IV) USO BÁSICO DEL PIC

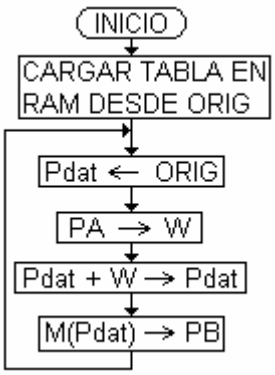
El programa fuente debe ser ordenado y tener muchos comentarios.
lineamientos a seguir
Nombre y explicación del programa.
Datos del autor y fecha de creación.
Selección, configuración e identificación del micro con LIST, __CONFIG e __IDLOCS.
Declaración de los nombres de los registros SFR y de sus bit.
Declaración de los nombres de las variables y de las constantes propias a usar.
Ubicación del inicio del programa en 000h y salto al principio del mismo por encima de las subrutinas.
Ubicación del Vector de Interrupción en 004h.
Ubicación de inicio de las subrutinas antes de 100h.
Ubicación del programa principal a partir de 100h.
Seteo de los periféricos: puertos y TIMER0
Seteo de interrupciones (INTCON)
Carga de tablas a usar en la memoria RAM
Desarrollo del programa según el diagrama fuente a partir de INICIO
Desarrollo de las subrutinas largas que comenzaron en la primera página (00h-FFh)
Todas las líneas con la mayor cantidad de comentarios posibles.
Terminar el programa fuente con el comando END y un retorno de carro

Manejo De Puertos

Los puertos en este micro son tratados como registros internos comunes. Se les puede aplicar cualquier operación igual que a los demás. Cuando una instrucción implica un puerto, los bits (pines) seteados como entradas son leídos al comienzo del ciclo de instrucción, y en los configurados como salidas se usa el último valor escrito. La escritura sobre un pin de puerto seteado como entrada no tiene efecto.
Una vez escrito un puerto este mantiene su valor hasta una nueva escritura, inclusive en modo de bajo consumo. La lectura en cambio se produce en un periodo de tiempo pequeño comparado con el ciclo de instrucción. Los puertos pueden ser escritos bit a bit o todo el byte en conjunto.
La configuración de los pines se realiza en el registro TRISA (85h) o TRISB (86).
Si TRISA,1 vale uno el pin es configurado como entrada, y si vale cero es configurado como salida.
El modo de funcionamiento de un pin puede variar a lo largo del programa. Después de un reset los pines quedan modificados como entradas.
Recordar que los registros TRIS se hallan en el banco 1. Antes de escribirlos hay que pasar del banco cero al uno, poniendo a uno el bit STATUS,RP0.

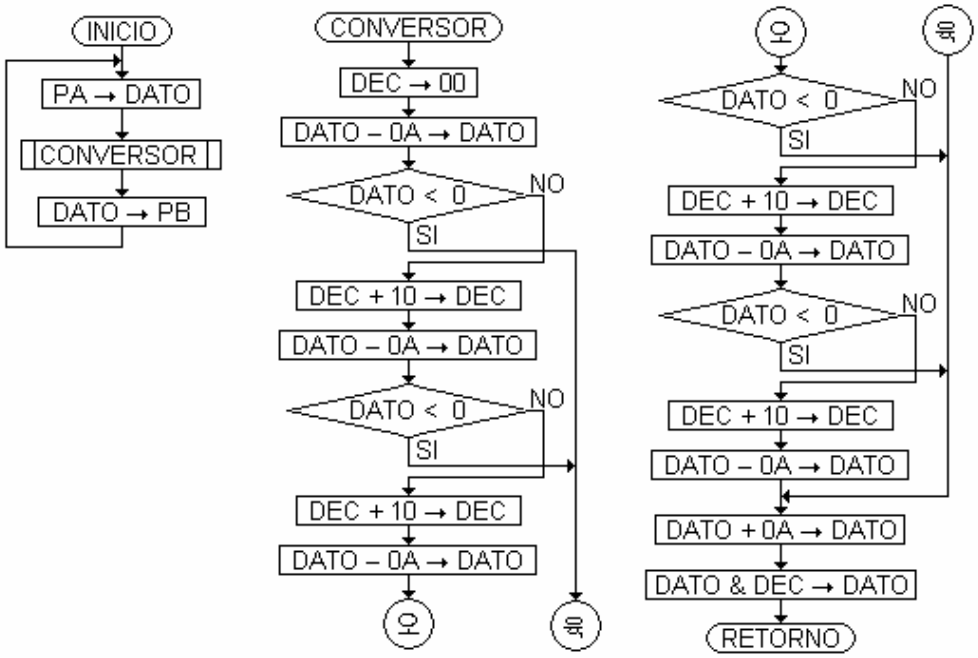
Ejercicio:
Realizar un conversor binario a BCD. La entrada son los 5 bit del Puerto A, la salida los dos nibles del Puerto B.
La solución se puede intentar por varios métodos. Uno seria con una tabla bi-univoca. Debido a que a cada valor binario le corresponde una pareja de dígitos BCD. En el diagrama de flujo no es necesario aclarar el seteo de los periféricos, pero agregarlo no es un error.

```
INICIO
;Cargar los 32 valores de la tabla a partir de ORIG
LAZO
movlw  ORIG
movwf  Pdat
movf   PORTA,W
addwf  Pdat,F
movf   INDF,W
movwf  PORTB
goto   LAZO
```

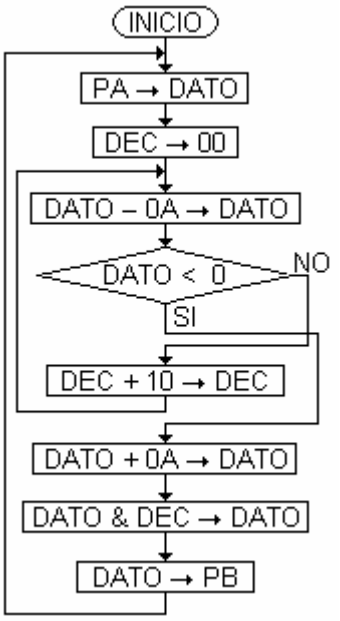


Otra solución es mediante cálculos matemáticos. Para convertir un numero hexadecimal menor de 1000_D, primero averiguamos cuantas centenas tiene. O sea se divide entre 100_D o 64_H. Al resto se le divide entre 10_D o $0A_H$ para averiguar el número de decenas y el resto son las unidades.

La división se implementa mediante sucesivas restas. Como en este ejercicio el valor máximo de entrada es $1F_H$ o 31_D, solo es necesario contar las decenas.



La subrutina CONVERSOR tiene una estructura de 3 bloques que se repite 4 veces. Se convierte en un lazo usando la variable DATO como variable de lazo, cuando vale menos de cero.
Este diagrama se puede asociar al siguiente programa fuente:

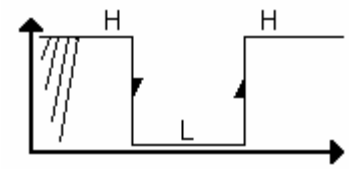



```
;Conversor de binario 5 bit a BCD de dos dígitos
;Manejo puertos, lazos, iteraciones y cálculos.
;Seleccionamos y configuramos el micro.
        LIST        p=16f84a        ;Indico compilar
                                    ; para el PIC16F84A
        __CONFIG    0x3FF1          ;Selecciono XT, PWRTE,
                                    ;No WDT  y No CP
        __IDLOCS    0x0001          ;Elijo un número que
                                    ;identifique  el software

;-------Declaro las variables a usar en el programa fuente-----
W               EQU     .0          ;El destino es el reg. W
F               EQU     .1          ;El destino es el propio reg

;----- Register Files----------------------------------------------------
INDF            EQU     H'00'       ;Registro auxiliar para
                                    ;direcc. Indirecto por registro
TMR0            EQU     H'01'       ;Buffer del TIMER0
PCL             EQU     H'02'       ;Parte baja del PC
STATUS          EQU     H'03'       ;Palabra de estado del programa
FSR             EQU     H'04'       ;Puntero de la memoria RAM para direcc. Indirecto
PORTA           EQU     H'05'       ;Buffer de los pines RA0-RA4
PORTB           EQU     H'06'       ;Buffer de los pines RB0-RB7
INTCON          EQU     H'0B'       ;Registro de control de interrupciones
OPCION          EQU     H'81'       ;Registro de configuración de periféricos
TRISA           EQU     H'85'       ;Registro de control de dirección del puerto A
TRISB           EQU     H'86'       ;Registro de control de dirección del puerto B

;----- STATUS Bits --------------------------------------------------------
IRP             EQU     .7          ;Bit selector bancos del FSR, no se usa en el PIC16F84
RP1             EQU     .6          ;Bit alto de selección de bancos RAM en direcc. directo
```

```
RP0             EQU     .5              ;Bit bajo de selección de banco RAM en direcc. directo
TO              EQU     .4              ;Bit de estado del micro
PD              EQU     .3              ;Bit de estado del micro
Z               EQU     .2              ;Bandera de resultado cero
DC              EQU     .1              ;Bandera de semiacarreo en sumas y restas
CY              EQU     .0              ;Bandera de acarreo en sumas y restas

;----- INTCON Bits ----------------------------------------------------------
GIE             EQU     .7              ;Habilitación general de todas las interrupciones
EEIE            EQU     .6              ;Mascara de interp final del ciclo de escritura EEPROM
T0IE            EQU     .5              ;Mascara de interrupción por desborde del TIMER0
INTE            EQU     .4              ;Mascara de interrupción por evento en RB0
RBIE            EQU     .3              ;Mascara de interrupción por cambios en RB4-RB7
T0IF            EQU     .2              ;Bandera de aviso de desborde de TIMER0
INTF            EQU     .1              ;Bandera de aviso de evento en RB0
RBIF            EQU     .0              ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits ----------------------------------------------------------
RBPU            EQU     .7              ;Habilitación de pull-up en el PB
INTA            EQU     .6              ;Selector de flanco activo de RB0
TOCS            EQU     .5              ;Selector de fuente de incremento del TIMER0
TOSE            EQU     .4              ;Selector de flanco activo de RA4
PSA             EQU     .3              ;Selector de uso del pre-escalador
PS2             EQU     .2              ;Bit de configuración del pre-escalador
PS1             EQU     .1              ;Bit de configuración del pre-escalador
PS0             EQU     .0              ;Bit de configuración del pre-escalador

;Asocio el nombre de cada variable a una posición de memoria en los GPR
DEC             EQU     0x10
DATO            EQU     0x11

;En este micro la primera linea que busca para ejecutar esta en 0x0000

                ORG     0x0000
RESET
                goto    PRINCIPIO       ;Salto por enciam de las subrutinas

                ORG     0x0004
SUBRUTINAS

                ORG     0x0100
PRINCIPIO
;Lo primero es setear el periferico puertos
                bsf     STATUS,RP0      ;CAmbio al banco 1
                movlw   0xFF
                movwf   TRISA           ;Puerto A todo entradas
                clrf    TRISB           ;Puerto B todo salidas
;Termino de setear perifericos
                bcf     STATUS,RP0      ;Cambio al banco 0

;Comienzo el programa segun el diagrama de flujo
INICIO
                movf    PORTA,W                 ;Preparo dato para procesarlo
                clrf    DEC                     ;Pongo a cero las decenas
BUCLE
                movlw   0x0A
                subwf   DATO,F
                btfss   STATUS,CY               ;Si DATO < 0 entonces CY=0
                goto    IRME
```

