplexados.

La atención a un teclado y a una serie de displays es una de las aplicaciones más comunes de los microcontroladores. En este capítulo se explica cómo se puede conseguir atender a un teclado matricial de 4x4 teclas y al mismo tiempo visualizar información sobre unos displays de forma multiplexada.

La aplicación propuesta consta de un teclado de 16 teclas distribuidas matricialmente, y de 3 displays en los cuales se mostrarán los 3 últimos números que se hayan pulsado en el teclado, mientras que si la última tecla pulsada es una letra se ignora.



7.1.- Muestreo del teclado 4x4.

El teclado esta conectado al puerto B del PICI6C54. Las cuatro columnas están conectadas a RB0-RB3 y las cuatro filas a RB4-RB7. El muestreo se realiza como sigue:

1. Las columnas están conectadas como entradas y las filas como salidas.
2. Se pone un nivel bajo en cada fila y se comprueba el valor de las columnas. Puesto que las columnas están conectadas a un nivel alto a través de las resistencias de pull-up (R7), cuando leemos las cuatro líneas, éstas se deben encontrar a nivel alto. Sin embargo, si pulsamos una tecla cuando en la fila correspondiente hay un nivel bajo, este nivel se traslada a través de la columna correspondiente hasta la línea de entrada indicando la pulsación de la tecla.
3. Conocida la columna pulsada, se pasa a averiguar qué fila ha producido la pulsación de la tecla. Se ponen una a una las filas a nivel bajo y se comprueba si la fila detectada pasa a nivel bajo. En el momento en que pasa a nivel bajo, se obtiene la fila que produce el nivel bajo en la columna. Conocida la fila y columna se convierte en su correspondiente código binario.
4. Una vez obtenida la tecla se realiza un ciclo de espera, hasta que deja de pulsarse la tecla. Este ciclo puede no ser necesario o puede sustituirse por una temporización de 50 ms, pero si no se realiza la espera o la temporización, con una sola pulsación se produce

un gran número de teclas pulsadas. Los paquetes de resistencias R7 y R6 se han elegido con una relación 1.20. Esto asegura una tensión inferior a 2.5v en las entradas RB0 - RB3 cuando se pulsa una tecla.

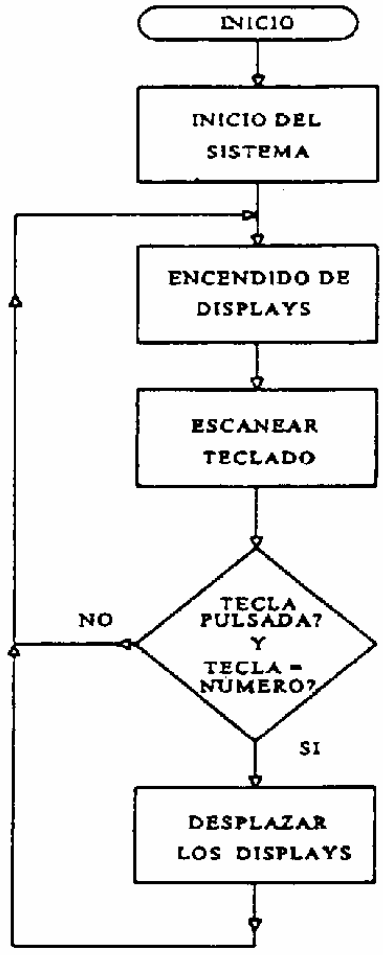
7.2.- Encendido de los displays.

Se utilizan displays de ánodo común. Todos los displays tienen unidos los segmentos homónimos y a su se utiliza un circuito excitador para encenderlos (7447), que además sirve de buffer. El dato que muestra este circuito es proporcionado por el PIC16C54 a través de las patillas RB4 - RB7 que comparte con el teclado. La elección del display se realiza con las líneas RA0 - RA3. La línea RA4 queda libre para poder darle una posible a aplicación.

El encendido se realiza de la forma siguiente:

1. Se pone el dato que se quiere mostrar en los pines RB4 - RB7, y se selecciona el display colocando un nivel bajo en la línea R4x correspondiente.
2. Espera 5 ms.
3. Se deselecciona el display poniendo a nivel alto la línea RAx correspondiente.
4. Se repiten los pasos 1, 2 y 3 con el resto de los displays.

La secuencia que sigue el programa se puede ver a continuación:



```
;*****************************************************************
;     MUESTRA COMO SE MULTIPLEXAN DISPLAYS Y TECLADO
;*****************************************************************

;     Nombre del fichero: TECLADIS.ASM
;     Programa encargado de visualizar en tres
;     displays los valores tecleados en un teclado de
;     4 filas y 4 columnas.
;     La línea PA3 queda libre para otra aplicación.
;     El microcontrolador utilizado es el PIC16C54
;     con un reloj de 4 MHz

;     Asignación de las líneas de los puertos A y B
;           PORTA
;     0 -> SEL0   salida selección de DIS0
;     1 -> SEL1   salida selección de DIS1
;     2 -> SEL2   salida selección de DIS2

;           PORTB
;     0 -> COLU1  entrada columna 1 de teclado
;     1 -> COLU2  entrada columna 2 de teclado
;     2 -> COLU3  entrada columna 3 de teclado
;     3 -> COLU4  entrada columna 4 de teclado
;     4 -> FILA1  entrada fila 1 o dato display
;     5 -> FILA2  entrada fila 2 o dato display
;     6 -> FILA3  entrada fila 3 o dato display
;     7 -> FILA4  entrada fila 4 o dato display

;*****************************************************************
;     Variables del sistema
;*****************************************************************
PC          EQU   0x02  ;Contador de programa
STATUS      EQU   0x03  ;Registro de estado
PORTA       EQU   0x05  ;Puerto A
PORTB       EQU   0x06  ;Puerto B
C           EQU   0     ;Bit de Acarreo
Z           EQU   2     ;Bit de Cero
;*****************************************************************
;     Definición de variables
;*****************************************************************
; Variables de display y cuenta

DIS0        EQU   0x10  ;Contiene el valor del
                        ;Display 0 y a descontar
DIS1        EQU   0x11  ;Contiene el valor del
                        ;Display 1 y a descontar
DIS2        EQU   0x12  ;Contiene el valor del
                        ;Display 2 y a descontar
AUX         EQU   0x13  ;Variables aux de tempo
                        ;y otras aplicaciones
SEL0        EQU   0     ;Salida selección de DIS0
SEL1        EQU   1     ;Salida selección de DIS1
SEL2        EQU   2     ;Salida selección de DIS2

TECLA       EQU   0x17  ;Contiene la tecla pulsada
PULSADA     EQU   0x18  ;Indica si ha sido pulsada una tecla

COLU1       EQU   0     ;Entrada Columna 1 de teclado
COLU2       EQU   1     ;Entrada Columna 2 de teclado
COLU3       EQU   2     ;Entrada Columna 3 de teclado
COLU4       EQU   3     ;Entrada Columna 4 de teclado
```

```
FILA1       EQU   4     ;Salida fila 1 o dato display
FILA2       EQU   5     ;Salida fila 2 o dato display
FILA3       EQU   6     ;Salida fila 3 o dato display
FILA4       EQU   7     ;Salida fila 4 o dato display

N           EQU   D'250' ;Valor del bucle de tempo
T_UNOS      EQU   0xFF   ;Valor de puesta a unos

; Las subrutinas se colocan en la parte baja
; del mapa de memoria de código

ORG         00H

; SUBRUTINAS DE DISPLAY Y TEMPORIZACIÓN

;     Subrutina de temporización de 2.5 ms
;     T=(N-1)*10us+6us
TEMP_25MS   MOVLW       N     ;Inicia cuenta de valor n
            MOVWF       AUX   ;en la variable AUX
LOOP        DECFSZ      AUX,F ;Decrementa AUX, AUX=0?
            GOTO        CONT  ;Aux no es 0
            RETLW       0     ;Aux=0 y fin
CONT
            NOP
            NOP
            NOP
            NOP
            NOP
            GOTO        LOOP
;     Subrutina de visualización de los displays
; Visualiza de forma multiplexada los displays
; Llama a la subrutina TEMP_25MS y utiliza un nivel de pila
VISUAL
            SWAPF       DIS0,W      ;Dato de DIS0
            MOVWF       PORTB       ;en el puerto
            BCF         PORTA,SEL0  ;Selección
            CALL        TEMP_25MS   ;Espera 2.5ms
            CALL        TEMP_25MS   ;Espera 2.5ms
            BSF         PORTA,SEL0  ;Reselección
            SWAPF       DIS1,W      ;Dato de dis1
            MOVWF       PORTB       ;en el puerto
            BCF         PORTA,SEL1  ;Selección
            CALL        TEMP_25MS   ;Espera 2.5ms
            CALL        TEMP_25MS   ;Espera 2.5ms
            BSF         PORTA,SEL1  ;Reselección
            SWAPF       DIS2,W      ;Dato de dis2
            MOVWF       PORTB       ;en el puerto
            BCF         PORTA,SEL2  ;Selección
            CALL        TEMP_25MS   ;Espera 2.5ms
            CALL        TEMP_25MS   ;Espera 2.5ms
            BSF         PORTA,SEL2  ;Reselección
            RETLW       00H
;     Subrutina de atención al teclado
; Retorna el código de la tecla
PULTECLA
            ADDWF       PC,F
            RETLW       0x01        ;Tecla=1
            RETLW       0x04        ;Tecla=4
            RETLW       0x07        ;Tecla=7
            RETLW       0x0F        ;Tecla=F
            RETLW       0x02        ;Tecla=2
```

```