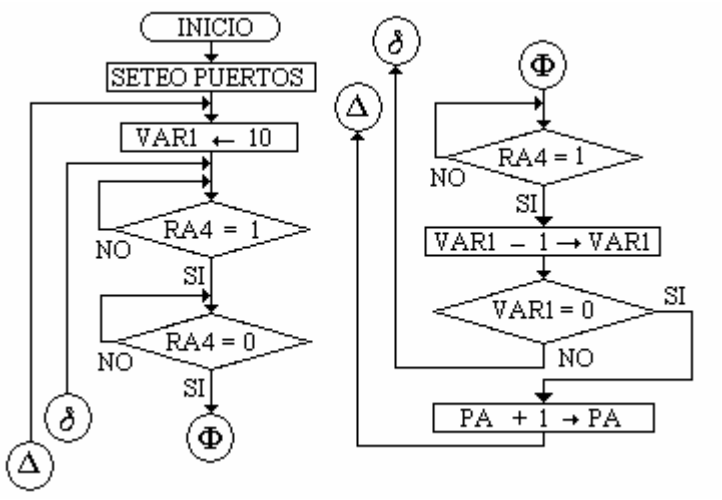
```
                movlw  0x10
                addwf  DEC,F            ;Incremento las decenas
                goto   BUCLE            ;voy a ver si queda otra decena
IRME
                movlw  0x0A
                addwf  DATO,W           ;Reconstruyo las unidades
                iorwf  DEC,W
                movwf  DATO
                movwf  PORTB
                goto   INICIO

                END
```

Detección de pulsos



Los pulsos son eventos de una señal, tienen un principio y un fin. El peor de los casos es cuando antes de estar estable la señal hay un valor indefinido. Para el caso del pulso negativo, antes de ocurrir la señal debe presentar un valor alto estable. El pulso comienza con un flanco de bajada, a continuación un valor bajo estable y terminar con un flanco de subida. El flanco es una discontinuidad en el valor de la señal que es difícil de detectar, pero si verificamos valores antes y después podemos detectar que ocurrió. Por software no se puede detectar flancos. El nivel bajo en este pulso negativo es el nivel activo. Cuando se trabaja con un tren de pulsos se puede obviar el detectar la región previa a estar estable la señal.

Ejercicio:
Detectar 10 pulso por RA4 e incrementar un contador de 4 bit en RA0-RA3. El pulso en nuestros módulos son activos en L (nivel 0).
Lo hacemos mediante detección de pulsos negativos en RA4. El pulso consta de un flanco de bajada, un estado activo bajo, un flanco de subida y estado de reposo alto.



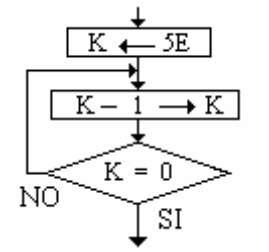
```
;Programa para detectar 10 pulsos por RA4.
;Al principio del programa fuente seteo el microprocesador
                LIST        p=16f84A        ;Indico compilar para el PIC16F84A
                __CONFIG    0x3FF1          ;Selecciono XT, PWTE, No WDT  y No CP
                __IDLOCS    0x0002          ;Elijo un número cualquiera para
                                            ;identificar el software
;-------Declaro las variables a usar en el programa fuente-----

W               EQU     .0              ;Destino el registro de trabajo
F               EQU     .1              ;Destino el propio registro direccionado

;----- Register Files------------------------------------------------------
INDF            EQU     H'00'           ;Registro auxiliar para direcc. Indirecto por registro
TMR0            EQU     H'01'           ;Registro de buffer del TIMER0
PCL             EQU     H'02'           ;Parte baja del PC
STATUS          EQU     H'03'           ;Palabra de estado del programa
FSR             EQU     H'04'           ;Puntero de la memoria RAM para direcc. Indirecto
PORTA           EQU     H'05'           ;Buffer de los pines RA0-RA4
PORTB           EQU     H'06'           ;Buffer de los pines RB0-RB7
INTCON          EQU     H'0B'           ;Registro de control de interrupciones
OPCION          EQU     H'81'           ;Registro de configuración de periféricos
TRISA           EQU     H'85'           ;Registro de control de dirección del puerto A
TRISB           EQU     H'86'           ;Registro de control de dirección del puerto B

;----- STATUS Bits --------------------------------------------------------
IRP             EQU     .7              ;Bit selector bancos del FSR, no se usa en el PIC16F84
RP1             EQU     .6              ;Bit alto de selección de bancos RAM en direcc. directo
RP0             EQU     .5              ;Bit bajo de selección de banco RAM en direcc. directo
TO              EQU     .4              ;Bit de estado del micro
PD              EQU     .3              ;Bit de estado del micro
Z               EQU     .2              ;Bandera de resultado cero
DC              EQU     .1              ;Bandera de semiacarreo en sumas y restas
CY              EQU     .0              ;Bandera de acarreo en sumas y restas

;----- INTCON Bits --------------------------------------------------------
GIE             EQU     .7              ;Habilitación general de todas las interrupciones
EEIE            EQU     .6              ;Mascara de interp final del ciclo de escritura EEPROM
T0IE            EQU     .5              ;Mascara de interrupción por desborde del TIMER0
INTE            EQU     .4              ;Mascara de interrupción por evento en RB0
RBIE            EQU     .3              ;Mascara de interrupción por cambios en RB4-RB7
T0IF            EQU     .2              ;Bandera de aviso de desborde de TIMER0
INTF            EQU     .1              ;Bandera de aviso de evento en RB0
RBIF            EQU     .0              ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits --------------------------------------------------------
RBPU            EQU     .7              ;Habilitación de pull-up en el PB
INTA            EQU     .6              ;Selector de flanco activo de RB0
TOCS            EQU     .5              ;Selector de fuente de incremento del TIMER0
TOSE            EQU     .4              ;Selector de flanco activo de RA4
PSA             EQU     .3              ;Selector de uso del pre-escalador
PS2             EQU     .2              ;Bit de configuración del pre-escalador
PS1             EQU     .1              ;Bit de configuración del pre-escalador
PS0             EQU     .0              ;Bit de configuración del pre-escalador

;Asocio el nombre de cada variable a una posición de memoria en los GPR
DIEZ            EQU         0x0a        ;Nro de pulsos a detectar
VAR1            EQU         0x12        ;Registro para contar nro de pulsos
RA4             EQU         .4          ;Pin a usar como entrada de pulsos
```

```
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
;Comienzo el programa indicando a partir de donde se colocarán las instrucciones.
            ORG     0x000
COMIENZO
            goto            PRINCIPIO

            ORG     0x004
VECTOR_DE_INTERRUPCION

SUBRUTINAS
            ORG     0x100
PRINCIPIO
;Seteo los periféricos
            bsf             STATUS,RP0      ;Paso al banco 1
            movlw           B'11110000'
            movwf           TRISA           ;Seteo los pines del puerto A
                                            ;todos como salidas menos RA4
                                            ;Los tres más altos no están implementados
            bcf             STATUS,RP0      ;Paso al banco 0

INICIO
            clrf            PORTA           ;Pongo a cero los pines RA0-RA3
LAZO1
            movlw           DIEZ
            movwf           VAR1            ;Cargo VAR1 con el nro de
                                            ;pulsos a detectar
LAZO2
            btfss           PORTA,RA4
            goto            LAZO2           ;Mientras no sea 1 sigo acá
LAZO3
            btfsc           PORTA,RA4
            goto            LAZO3           ;Estado de reposo mientras no
                                            ;halla un flanco de bajada
LAZO4
            btfss           PORTA,RA4
            goto            LAZO4           ;La ocurrencia de un flanco de
                                            ;subida marca el fin del pulso

            decfsz          VAR1,F          ;Cuento decrementando VAR1
            goto            LAZO2           ;Si no es el décimo sigo esperando

            incf            PORTA,F         ;Cada diez pulso incremento PORTA, solo
                                            ;lo vemos en el nible bajo. El superior o es
                                            ;salida o no existe.
            goto            LAZO1

            END                             ;Debo dejar un renglón en
                                            ;limpio después de END
```

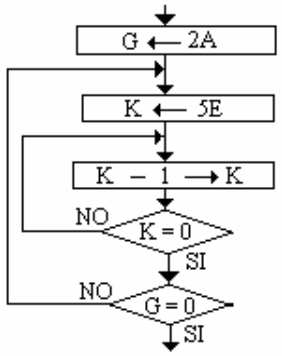
Contadores y Temporizadores:

Son estructuras lógicas que permiten contar eventos.
El primero comúnmente se refiere a eventos o señales, su período es variable según qué lo haga actuar, son de pequeño valor y baja frecuencia.
El segundo se refiere a medir el tiempo, su uso es en general para producir retardos. Lo normal es usar el ciclo de máquina para modificarlo, por lo que no es raro que sean de gran tamaño. Inclusive es común usar varios registros para formar una cifra mayor.