            RETLW       0x05        ;Tecla=5
            RETLW       0x08        ;Tecla=8
            RETLW       0x00        ;Tecla=0
            RETLW       0x03        ;Tecla=3
            RETLW       0x06        ;Tecla=6
            RETLW       0x09        ;Tecla=9
            RETLW       0x0E        ;Tecla=E
            RETLW       0x0A        ;Tecla=A
            RETLW       0x0B        ;Tecla=B
            RETLW       0x0C        ;Tecla=C
            RETLW       0x0D        ;Tecla=D

;     Subrutina de atención al teclado
; Retorna la tecla pulsada e indica pulsada
; Llama a la subrutina PULTECLA y utiliza un nivel de pila
TECLADO
            MOVLW       0x0F
            MOVWF       PORTB
            CLRF        PULSADA     ;Inicializa no pulsada
            BTFSS       PORTB,COLU1
            GOTO        COL1        ;Columna 1
            BTFSS       PORTB,COLU2
            GOTO        COL2        ;Columna 2
            BTFSS       PORTB,COLU3
            GOTO        COL3        ;Columna 3
            BTFSS       PORTB,COLU4
            GOTO        COL4        ;Columna 4
            RETLW       00H         ;No se ha pulsado tecla
COL1                                ;Escanea la columna 1
            CLRF        AUX         ;00 -> AUX
            GOTO        CFILA
COL2                                ;Escanea la columna 2
            MOVLW       0x04        ;04 -> W
            MOVWF       AUX         ;W -> AUX
            GOTO        CFILA
COL3                                ;Escanea la columna 3
            MOVLW       0x08        ;08 -> W
            MOVWF       AUX         ;W -> AUX
            GOTO        CFILA
COL4                                ;Escanea la columna 3
            MOVLW       0x0C        ;0C -> W
            MOVWF       AUX         ;W -> AUX
            GOTO        CFILA
CFILA
            MOVLW       0xEF
            MOVWF       PORTB
            NOP
            MOVF        PORTB,0
            IORLW       0xF0
            XORLW       0xFF
            BTFSS       STATUS,Z    ;¿Fila 1?
            GOTO        TECFIN      ;Si es fila 1
            MOVLW       0xDF        ;No es fila 1
            MOVWF       PORTB
            INCF        AUX,F
            MOVF        PORTB,W
            IORLW       0xF0
            XORLW       0xFF
            BTFSS       STATUS,Z    ;¿Fila 2?
            GOTO        TECFIN      ;Si es fila 2
            MOVLW       0xBF        ;No es fila 2
```

```
            MOVWF       PORTB
            INCF        AUX,F
            MOVF        PORTB,W
            IORLW       0xF0
            XORLW       0xFF
            BTFSS       STATUS,Z    ;¿Fila 3?
            GOTO        TECFIN      ;Si es fila 3
            MOVLW       0x7F        ;No es fila 3
            MOVWF       PORTB
            INCF        AUX,F
            MOVF        PORTB,W
            IORLW       0xF0
            XORLW       0xFF
            BTFSS       STATUS,Z    ;¿Fila 4?
            GOTO        TECFIN      ;Si es fila 4
            RETLW       00H         ;No se ha pulsado tecla
TECFIN
            MOVF        AUX,W       ;AUX->W
            CALL        PULTECLA    ;Obtiene la tecla
            MOVWF       TECLA       ;Guarda el valor en tecla
            MOVLW       T_UNOS
            MOVWF       PULSADA     ;Indica que está pulsada
            MOVLW       0x0F
            MOVWF       PORTB
ESPERA
            MOVLW       0x0F0       ;Bucle de espera que
            ADDWF       PORTB,W     ;deje de pulsarse la
            MOVWF       AUX,W       ;tecla
            BTFSS       STATUS,Z
            GOTO        ESPERA      ;Esta pulsada y espera
            RETLW       00H         ;No está pulsada y sigue
;*****************************************************************
;     Programa principal
;*****************************************************************
;Inicialización del sistema

INICIO
            MOVLW       0x07
            MOVWF       PORTA       ;Carga puerto A=07h
            CLRW
            TRIS        PORTA       ;Inicia el puerto A
                                    ;como salidas
            MOVLW       0x0F
            MOVWF       PORTB       ;Carga puerto B=0FH
            TRIS        PORTB       ;Inicia PORTB <0:3>
                                    ;como entradas y
                                    ;PORTB <4:7> como salidas
            CLRF        DIS0        ;Displays a cero
            CLRF        DIS1
            CLRF        DIS2
            CLRF        TECLA       ;00->TECLA
            CLRF        PULSADA     ;00->PULSADA
            CLRF        AUX         ;00->AUX
; Parte del programa que se ejecuta continuamente
CONTINUA
            CALL        VISUAL      ;Visualiza los displays
            CALL        TECLADO     ;Escanea el teclado
            BTFSS       PULSADA,7   ;¿Tecla pulsada?
            GOTO        CONTINUA    ;No pulsada
            CLRF        PULSADA     ;Si pulsada
            MOVLW       0x0A
```

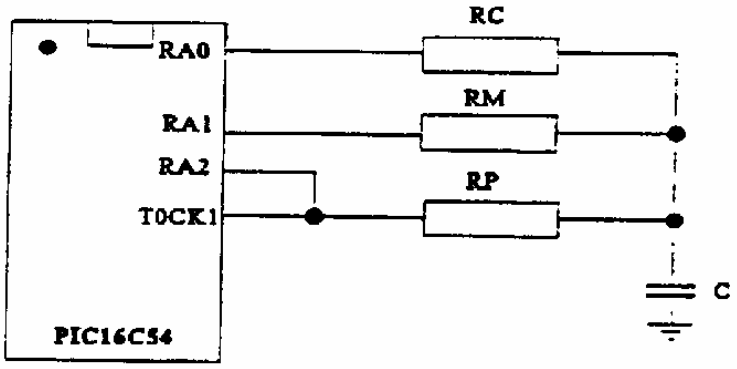
```
            SUBWF       TECLA,W
            MOVWF       AUX
            BTFSC       AUX,7       ;¿Son numeros 0-9?
            GOTO        NUMERO      ;Si
            GOTO        CONTINUA    ;NO, espera entre un número
NUMERO
            MOVF        DIS1,W      ;Pasa DIS1 a DIS2
            MOVWF       DIS2
            MOVF        DIS0,W      ;Pasa DIS0 a DIS1
            MOVWF       DIS1
            MOVF        TECLA,W     ;Pone TECLA en DIS0
            MOVWF       DIS0
            GOTO        CONTINUA
;Inicio de la ejecución del programa
            ORG         1FF
            GOTO        INICIO      ;Comienzo del programa
END
```