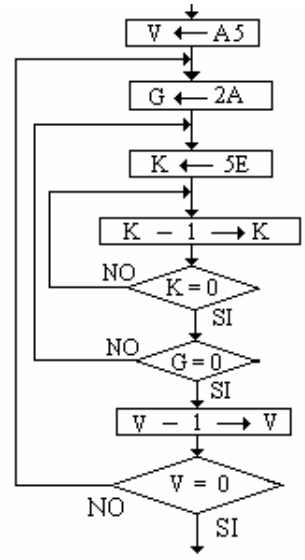
El micro es más rápido que la mayoría de los actuadores o que las acciones del operador. Es vital en algunos casos generar demoras para poder esperar resultados, ahí el lazo en conjunto con un contador permite hacer un retardo o espera. Los micros más modernos permiten colocar al chip en modo STANDBY o POWER DOWN, donde no se ejecutan instrucciones ni se avanza en el programa. Se sale de dicho estado si se presentan ciertas señales externas como peticiones de interrupción o reset.



En los casos que no se puede detener el micro o se necesita seguir haciendo otras funciones se pueden implementar retardo. El temporizador es una variable dentro de un lazo, donde en cada iteración se le suma o resta uno. Asignarle un valor antes del lazo y decrementar la variable del contador es la más usual. Debido a que es más sencillo detectar cuando se hace cero (bandera Z=1) que cualquier otro valor.



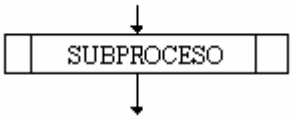
Si con una variable no alcanza se pueden usar dos contadores encadenados. Encadenar dos contadores permite dar mayor exactitud al contador completo, y más fácil de variar si queremos modificarlo que si fuese una variable sola. Además muchas veces los registros del micro son pequeños para el valor que se necesita del retardo. En el ejemplo suministrado se realiza el lazo central 5E x 2A veces. El cálculo exacto de cuánto dura un retardo depende del número de instrucciones ejecutadas antes de terminar el contador, cuanto dura cada instrucción, etc. A su vez la duración de la instrucción depende del oscilador principal, una diferencia de nanosegundos en cada ciclo de instrucción se convierte en varios minutos al cabo de un rato. El ajuste final se debe realizar en forma empírica, así que cuanto más ajuste pueda tener el programa más exacto puede ser el temporizador.



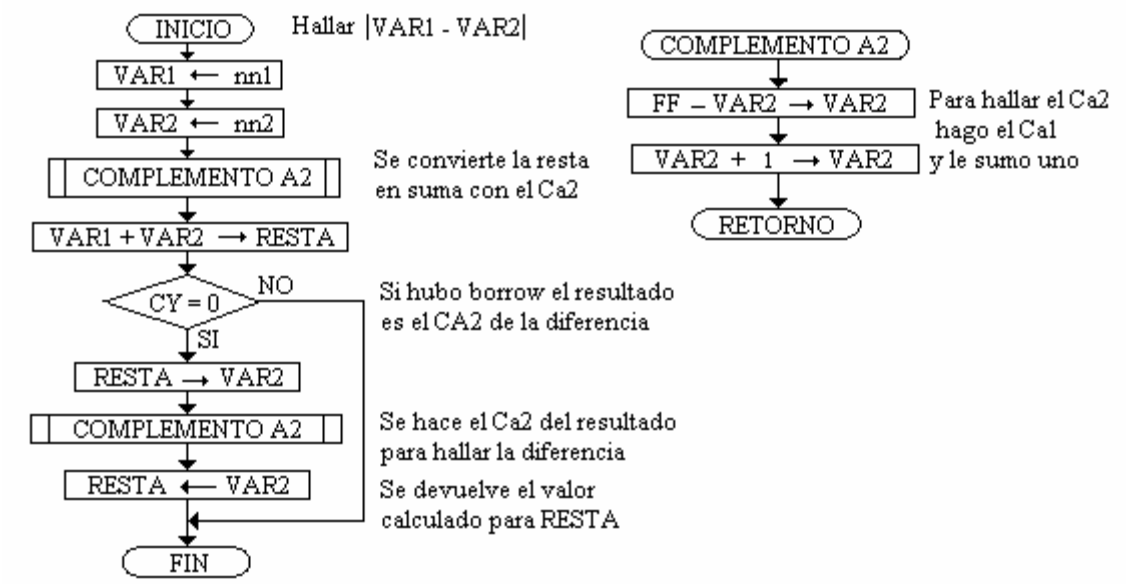
El encadenamiento de variables para ampliar un temporizador o contador puede ser tan grande como se necesite. Cada variable agrega 8 bit al contador/temporizador.
Se llaman contadores de 8 bit, 16 bit, 24 bit, etc

SUBRUTINAS

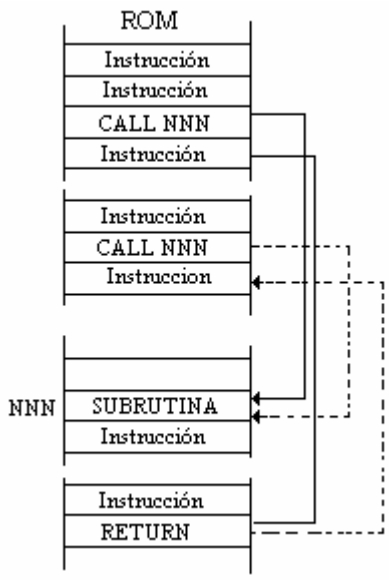
Son programas, usados en el programa principal. Puede ser más pequeños o mayores que el programa principal.



El diagrama se debe desarrollar aparte (sin ramas que lo conecten) del diagrama principal. En el bloque de inicio va el nombre de la subrutina.
La conexión entre el programa principal y la subrutina son las variables que se comparten. Es decir las usa el programa principal y las usa la subrutina. Se debe tener cuidado de no alterar ninguna variable que se necesite con su valor original después de ejecutar la subrutina y regresar al programa principal.
La unión entre la subrutina y el programa principal es la o las variables que usen en común. El programa principal en esas variables entrega los datos de entrada a la subrutina y en esas variables la subrutina entrega sus datos de salida. Con ellas el programa principal continúa trabajando.

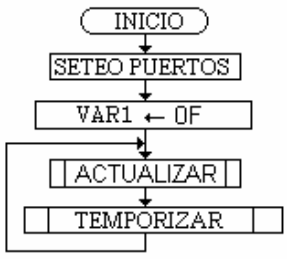


El ejemplo anterior es un programa que halla la diferencia entre dos números cargados en forma directa.



La subrutina puede ser invocada tantas veces como se necesite. Es necesario asignar a las variables que usa el valor adecuado. En el ejemplo la variable de entrada y de salida es VAR2. Para operar en la subrutina con RESTA se copio su valor en VAR2. Al volver de la misma se copia el valor de la variable de salida en RESTA. Al finalizar el programa en la variable resta se encuentre el valor de la diferencia entre nn1 y nn2.
Si la subrutina altera alguna variable de la que no queremos perder su valor debemos hacer un respaldo o copia de ella.
Al finalizar la subrutina esta automáticamente vuelve al programa principal. Se retorna a la siguiente instrucción después del llamado a subrutina. Esto es posible gracias a la pila, la cual es una estructura lógica que permite almacenar datos en forma temporal. En algunos micros es un área de la memoria RAM definida por el usuario, en otros son registros especiales para tal fin. La pila puede definirse por software en la RAM o por hardware en registros especiales. Al menos se debe guardar el valor de contador de programa. La profundidad o capacidad de la pila limita la posibilidad de encadenar subrutinas dentro de una subrutina.
La pila del PIC16F84A es un banco de 8 registros (por hardware) de 13 bit usados únicamente para este propósito.
El o los bloques de finalización se llaman de retorno. Puede tener más de un bloque de finalización. Va al terminar la subrutina o estar en una de las salidas de un bloque de decisión, a este retorno se le llama **retorno condicionado**.

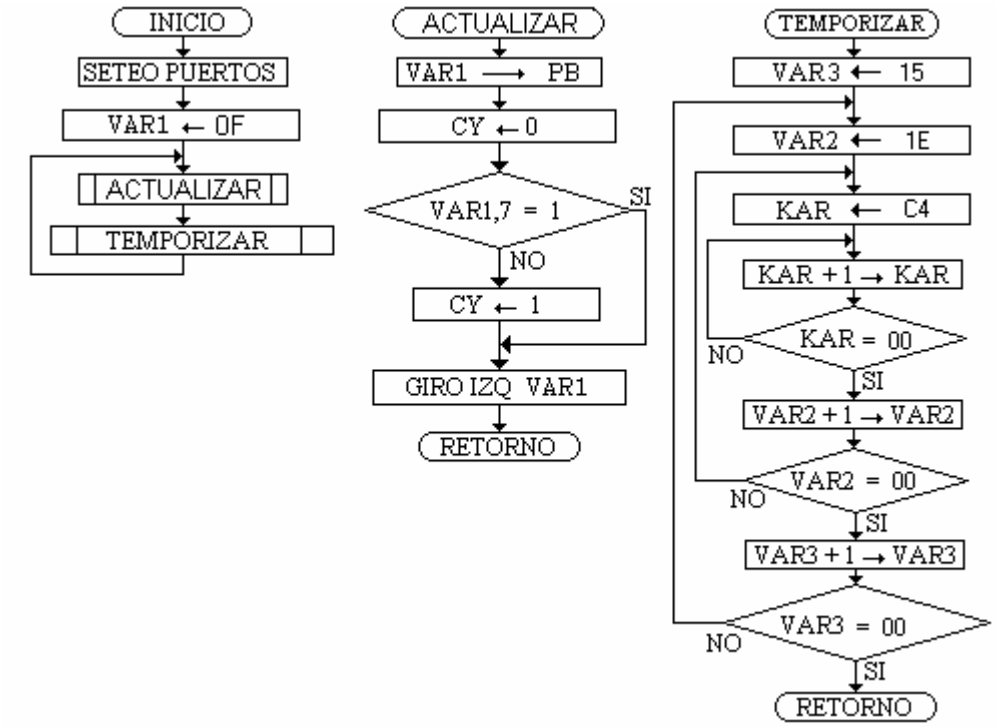
Ejercicio:
Realizar un contador Johnson de 8 bit que se muestre en el puerto B. Cada elemento debe ser mostrado durante un segundo aproximadamente.
El realizar un diagrama de flujo es mas facil si dividimos el problema en etapas. En este ejemplo tenemos dos acciones, elegir el elemento a mostrar y contar el tiempo solicitado.
Un diagrama de flujo que contemple estas dos características seria:
Ahora por separado se considera cada aspecto del problema y se desarrollan las subrutinas correspondientes.