8.- Medida de resistencias y capacidades.

Esta aplicación describe un método para implementar medidas de resistencias o condensadores usando el PIC16FXX. En esta aplicación se describe nada más cómo medir resistencias; esto se puede trasladar fácilmente a los condensadores cambiando el condensador patrón por uno desconocido. El ohmetro necesita tan sólo dos componentes externos, y puede configurarse mediante hardware y software para medir resistencias con resoluciones desde 6 bits hasta más de 10 bits, con tiempos de medida de 250 μs (6 bits a 8 MHz) o mayores. Este método usa la técnica de calibración en software, que compensa variaciones de tensiones, tiempos y temperatura, así como los errores de los componentes.

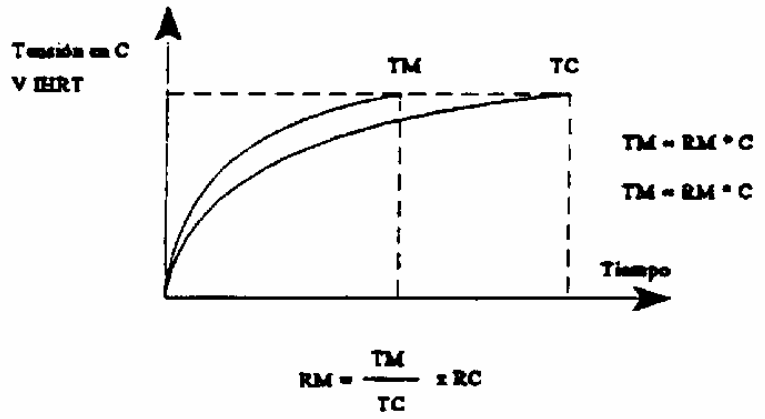
8.1.- Teoría de la operación.

La aplicación utiliza el circuito de carga del condensador (véase la figura) para convertir la resistencia en tiempo, el cual puede ser medido con facilidad utilizando un microcontrolador. RP es una pequeña resistencia de 100 Ω para limitar los picos de tensión en la descarga del condensador.



Primero, una tensión de referencia (normalmente VDD) se aplica a una resistencia calibrada, RC. El condensador C se carga positivamente hasta el valor de tensión en que la entrada bascula. Esto genera el valor de calibración por software que se usa para calibrar la mayor parte de los errores de los circuitos y componentes, incluidos los defectos del condensador, cambios en los valores de entrada de tensión umbral y variaciones de temperatura. Después C es descargado; la tensión de referencia se aplica en la resistencia que va a ser medida. El tiempo que tarda en alcanzar la tensión umbral es medido y comparado con el valor de calibración para

determinar la resistencia actual (RM) (véase la figura). Cuando se utiliza un termistor la obtención de la temperatura se realiza mediante la utilización de una tabla de datos.



8.2.- Configuración del circuito.

Los valores de RC y C son seleccionados basándose en el número de bits de resolución que se necesitan. RC debe tener aproximadamente la mitad del mayor valor de resistencia que se desea medir y se cumple :

$$C = \frac{-T}{RM * ln\left(1 - \frac{Vt}{Vr}\right)}$$

Donde:

Vr = Tensión de referencia.
T = Tiempo necesario para alcanzar la resolución deseada.
Vt = Tensión umbral de una entrada del PIC.
RM = Máximo valor de la resistencia que va a ser medido.

El valor que debe elegirse de C tiene que ser algo menor que el calculado con la expresión anterior para asegurar que el PIC no rebase la cuenta durante la medición. Por ejemplo, si usamos una RM = 200KΩ para una resolución de 8 bits con un reloj de 8 MHz, Vr = 5v, Vt = 3v, RC = l00K y el programa de cuenta utiliza 6 ciclos de instrucción por cada incremento de cuenta, nos queda:

T = 256 cuentas * 118 MHz * 4 cicloslinstrucción * 6 instruccioneslcuenta = 768 ps.
C = 4200 pF (se usará 3900 pF).