Generar la secuencia se puede hacer de dos formas. O se cargan datos desde una tabla, o se calcula una relación matemática entre los elementos. De la observación de la secuencia se deduce que girando hacia la izquierda los bits de cada elemento, con el valor en carry adecuado, se obtiene el siguiente elemento. Podemos ver que en los

primeros 8 elementos el carry debe estar en 1 y en los demás debe estar en 0. Siempre ingresa por el bit 0 el opuesto del bit 7. Colocando el complemento del bit 7 en el carry antes de girar alcanza.

El cálculo de la temporización se hace mejor con el programa fuente. Pero por aproximación se necesitan 3 registros en el temporizador.



El programa fuente para este diagrama seria:

```
; Contador Johnson con un mínimo de recursos.
; primero seteamos Palabra de Configuración
                LIST        p=16f84a        ;Indico compilar para el PIC16F84A
                __CONFIG    0x3FF1          ;Selecciono XT, PWTE, No WDT  y No CP
                __IDLOCS    0x0003          ;Identifico el software

;Declaro los nombres de todos los SFR (registros de uso especial)
;Como el micro trata a los puertos como registros  internos se le puede aplicar cualquier
operación
; valida para los SFR
;-------Declaro las variables a usar en el programa fuente-----

W               EQU     .0              ;Destino el registro de trabajo
F               EQU     .1              ;Destino el propio registro direccionado

;----- Register Files----------------------------------------------------
INDF            EQU     H'00'           ;Registro auxiliar para direcc. Indirecto por registro
TMR0            EQU     H'01'           ;Registro de buffer del TIMER0
PCL             EQU     H'02'           ;Parte baja del PC
STATUS          EQU     H'03'           ;Palabra de estado del programa
FSR             EQU     H'04'           ;Puntero de la memoria RAM para direcc. Indirecto
PORTA           EQU     H'05'           ;Buffer de los pines RA0-RA4
PORTB           EQU     H'06'           ;Buffer de los pines RB0-RB7
INTCON          EQU     H'0B'           ;Registro de control de interrupciones
OPCION          EQU     H'81'           ;Registro de configuración de periféricos
TRISA           EQU     H'85'           ;Registro de control de dirección del puerto A
TRISB           EQU     H'86'           ;Registro de control de dirección del puerto B
```

```
;----- STATUS Bits --------------------------------------------------------
IRP             EQU     .7              ;Bit selector bancos del FSR, no se usa en el PIC16F84
RP1             EQU     .6              ;Bit alto de selección de bancos RAM en direcc. directo
RP0             EQU     .5              ;Bit bajo de selección de banco RAM en direcc. directo
TO              EQU     .4              ;Bit de estado del micro
PD              EQU     .3              ;Bit de estado del micro
Z               EQU     .2              ;Bandera de resultado cero
DC              EQU     .1              ;Bandera de semiacarreo en sumas y restas
CY              EQU     .0              ;Bandera de acarreo en sumas y restas

;----- INTCON Bits --------------------------------------------------------
GIE             EQU     .7              ;Habilitación general de todas las interrupciones
EEIE            EQU     .6              ;Mascara de interp final del ciclo de escritura EEPROM
T0IE            EQU     .5              ;Mascara de interrupción por desborde del TIMER0
INTE            EQU     .4              ;Mascara de interrupción por evento en RB0
RBIE            EQU     .3              ;Mascara de interrupción por cambios en RB4-RB7
T0IF            EQU     .2              ;Bandera de aviso de desborde de TIMER0
INTF            EQU     .1              ;Bandera de aviso de evento en RB0
RBIF            EQU     .0              ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits --------------------------------------------------------
RBPU            EQU     .7              ;Habilitación de pull-up en el PB
INTA            EQU     .6              ;Selector de flanco activo de RB0
TOCS            EQU     .5              ;Selector de fuente de incremento del TIMER0
TOSE            EQU     .4              ;Selector de flanco activo de RA4
PSA             EQU     .3              ;Selector de uso del pre-escalador
PS2             EQU     .2              ;Bit de configuración del pre-escalador
PS1             EQU     .1              ;Bit de configuración del pre-escalador
PS0             EQU     .0              ;Bit de configuración del pre-escalador

;------------ Declaro todas las constantes, etiquetas y variables a usar----------
VAR1            EQU             0x1F
VAR2            EQU             0x20
VAR3            EQU             0X21
KAR             EQU             0X22
NN1             EQU             0x15
NN2             EQU             0x1E
NN3             EQU             0xC4

; El comienzo de programa es en la posición 000h y salto por encima del comienzo de todas las
;subrutinas.
                ORG             0x000
COMIENZO
                goto            PRINCIPIO

                ORG             0x004
VECTOR_DE_INTERRUPCION

; Inserto la subrutina primero por que el fabricante indica que deben comenzar antes de 0xFF
; Coloco el comienzo de las subrutinas antes de la posición 0x0FF.
TEMPORIZAR
                movlw           NN1             ;Variando los valores de VAR3, VAR2 y KAR se
                movwf           VAR3            ;puede modificar el tiempo de led ON
LOOP3
                movlw           NN2
                movwf           VAR2
LOOP2
                movlw           NN3
                movwf           KAR
```

```
LOOP1
                decfsz      KAR,F
                goto        LOOP1

                decfsz      VAR2,F
                goto        LOOP2

                decfsz      VAR3,F
                goto        LOOP3
                return

ACTUALIZAR
                movf        VAR1,F
                movwf       PORTB       ;Cargo el valor actualizado en el PB
                bcf         STATUS,CY   ; 0 → CY
                btfss       VAR1,.7     ;Verifico el valor del MSB de VAR1.
                bsf         STATUS,CY   ;Si es 0 entonces pongo a uno el  CY
                rlf         VAR1,F      ;Giro el puerto VAR1
                return
FIN_SUBRUTINA

;Desarrollo el programa principal después de las subrutinas.
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-
                ORG         0x100
PRINCIPIO
;Lo primero es setear los puertos
                bsf         STATUS,RP0  ;Cambio al Banco 1
                movlw       0x00        ;Cargo valor para que el puerto B
                movwf       TRISB       ; sea salidas
                                        ;El puerto A no lo uso
                bcf         STATUS,RP0  ;Cambio al Banco 0
;Termino de setear los periféricos y comienza el programa propiamente dicho
INICIO
                clrf        VAR1        ;Inicializo variable de intercambio con el PB
LOOP
                call        ACTUALIZAR
                call        TEMPORIZAR
                goto        LOOP
                                        ;El programa no tiene fin
                END                     ;Pero el archivo a compilar si
```

MATRICES Y SERIES:

Las **series** permiten trabajar con grupos de valores tratándolos como un todo, por ejemplo dentro de un lazo. Se define el puntero de inicio y después se le actualiza o indexa para ir seleccionando cada uno de sus elementos. Se usa el número de elementos para cargar la variable de lazo.

Las **matrices** sirven igual que las series, pero usan más de un puntero. Su utilidad radica en que permiten organizar los datos dentro de la memoria.

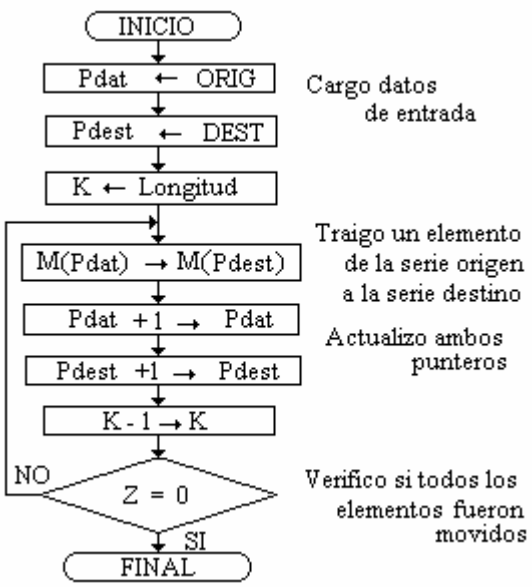
En el manejo o representación de matrices y series se usa un puntero por cada dimensión. La selección de datos comúnmente se hace en orden secuencial, o sea se toman uno a uno y en forma consecutiva y además se les aplica a todos un proceso parecido. Por estas características es muy conveniente el uso de lazos en el manejo de dichas estructuras. En cada repetición se modifican los vectores de entrada a la matriz y se selecciona un nuevo dato.

La matriz más simple es la serie, solo se necesita saber el valor del puntero, que indicará la posición del primer elemento, y el número de datos que componen la serie llamado longitud.

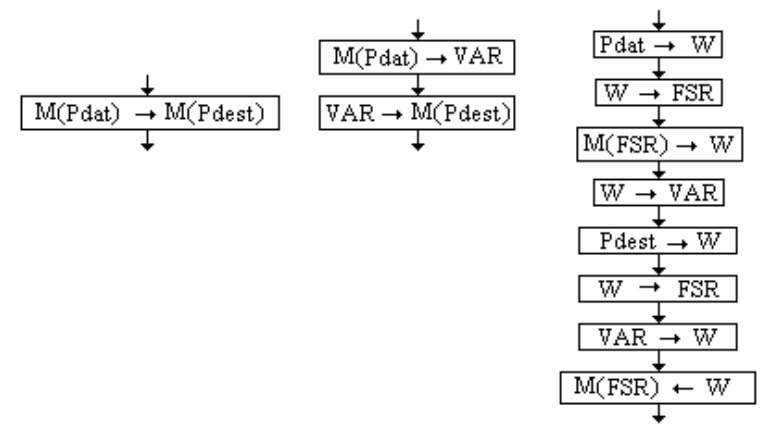
En el PIC16F84A solo se tiene un puntero de memoria RAM, el registro FSR (posición 04h). Permite direccionar cualquier posición de RAM en cualquiera de los dos bancos.

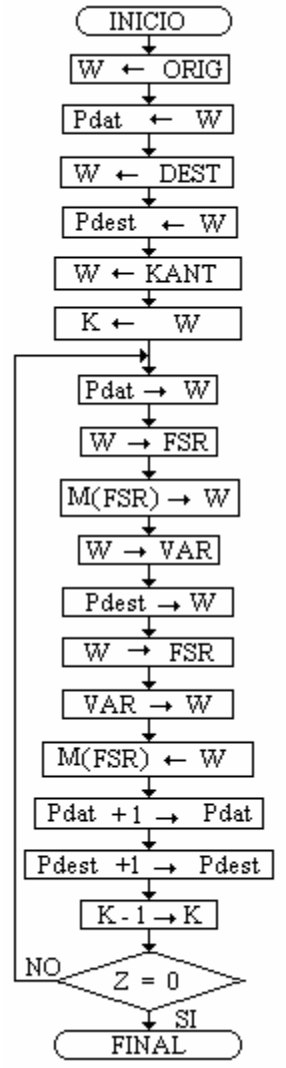
Ejercicio: Dado una serie de *KANT* elementos a partir de la posición *ORIG*, colocarla a partir de la posición DEST.