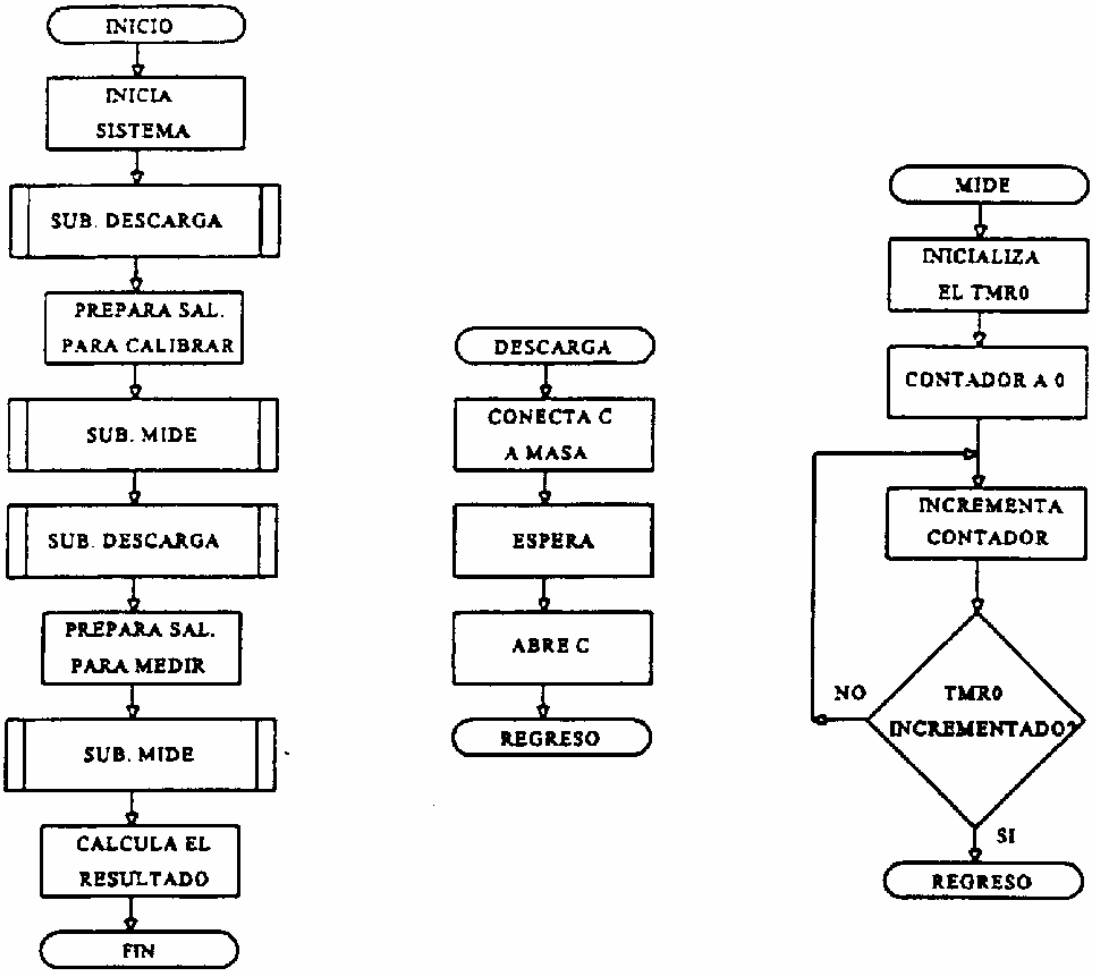
8.3.- Realización del circuito.

El ciclo de calibración elimina todos los errores de primer orden (offset, ganancia, defectos de C, Tensión de alimentación y temperatura) excepto la precisión de R. Debe almacenarse el valor de R con una pequeña variación para reducir los errores en la medición. Otras fuentes de error son las corrientes de fugas de los pines de I/O, la no linealidad de las resistencias y condensadores, la inexactitud de la tensión umbral y del tiempo de medición (con un tiempo de +/- un ciclo de instrucción). La realización de la medición mostrada por el ohmetro puede considerarse con una precisión de +/- 1 % por cada década.

8.4.- Implementación.

El programa que implementa el circuito de la figura se puede ver a continuación. Este código implementa mediciones de hasta 16 bits (65535 ciclos de medida) y calcula el resultado utilizando subrutinas de multiplicación y división de 16 bits. Esto conlleva emplear mucho tiempo en la obtención del valor de la resistencia.

Es mucho más eficiente, si la aplicación lo permite, utilizar mediciones de 8 bits. Con ello las subrutinas matemáticas se reducen sustancialmente, así como el tiempo de medición por simplificar el código y reducir la cuenta. Los organigramas que describen el proceso pueden verse a continuación:




```
;*****************************************************************
;     LECTURA DEL VALOR DE UNA RESISTENCIA
;*****************************************************************
      LIST  P=16c54, f=inhx8m
;     Nombre del fichero:     RESIS.ASM
; Calcula el valor de una resistencia con 16 bits
; Utiliza un reloj de 8 MHz
; Variables del sistema

INDI        EQU         0x00        ;Dir. Indirecto
TMR0        EQU         0x01        ;Reloj de tiempo real
PC          EQU         0x02        ;Contador de programa
```

```
STATUS      EQU         0x03        ;Registro de estado
FSR         EQU         0x04        ;Registro puntero
PORTA       EQU         0x05        ;Puerto A
PORTB       EQU         0x06        ;Puerto B
C           EQU         0           ;Bit de Carry
Z           EQU         2           ;Bit de Cero
F           EQU         1
;Definición de variables
ACCA        EQU         8           ;Parte alta de ACCA
                                    ;ACCA+1=parte baja de ACCA
ACCB        EQU         0A          ;Parte alta de ACCB
                                    ;ACCB+1=parte baja de ACCB
ACCC        EQU         0C          ;Parte alta de ACCC
                                    ;ACCC+1=parte baja de ACCC
ACCD        EQU         0E          ;Parte alta de ACCD
                                    ;ACCD+1=parte baja de ACCD
ACCE        EQU         10          ;Parte alta de ACCE
                                    ;ACCE+1=parte baja de ACCE
TCAL        EQU         12          ;Parte alta de TCAL
                                    ;TCAL+1=parte baja de TCAL
                                    ;TCAL contiene el tiempo calculado
TEMP        EQU         14

; Valor en la resistencia de calibración (12092 OHMS)

RCALMS      EQU         2F          ;Byte de mayor peso en HEX
RCALLS      EQU         3C          ;Byte de menor peso en HEX

; ********SUBRUTINAS*******
;*****************************************************************
; Subrutina que suma dos numeros de 16 bits
; el resultado es de 16 bits
; ACCA + ACCB => ACCB
;*****************************************************************
MADD        MOVF        ACCA+1,W
            ADDWF       ACCB+1,F    ;Suma el byte bajo
            BTFSC       STATUS,C    ;Suma con acarreo
            INCF        ACCB,F
            MOVF        ACCA,W
            ADDWF       ACCB,F      ;Suma el byte alto
            RETLW       00H
;*****************************************************************
; Subrutina que realiza una multiplicación de 16 bits
; el resultado es de 32 bits
; ACCA * ACCB => ACCB (16 MSB) Y ACCC (16 LSB)
; Utiliza las variables ACCA,ACCB,ACCD,ACCE,TEMP.
;*****************************************************************
MPY         CALL        SETUP       ;El resultado en ACCB (16 MSB)
            MOVLW       10
            MOVWF       TEMP
MLOOP       RRF         ACCD,F      ;Rotación de ACCD a la derecha
            RRF         ACCD+1,F
            BTFSC       STATUS,C    ;Es necesario sumar?
            CALL        MADD
            RRF         ACCB,F
            RRF         ACCB+1,F
            RRF         ACCC,F
            RRF         ACCC+1,F
            DECFSZ      TEMP,F      ;Otro bucle hasta que se
                                    ;comprueben todos los bits
            GOTO        MLOOP
```

```
            RETLW       00H

SETUP       MOVF        ACCB,W      ;Mueve ACCB a ACCD
            MOVWF       ACCD
            MOVF        ACCB+1,W
            MOVWF       ACCD+1
            MOVF        ACCC,W      ;Mueve ACCC a ACCE
            MOVWF       ACCE
            MOVF        ACCC+1,W
            MOVWF       ACCE+1
            CLRF        ACCB
            CLRF        ACCB+1
            RETLW       00H
;*****************************************************************
; Subrutina que realiza una división de 16 bits
; el resultado es de 16 bits
; ACCB / ACCA => ACCB
; Utiliza las variables ACCA,ACCB,ACCD,ACCE,TEMP
;*****************************************************************
DIV         CALL        SETUP