Hay un solo puntero físico (FSR) en el micro, por lo que es necesario ir alternando la información de cada uno de los punteros lógicos del programa.
La mayoría de los micros, y en particular el PIC16F84A, no pueden mover de una posición de memoria direccionada por un puntero a otra posición de memoria direccionada de la misma forma. Por lo tanto hay que usar una variable que sirva de intermediario.
Antes de entrar en el lazo a todas las variables se le asigna un valor inicial. Valor con el que operan durante la primera vez que recorran el lazo. Al salir de la iteración vemos que su valor se ha modificado.
K = 0
Pdest = DEST + K + 1
Pdat = ORIG + K + 1
Observar que los punteros en este diagrama terminan valiendo uno más que la posición de los últimos elementos.
Personalizar este diagrama presenta una particular dificultad en el bloque de movimiento con el origen y el destino direccionados en forma indirecta por registro.



Todo movimiento pasa a través del registro W. Los punteros Pdat y Pdest se van cargando alternadamente en el registro FSR para direccionar a los elementos de la serie. La variable VAR es para almacenar momentáneamente el valor que se esta trasladando.

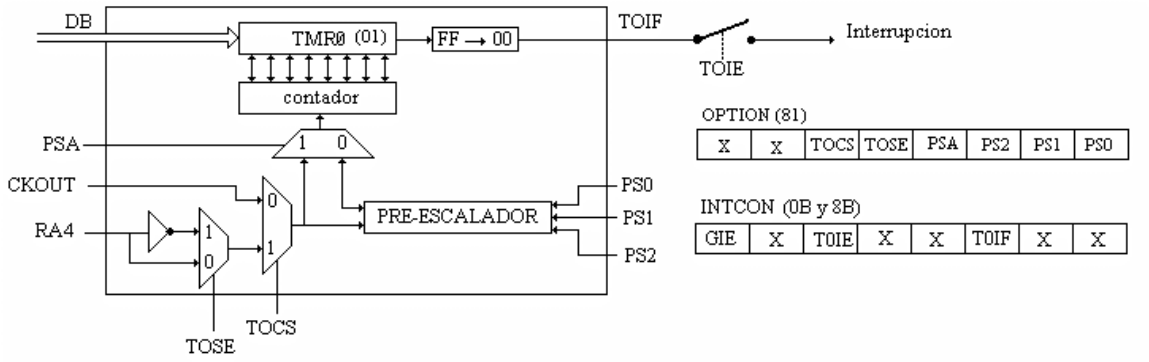





```
INICIO
        movlw  ORIG   ;Setea puntero donde esta la serie
        movwf  PDAT

        movlw  DEST   ;Setea puntero destino de la serie
        movwf  PDEST

        movlw  KANT   ;Carga el número de elementos a mover
        movwf  K

LAZO
        movf    PDAT,W
        movwf  FSR     ;Coloca el 1º puntero lógico en el puntero físico

        movf    INDF,W
        movwf  VAR     ;El dato traído es colocado en la variable de
                       ; intercambio

        movf    PDEST,W
        movwf  FSR     ;Coloca el 2º puntero lógico en el puntero físico

        movf    VAR,W  ;Trae el dato guardado en la variable de
                       ; intercambio
        movwf  INDF    ;Guarda el dato en la posición destinada

        incf    PDAT,F                  ;Incrementa puntero origen

        incf    PDEST,F         ;Incrementa puntero destino

        decfsz  K,F     ;Verifica si queda algún elemento por trasladar

        goto    LAZO

FINAL
```

TIMER 0

El modulo de TIMER0 en un periférico interno que tiene sus controles dentro del mapa de memoria RAM.
Esquema en bloques



El registro **TMR0** (posición 01) es el buffer del contador de TIMER0, cada vez que lo leemos este se actualiza con el contador. Si lo escribimos, 2 ciclos de instrucción más tarde se carga en el contador. Este contador es progresivo, o sea solo incremental. El contador no se detiene nunca en su incremento. Los bit para controlar este modulo están en los registros **OPTION** (posición 81) e **INTCON** (posición 0B).

Este contador acepta dos fuentes para incrementarse. Mediante el bit **TOCS** (OPTION,5) seleccionamos entre el reloj de instrucción **CKOUT** y el flanco del pin **RA4** (pin 3).
El flanco activo en RA4 para variar el TIMER0 puede ser de subida o bajada según el bit **TOSE** (OPTION,4). Si vale 0 es el flanco de subida y si vale 1 es la transición de 1 a 0.
El pre-escalador permite enlentecer el contador. Con los bit **PS2:PS1:PS0** (bit 2, 1 y 0 de OPTION) seteamos su valor.

Valor PS2:PS1:PS0	Valor del Divisor
000	1:2
001	1:4
010	1:8
011	1:16
100	1:32
101	1:64
110	1:128
111	1:256

Se le aplica el divisor al TIMER0 cuando el bit **PSA** (OPTION,3) vale 0. Cuando el divisor se aplica al TIMER0 toda operación que implique al TMR0 (Posición 01) también resetea el pre-escalador, pero no modifica su programación. Por Ej.: CLRF 1, MOVWF 1, … , etc.
Cuando el contador desborda de FF a 00 se activa la bandera (flag) **T0IF** (INTCON,2) , el cual si esta habilitado por **T0IE** (INTCON,5), genera una interrupción. El bit T0IF debe ser reestablecido o puesto a cero por software.
El TIMER0 puede generar interrupciones. Pero no puede sacar al micro del estado de bajo consumo.
Ejercicio:
Detectar 10 pulso por RA4 e incrementar un contador de 4 bit en RA0-RA3. El pulso en nuestros módulos son activos en L (nivel 0).

Se usa el TIMER0 para contar los eventos en RA4. Dependiendo del Nro de Eventos a contar se calcula el valor inicial del TIMER0.

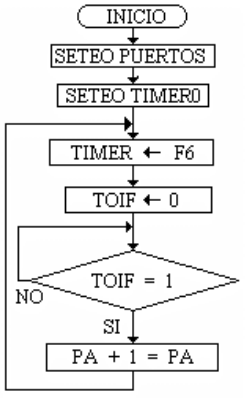
00 - Nro de Eventos = Valor inicial del TMR0

Esta forma no es muy práctica y las calculadoras no dan directamente dicho valor. Es debido a que estamos trabajando en binario limitado a 8 cifras o hexadecimal limitado a 2 cifras. Una forma más simple es:

FF - Nro de Eventos + 1 = Valor inicial del TMR0

Lo seteamos en F6h para que desborde con 10 pulsos en RA4. Para detectar el pulso solo detectamos uno de los flancos. Si se elige el flanco de bajada será el comienzo del pulso y si es el flanco de subida será el final del pulso negativo. Verificamos cuando el banderín T0IF pase a valer 1, el cual nos indica que desbordo el TIMER0. Después se debe setear el valor del TMR0 y colocar a 0 el bit T0IF. Si no se baja (poner a cero) T0IF no se puede volver a detectar o se detecta dos veces.
Para que RA4 incremente el TIMER0 se le debe configurar como entrada.
En el diagrama de flujo no es necesario aclarar el seteo de los periféricos, pero agregarlo no es un error.
Seguimos usando el esquema dado para escribir el programa fuente.

Ejercicio: Detectar 10 pulsos por RA4 e incrementar un contador binario de 4 bit en el puerto A
Se utiliza la característica del modulo de TIMER0 de que pone a uno el bit T0IF cuando desborda.
Solo se configura adecuadamente el TIMER0 y se limpia el bit INCON,T0IF

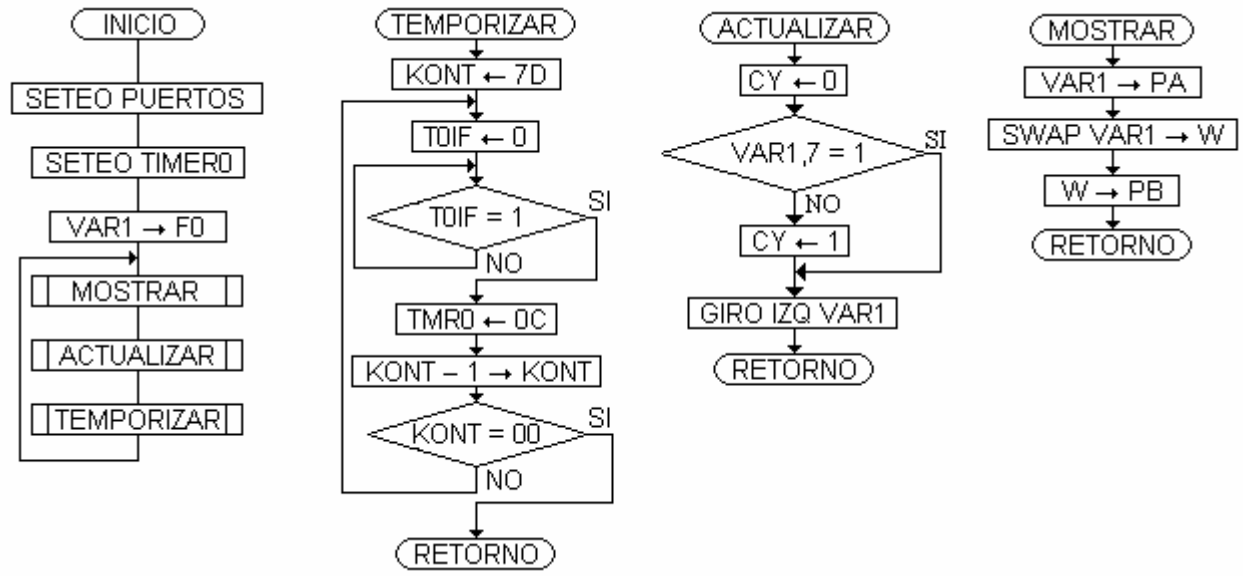



```
;Detectar 10 pulsos en RA4 mediante el desborde del TMR0.
; E incrementar un contador en RA0-RA3
;Al principio del programa fuente seteo el microprocesador
            LIST        p=16f84a        ;Indico usar PIC16F84A
            __CONFIG    0x3FF1          ;Configuro el pic
            __IDLOCS    0x0004          ;Elijo un número para
                                        ;identificar el software
;-------Declaro las variables a usar en el programa fuente-----

W           EQU     .0          ;Destino el registro de trabajo
F           EQU     .1          ;Destino el registro direccionado
```

```
;----- Register Files---------------------------------------------------
INDF            EQU    H'00'          ;Registro auxiliar para direcc. Indirecto por registro
TMR0            EQU    H'01'          ;Registro de buffer del TIMER0
PCL             EQU    H'02'          ;Parte baja del PC
STATUS          EQU    H'03'          ;Palabra de estado del programa
FSR             EQU    H'04'          ;Puntero de la memoria RAM para direcc. Indirecto
PORTA           EQU    H'05'          ;Buffer de los pines RA0-RA4
PORTB           EQU    H'06'          ;Buffer de los pines RB0-RB7
INTCON          EQU    H'0B'          ;Registro de control de interrupciones
OPCION          EQU    H'81'          ;Registro de configuración de periféricos
TRISA           EQU    H'85'          ;Registro de control de dirección del puerto A
TRISB           EQU    H'86'          ;Registro de control de dirección del puerto B

;----- STATUS Bits -----------------------------------------------------
IRP             EQU    .7             ;Bit selector bancos del FSR, no se usa en el PIC16F84
RP1             EQU    .6             ;Bit alto de selección de bancos RAM en direcc. directo
RP0             EQU    .5             ;Bit bajo de selección de banco RAM en direcc. directo
TO              EQU    .4             ;Bit de estado del micro
PD              EQU    .3             ;Bit de estado del micro
Z               EQU    .2             ;Bandera de resultado cero
DC              EQU    .1             ;Bandera de semiacarreo en sumas y restas
CY              EQU    .0             ;Bandera de acarreo en sumas y restas

;----- INTCON Bits -----------------------------------------------------
GIE             EQU    .7             ;Habilitación general de todas las interrupciones
EEIE            EQU    .6             ;Mascara de interp final del ciclo de escritura EEPROM
T0IE            EQU    .5             ;Mascara de interrupción por desborde del TIMER0
INTE            EQU    .4             ;Mascara de interrupción por evento en RB0
RBIE            EQU    .3             ;Mascara de interrupción por cambios en RB4-RB7
T0IF            EQU    .2             ;Bandera de aviso de desborde de TIMER0
INTF            EQU    .1             ;Bandera de aviso de evento en RB0
RBIF            EQU    .0             ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits -----------------------------------------------------
RBPU            EQU    .7             ;Habilitación de pull-up en el PB
INTA            EQU    .6             ;Selector de flanco activo de RB0
TOCS            EQU    .5             ;Selector de fuente de incremento del TIMER0
TOSE            EQU    .4             ;Selector de flanco activo de RA4
PSA             EQU    .3             ;Selector de uso del pre-escalador
PS2             EQU    .2             ;Bit de configuración del pre-escalador
PS1             EQU    .1             ;Bit de configuración del pre-escalador
PS0             EQU    .0             ;Bit de configuración del pre-escalador

;Asocio cada etiqueta propia con un valor fijo o posición de memoria
VAR1            EQU           0x12          ;Registro para contar nro de pulsos
DIEZ            EQU           0xF6          ;FF - Nro de pulsos a detectar + 1

;Ubico el programa en la primera posición de la memoria ROM
                ORG           0x000
COMIENZO
                goto          PRINCIPIO
                ORG           0x004
VECTOR_DE_INICIO

SUBRUTINAS
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*
                ORG           0x100
PRINCIPIO
```

```
;----------Seteo puertos
                bsf             STATUS,RP0   ;Cambio al banco 1
                movlw           b'11110000'
                movwf           TRISA        ;Seteo pines del puerto A
;-----------Seteo funcionamiento del modulo de TIMER0
                bsf             OPCION,PSA   ;Selecciono no usar divisor
                bcf             OPCION,TOCS          ;Selecciono como fuente del
                                                     ;contador los flancos en RA4
                bcf             OPCION,TOSE          ;Detecto el pulso cuando termina

                bcf             STATUS,RP0   ;Cambio al banco 0

INICIO
;-----------Seteo el valor inicial del timer0
                movlw           DIEZ         ;Valor q incrementado 10 desborde TMR0
                movwf           TMR0         ;Inicializo el contador del TIMER0

                bcf             INTCON,T0IF  ;Bajo bandera de desborde de TIMER0
ESPERA
                btfss           INTCON,T0IF  ;Cuando T0IF = 1 desborde  TMR0
                goto            ESPERA

                incf            PORTA,F      ;Incremento contador del nible bajo de PA
                goto            INICIO
                END                          ;Se debe agregar un <ENTER> o renglón
                                             ;vacío a continuación
```

Ejercicio:
Realizar un contador Johnson de 8 bit que el nible bajo aparezca en el nible bajo del puerto a y el nible alto en el nible bajo del puerto B. Usar el TIMER0 para temporizar durante un segundo cada muestra.
Se usa el Timer0 como temporizador. Se ajusta el pre-escalador en 1/32. Si parte del valor 06h el desborde se produce cada 8000 ciclos de la señal CK. Contando 125 desbordes del TMR0 transcurre 1 segundos.



Siguiendo el esquema el programa fuente este quedaría:

```
;Ejercicio de contador Johnson con temporización mediante desbordes del TIMER0.
;se muestra cada nible en un puerto.
```

```
;Se cuentan 125 desborde del TMR0 comenzando en 0Ch y el divisor en 1/64.

            LIST        p=16F84A    ;Selecciono el pic a usar en la compilación
            __CONFIG    0x3FF1      ;Configuro el funcionamiento del micro
            __IDLOCS    0x005       ;Identifico el software

;Declaro los nombres de los SFR y sus bits.
;-------Declaro las variables a usar en el programa fuente-----

W           EQU     .0          ;Destino el registro de trabajo
F           EQU     .1          ;Destino el propio registro direccionado

;----- Register Files------------------------------------------------------
INDF        EQU     H'00'       ;Registro auxiliar para direcc. Indirecto por registro
TMR0        EQU     H'01'       ;Registro de buffer del TIMER0
PCL         EQU     H'02'       ;Parte baja del PC
STATUS      EQU     H'03'       ;Palabra de estado del programa
FSR         EQU     H'04'       ;Puntero de la memoria RAM para direcc. Indirecto
PORTA       EQU     H'05'       ;Buffer de los pines RA0-RA4
PORTB       EQU     H'06'       ;Buffer de los pines RB0-RB7
INTCON      EQU     H'0B'       ;Registro de control de interrupciones
OPCION      EQU     H'81'       ;Registro de configuración de periféricos
TRISA       EQU     H'85'       ;Registro de control de dirección del puerto A
TRISB       EQU     H'86'       ;Registro de control de dirección del puerto B

;----- STATUS Bits --------------------------------------------------------
IRP         EQU     .7          ;Bit selector bancos del FSR, no se usa en el PIC16F84
RP1         EQU     .6          ;Bit alto de selección de bancos RAM en direcc. directo
RP0         EQU     .5          ;Bit bajo de selección de banco RAM en direcc. directo
TO          EQU     .4          ;Bit de estado del micro
PD          EQU     .3          ;Bit de estado del micro
Z           EQU     .2          ;Bandera de resultado cero
DC          EQU     .1          ;Bandera de semiacarreo en sumas y restas
CY          EQU     .0          ;Bandera de acarreo en sumas y restas

;----- INTCON Bits --------------------------------------------------------
GIE         EQU     .7          ;Habilitación general de todas las interrupciones
EEIE        EQU     .6          ;Mascara de interp final del ciclo de escritura EEPROM
T0IE        EQU     .5          ;Mascara de interrupción por desborde del TIMER0
INTE        EQU     .4          ;Mascara de interrupción por evento en RB0
RBIE        EQU     .3          ;Mascara de interrupción por cambios en RB4-RB7
T0IF        EQU     .2          ;Bandera de aviso de desborde de TIMER0
INTF        EQU     .1          ;Bandera de aviso de evento en RB0
RBIF        EQU     .0          ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits --------------------------------------------------------
RBPU        EQU     .7          ;Habilitación de pull-up en el PB
INTA        EQU     .6          ;Selector de flanco activo de RB0
T0CS        EQU     .5          ;Selector de fuente de incremento del TIMER0
T0SE        EQU     .4          ;Selector de flanco activo de RA4
PSA         EQU     .3          ;Selector de uso del pre-escalador
PS2         EQU     .2          ;Bit de configuración del pre-escalador
PS1         EQU     .1          ;Bit de configuración del pre-escalador
PS0         EQU     .0          ;Bit de configuración del pre-escalador

;Declaro las  variables propias del ejercicio
VAR1        EQU         0x20        ;Variable de intercambio con el puerto B
K2          EQU         0x21
NN1         EQU         0x0C
```

```
NN2             EQU         0x10

;Comienzo el programa colocando el  programa en la posición ROM donde se buscara la primera
;instrucción al salir de reset
                ORG         0x000
COMIENZO
                goto        PRINCIPIO

                ORG         0x004
VECTOR_DE_INTERRUPCION

SUBRUTINAS
TEMPORIZAR
                movlw       NN2
                movwf       K2          ;Contador de desbordes del TMR0
LAZO1
                bcf         INTCON,T0IF     ;Reseteo el aviso de desborde
LAZO2
                btfss       INTCON,T0IF     ;Detecto el desborde de TMR0
                goto        LAZO2

                movlw       NN1
                movwf       TMR0        ;Recargo el TMR0

                decfsz      K2,F
                goto        LAZO1       ;Cuento los desbordes de TMR0

                return

ACTUALIZAR
                bcf         STATUS,CY
                btfss       VAR1,.7
                bsf         STATUS,CY
                rlf         VAR1,F
                return

MOSTRAR
                movf        VAR1,W
                movwf       PORTA
                swapf       VAR1,w
                movwf       PORTB
                return


;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-
                ORG         0x100
PRINCIPIO
;Lo primero es setear los puertos
                bsf         STATUS,RP0      ;Cambio al Banco 1
                movlw       0x00        ;Cargo valor para que el puerto B
                movwf       TRISB       ; sea salidas
                                            ;El puerto A no lo uso
;Seteo el funcionamiento del TMR0
                bcf         OPCION,T0CS     ;TMR0 incrementado por CK
                bcf         OPCION,PSA      ;Pre-escalador aplicado al TMR0
                bsf         OPCION,PS0
                bsf         OPCION,PS1
                bsf         OPCION,PS2      ;Seteo pre-escalador a 1/256
```

```
                bcf             STATUS,RP0      ;Cambio al Banco 0
                movlw           NN1
                movwf           TMR0

;Termino de setear los periféricos y comienza el programa propiamente dicho
INICIO
                movlw           0xf0
                movwf           VAR1            ;Inicializo variable de intercambio con el PB
LOOP
                call            MOSTRAR
                call            ACTUALIZAR
                call            TEMPORIZAR
                goto            LOOP
                                                ;El programa no tiene fin
                END                             ;Pero el archivo a compilar si
```

Interrupciones

Son señales externas a la CPU. Generan desvíos en la secuencia del programa principal hacia una subrutina llamada **subrutina de atención a interrupción**.
La señal externa se llama **petición de interrupción**. La ocurrencia de la misma puede activa un bit llamado **Bandera (FLAG) de solicitud de interrupción**.
Puede estar habilitada o no para ser atendida según el valor de la **máscara de interrupción**, representado por un bit. En caso de no estar habilitada pasa inadvertida.
Cuando se produce una petición de interrupción se guarda esta señal (por si desaparece) en la bandera de solicitud de interrupción.
Si esta habilitada por la mascara de interrupción se produce una atención a la interrupción.
-Sucesos automáticos ocurridos sin intervención del programa principal.