            MOVLW       20
            MOVWF       TEMP
            CLRF        ACCC
            CLRF        ACCC+1
DLOOP       CLRC
            RLF         ACCE+1,F
            RLF         ACCE,F
            RLF         ACCD+1,F
            RLF         ACCD,F
            RLF         ACCC+1,F
            RLF         ACCC,F
            MOVF        ACCA,W
            SUBWF       ACCC,W      ;Comprueba si A>C
            BTFSC       STATUS,C
            GOTO        NOCHK
            MOVF        ACCA+1,W
            SUBWF       ACCC+1,W    ;Si MSB son iguales
                                    ;entonces comprueba LSB
NOCHK       BTFSC       STATUS,C    ;Carry=1 si C > A
            GOTO        NOGO
            MOVF        ACCA+1,W    ;C-A en C
            SUBWF       ACCA+1,F
            BTFSS       STATUS,C
            DECF        ACCC,F
            MOVF        ACCA,W
            SUBWF       ACCC,F
            SETC                    ;Desplaza A 1 en B(RESULTADO)
NOGO        RLF         ACCB+1,F
            RLF         ACCB,F
            DECFSZ      TEMP,F      ;Bucle hasta que son
            GOTO        DLOOP       ;comprobados todos los bits
            RETLW       00H
;*****************************************************************
; Subrutina que realiza la descarga del condensador
;*****************************************************************
DSCHRG      MOVLW       B'00001011' ;Inicializa el TMR0
            TRIS        PORTA
            MOVLW       0FFH
            MOVWF       TEMP
LOOP        DECFSZ      TEMP,F      ;Espera 255x3 ciclos
            GOTO        LOOP
```

```
            MOVLW       B'00001111' ;Desactiva RA2
            TRIS        PORTA
            RETLW       00H
;*****************************************************************
; Subrutina que cuenta el tiempo del TMR0
;*****************************************************************
M_TIME      CLRF        TMR0        ;Inicializa el TMR0
            CLRF        ACCA+1
            CLRF        ACCA
TLOOP       INCFSZ      ACCA+1,F
            GOTO        ENDCHK
            INCFSZ      ACCA,F
            GOTO        ENDCHK
            GOTO        END_M
ENDCHK      BTFSS       TMR0,C      ;Comprueba el paso del TMR0
            GOTO        TLOOP
END_M       MOVF        TMR0,W
            RETLW       00H
;*****************************************************************
; Programa principal
;*****************************************************************
OHMS        MOVLW       B'00000011' ;Prepara RA0, RA1 a '1'
            MOVWF       PORTA       ;Para cuando se activen
            MOVLW       B'00101000' ;Selecciona flanco ascendente
            OPTION                  ;en el TMR0, Transición en
                                    ;el pin TMR0,Preescaler
                                    ;asignado al WDT
CAL         CALL        DSCHRG      ;Descarga el condensador
            MOVLW       B'00001110' ;Activa RA0
            TRIS        PORTA
            CALL        M_TIME      ;Mide el tiempo
            MOVF        ACCA+1,W
            MOVWF       TCAL+1      ;Almacena LSB
            MOVF        ACCA,W
            MOVWF       TCAL        ;Almacena MSB
MEAS        CALL        DSCHRG      ;Descarga el condensador
            MOVLW       B'00001101' ;Activa RA1
            TRIS        PORTA
            CALL        M_TIME      ;Mide el tiempo
            MOVLW       RCALLS      ;Valor LSB de la R de calibración
            MOVWF       ACCB+1
            MOVLW       RCALMS      ;Valor MSB de la R de calibración
            MOVWF       ACCB
            CALL        MPY         ;Multiplica ACCA (medida) *
                                    ;ACCB (R de calibración)
            MOVF        TCAL+1,W
            MOVWF       ACCA+1
            MOVF        TCAL,W
            MOVWF       ACCA
            CALL        DIV         ;Divide ACCB (medida*R)/ACCA (TCAL)
                                    ;El resultado queda en ACCB
            NOP
            NOP
            NOP                     ;Operación a realizar con
                                    ;la resistencia obtenida
            GOTO        OHMS        ;Repite el cálculo de la R
; INICIO DE LA EJECUCIÓN DEL PROGRAMA
            ORG         0x1FF
            GOTO        OHMS        ;Comienzo del programa
            END
```