Evento exterior al CPU
Se activa la bandera de solicitud de interrupción
Si la mascara de interrupción habilita se produce una atención a la interrupción
-Lo cual implica:

Se termina de ejecutar la instrucción en curso
Se coloca el PC en la pila (STACK)
Se deshabilitan todas las interrupciones de igual o menor prioridad y jerarquía.
Se carga el PC con el vector de interrupción correspondiente a la bandera de solicitud.

El programador debe :

Setear la forma de respuesta a las interrupciones al comienzo del programa o durante el mismo.

Desarrollar las subrutinas de atención correspondiente a cada posible interrupción a partir de la posición de memoria indicada por el fabricante. Eso implica discriminar cual interrupción fue solicitada, si el micro no lo hace, y generar la respuesta adecuada.

Cuidar de no alterar ningún registro usado en el programa principal. Si es necesario guardar una copia en la pila u otro lado para poder restaurarlo antes de salir de la subrutina

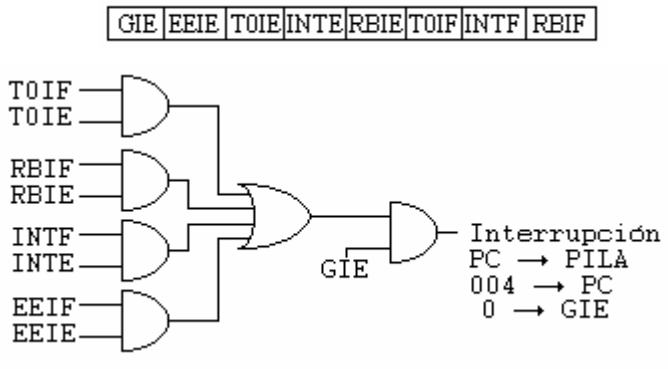
Al terminar la subrutina, e inmediatamente antes de retornar, resetear la Bandera de Solicitud de interrupción correspondiente a la interrupción atendida. De esa manera si hay alguna nueva solicitud pendiente podrá ser detectada sin problemas.

Retornar de la forma adecuada. (habilitando todas las interrupciones y restaurando los registros alterados que se deseen conservar intactos)

En este PIC hay 4 formas de interrumpir, cada una con su bandera y su mascara de interrupción.
Todos los bits son activos en 1.

Interrupcion	Bandera	Mascara	Habilitacion Gral
Desborde de Timer0	TOIF	TOIE	GIE
Cambio de NIvel de RB4-RB7	RBIF	RBIE	
Final de Ciclo de Escritura EEPROM	EEIF	EEIE	
Flanco de RB0/INT	INTE	INTF	

Se puede hacer un esquema del funcionamiento de las interrupciones, donde se observa la dependencia de los avisos de interrupción con sus respectivas mascaras y la habilitación general GIE.



T0IF es el aviso de desborde del TIMER0 cuando se incrementa de FFh a 00h. Su mascara es T0IE, que lo habilita cuando vale 1.
INTF es el aviso de un flanco valido en RB0, si este esta configurado como entrada. Mediante INTE igual a 1 se habilita. Con el bit OPTION,INTA se elige el flanco válido de RB0 para la interrupción INT. Si vale 1 se activa con el flanco de subida , y si vale 0 se activa con el flanco descendente.
RBIF es el aviso de algún cambio en el valor de pines RB4-RB7, que estén configurados como entradas. No discrimina cual es el pin que cambio su valor. Se habilita a interrumpir con el bit RBIE igual a 1. A todo el puerto B (RB0-RB7) se le puede activar un símil de resistencias pull-up de aproximadamente 10kohm. Solo son activadas en los pines configurados como entradas y se mantienen aun en modo de bajo consumo (atención si se activan y se quiere economizar energía).
Al entrar en la interrupción se debe determinar cual es la causa de la misma. En el caso de no habilitar ninguna otra fuente es un paso innecesario. La determinación de cual es la fuente se hace verificando las banderas de aviso de interrupción. En el caso de desborde del TMR0 es verificando el bit T0IF del registro INTCON.
Como la interrupción no es predecible cuando ocurre o no interesa, se debe prever no alterar el programa principal. Los registros más importantes para el mantenimiento normal del programa son W, STATUS y FSR. Los guardamos en registros auxiliares, debido a que este micro no cuenta con una pila de uso general. En la pila solo se guarda el PC.
El fabricante recomienda este código.

```
MOVWF   W_TEMP          ;copy W to temp register,
                        ;could be in either bank
SWAPF   STATUS,W        ;swap status to be saved into W
MOVWF   STATUS_TEMP     ;save status to bank 0 register
```

Previo a salir de la subrutina restauramos el valor de estos registros

```
SWAPF   STATUS_TEMP,W   ;swap STATUS_TEMP register
                        ;into W, sets bank to original state
MOVWF   STATUS          ;move W into STATUS register
SWAPF   W_TEMP,F        ;swap W_TEMP
SWAPF   W_TEMP,W        ;swap W_TEMP into W
```

Modo Bajo Consumo - SLEEP (Power Down Mode)

El modo de bajo consumo, SLEEP o Power Down se entra mediante software.
La instrucción

```
sleep           ;Entra al micro en modo bajo consumo.
```

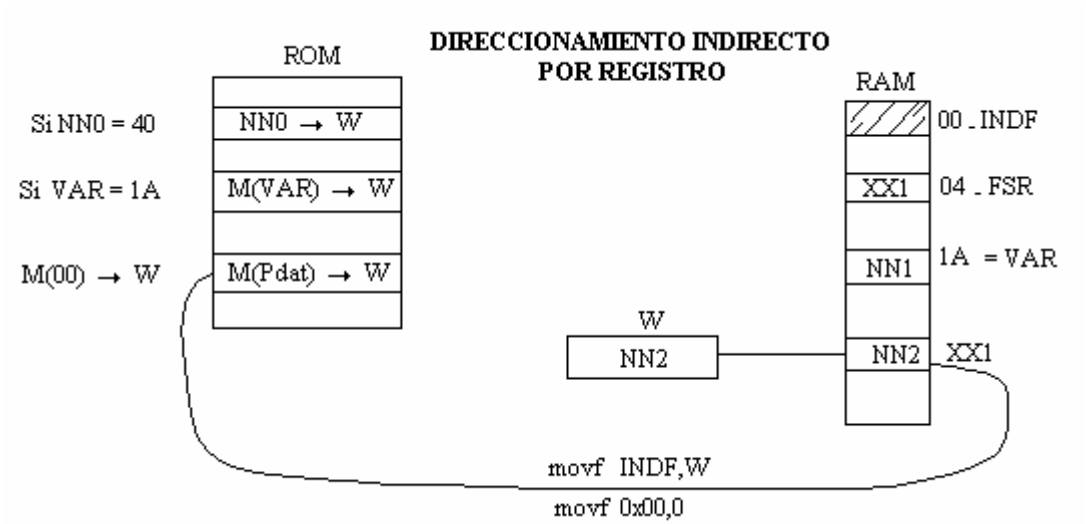
El bit PD (STATUS,3) es puesto a cero y el bit TO (STATUS,4 es puesto a uno. Se detiene el modulo del oscilador y del TIMER0. El modulo WDT si esta habilitado es reseteado pero continua funcionado.
El consumo de corriente baja a menos de 0,5 mA.
Los pines de los puertos seteados como salida mantienen el último valor escrito (1, 0 o alta impedancia). A los pines setados como entradas es aconsejable no dejarlos flotantes sino puestos a algún nivel definido (alto o bajo). Las resistencias de pull-up deben considerarse por su consumo de corriente si se deja el pin a nivel bajo.

Solo hay tres formas de sacar al micro de ese estado, despertar (Wake Up).

Por Flanco en RB0/INT
Por Cambio de Nivel de RB4-RB7
Por Finalizacion del ciclo de escritura en EEPROM
Por Reset General (MCLRST)
Por desborde del Vigilante (WDT). Genera un reset general interno, pero no pone a cero el pin MCRST

EL Timer0 no genera interrupciones debido a que se detiene el oscilador principal.
Al ejecutarse la instrucción SLEEP se produce la búsqueda (FETCH) de la siguiente instrucción.
Las interrupciones INTF , EEIF y RBIF despiertan al micro si estan habilitadas mediante INTE , EEIE y RBIE.
En cuanto se despierta el micro ejecuta la instrucción de la posición siguiente a la instrucción SLEEP, a la cual había hecho su búsqueda antes de entrar en modo bajo consumo. Es aconsejable después de la instrucción SLEEP colocar una instrucción NOP.
Si el bit GIE vale 1 antes de entrar en modo de bajo consumo, al despertar por una interrupción salta a la posición 004h, después de ejecutar la instrucción inmediata a SLEEP.
Si GIE vale 0 antes de SLEEP, continua con las instrucciones si saltar a 004h.
Si la interrupción ocurre durante la ejecución de la instrucción SLEEP , primero se entra completamente en modo de bajo consumo y después se despierta.
El despertar dura 1024 ciclos de la frecuencia del oscilador. Para un cristal de 4 Mhz reporesenta 256 microsegundos.
.

DIRECIONAMIENTO INDIRECTO POR REGISTRO



-La instrucción indica, no la posición donde se guarda el dato sino donde se guarda el valor de la posición de memoria. Se invoca el uso de un puntero el cual almacena la posición del dato a usar. Se llama direccionamiento indirecto por registro.
Este micro tiene un solo puntero (FSR) para la memoria RAM, pero no tiene instrucciones específicas para usarlo. Cualquier instrucción que direcciona en forma directa la posición cero de RAM o sea el registro INDF, es interpretada como que utiliza al puntero FSR. El registro INDF no tiene implementación física en el hardware.
La nomenclatura seria
M(Pdat) → W o M(FSR) → W
Se debe asociar este puntero lógico Pdat al único puntero físico FSR.
La instrucción o mnemónico es

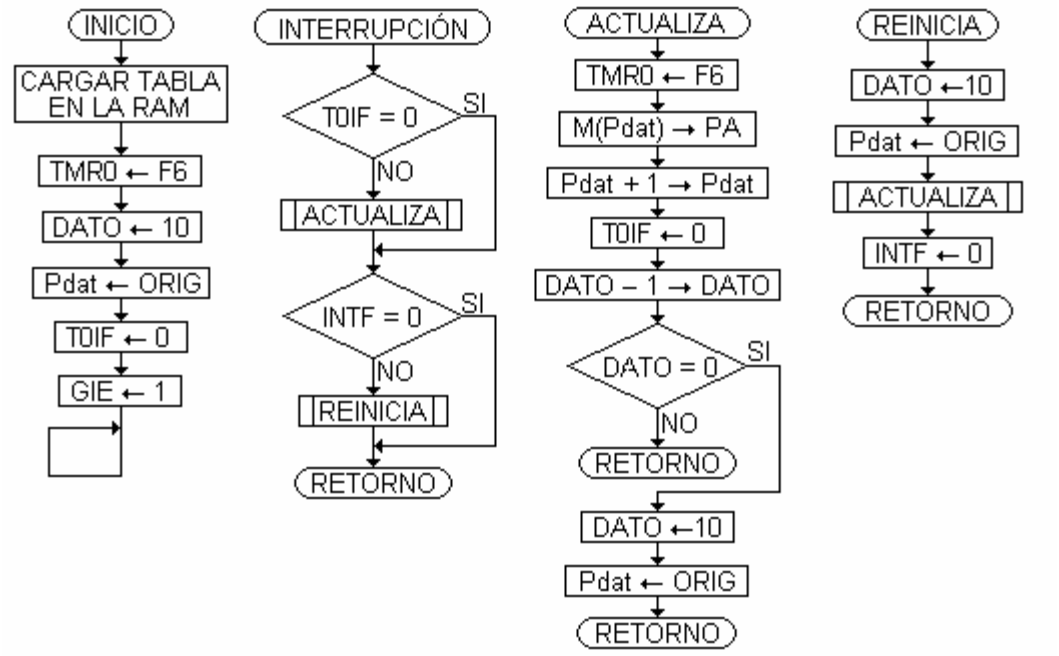
```
movf    0x00,.0
movf    INDF,W
```

Observar que en el mnemónico no se nombra al FSR y se usa la misma instrucción que en el direccionamiento directo. La diferencia esta en el argumento, el cual es 00 o INDF. Estas instrucciones no alteran el valor del FSR.
La existencia de más de un banco implica una dificultad si se desea trabajar con registros en ambos bancos. El FSR tiene la capacidad de con sus 8 bit seleccionar completamente un registro en

particular de un banco determinado. El bit 7 del FSR sirve para determinar en que banco se moverá. Si vale 0 se mueve sobre el banco o y si vale 1 trabaja con el banco 1. Pero como el pic16f84 tiene implementado el banco 1 sobre el banco 0 si direccionamos a 9Eh en realidad se trabaja con la posición 1Eh.

Ejercicio:
Realizar un contador Grey de 4 bit en el nible bajo del puerto A, que avanza cada 10 pulsos detectados en RA4. Cuando reciba una señal INT el contador volverá al valor inicial. El pulso en nuestros módulos son activos en L (nivel 0).

En este ejercicio usamos la interrupción del desborde del timer0 y la petición de interrupción INT. Se involucra a las Banderas de Solicitud de Interrupción T0IF y INTF, a las máscaras de interrupción T0IE y INTF, y a la habilitación general de interrupciones GIE. El final del programa principal es un bucle infinito, no es necesario hacer nada más. Del resto se encarga la interrupción.
Este lazo es para no hacer nada más, aunque el programa fuente tiene un final, el micro debe seguir funcionando y esperando el desborde del TMR0.




```
;Ejercico detectar 10 pulsois por RA4 y variar un generador Grey de 4 bit en el puerto A (RA0-RA3)
;Controlado con pin RB0
                list            p=16f84a
                __config        0x3FF1
                __idlocs        0x0005

;-------Declaro las variables a usar en el programa fuente-----

W               EQU     .0          ;Destino el registro de trabajo
F               EQU     .1          ;Destino el propio registro direccionado

;----- Register Files----------------------------------------------------
INDF            EQU     H'00'       ;Registro auxiliar para direcc. Indirecto por registro
TMR0            EQU     H'01'       ;Registro de buffer del TIMER0
PCL             EQU     H'02'       ;Parte baja del PC
STATUS          EQU     H'03'       ;Palabra de estado del programa
FSR             EQU     H'04'       ;Puntero de la memoria RAM para direcc. Indirecto
PORTA           EQU     H'05'       ;Buffer de los pines RA0-RA4
PORTB           EQU     H'06'       ;Buffer de los pines RB0-RB7
INTCON          EQU     H'0B'       ;Registro de control de interrupciones
OPCION          EQU     H'81'       ;Registro de configuración de periféricos
```

```
TRISA           EQU     H'85'       ;Registro de control de dirección del puerto A
TRISB           EQU     H'86'       ;Registro de control de dirección del puerto B

;----- STATUS Bits --------------------------------------------------------
IRP             EQU     .7          ;Bit de selección bancos para el FSR
RP1             EQU     .6          ;Bit alto de selección de bancos RAM en direcc. directo
RP0             EQU     .5          ;Bit bajo de selección de banco RAM en direcc. directo
TO              EQU     .4          ;Bit de estado del micro
PD              EQU     .3          ;Bit de estado del micro
Z               EQU     .2          ;Bandera de resultado cero
DC              EQU     .1          ;Bandera de semiacarreo en sumas y restas
CY              EQU     .0          ;Bandera de acarreo en sumas y restas

;----- INTCON Bits --------------------------------------------------------
GIE             EQU     .7          ;Habilitación general de todas las interrupciones
EEIE            EQU     .6          ;Mascara de interrup final del ciclo de escritura EEPROM
T0IE            EQU     .5          ;Mascara de interrupción por desborde del TIMER0
INTE            EQU     .4          ;Mascara de interrupción por evento en RB0
RBIE            EQU     .3          ;Mascara de interrupción por cambios en RB4-RB7
T0IF            EQU     .2          ;Bandera de aviso de desborde de TIMER0
INTF            EQU     .1          ;Bandera de aviso de evento en RB0
RBIF            EQU     .0          ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits --------------------------------------------------------
RBPU            EQU     .7          ;Habilitación de pull-up en el PB
INTA            EQU     .6          ;Selector de flanco activo de RB0
T0CS            EQU     .5          ;Selector de fuente de incremento del TIMER0
T0SE            EQU     .4          ;Selector de flanco activo de RA4
PSA             EQU     .3          ;Selector de uso del pre-escalador
PS2             EQU     .2          ;Bit de configuración del pre-escalador
PS1             EQU     .1          ;Bit de configuración del pre-escalador
PS0             EQU     .0          ;Bit de configuración del pre-escalador

;Declaro mis variables propias
AUX_W           EQU     0x10
AUX_ST          EQU     0x11
DATO            EQU     0x12
PDAT            EQU     FSR
ORIG            EQU     0x20

                ORG         0x0000
RESET
                goto        PRINCIPIO

                ORG         0x0004
INTERRUPCION
;Primero guardo los registros W y STATUS
                movwf       AUX_W
                swapf       STATUS,W
                movwf       AUX_ST
;Verifico si la interrupcion la realizop el Timer0
                btfsc       INTCON,T0IF
                call        ACTUALIZAR
;Verifico si la interrupcion es por reinicio de RB0/INT
                btfsc       INTCON,INTF
                call        REINICIA
;Antes de irme recupero los registros STATUS y W
                swapf       AUX_ST,W
                movwf       STATUS
```

```
                swapf       AUX_W,F
                swapf       AUX_W,W
                retfie                  ;Retorno de las interrupciones

;*********
ACTUALIZAR
                movlw       0xF6
                movwf       TMR0

                movf        INDF,W
                movwf       PORTA

                incf        PDAT,F

                bcf         INTCON,T0IF

                decfsz      DATO,F
                return

                movlw       0x10        ;Después de 16 símbolos vuelvo a empezar
                movwf       DATO

                movlw       ORIG
                movwf       PDAT

                return

;***************
REINICIA
                movlw       0x10
                movwf       DATO

                movlw       ORIG
                movwf       PDAT

                call        ACTUALIZAR      ;Si no actualizo no se nota que reinicie

                bcf         INTCON,INTF
                return

;.-.----.-..--.-.-.-.-.----....-.-.-.-.-.-.-.-.-.-.-.-.-.-.
;.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
                ORG         0x0100
PRINCIPIO
;Primero seteamos periféricos - puertos
                bsf             STATUS,RP0      ;Cambio de Banco
                movlw           b'11110000'
                movwf           TRISA
                movlw           b'00000001'
                movwf           TRISB
;Seteamos periféricos  - timer0
                bsf         OPCION,T0CS     ;Fuente del TIMER0 es RA4
                bsf         OPCION,PSA      ;Sin pre-escalador
;Seteo interrupciones
                bsf         INTCON,T0IE     ;Habilito la interrupción por TIMER0
                bsf         INTCON,INTE     ;Habilito la interrupción por RB0/INT
;Volvemos al banco 0
                bcf         STATUS,RP0      ;Cambio al banco 0
```

```
INICIO
;Cargar la tabla es lo primero
TABLA
                movlw     b'00000000'     ;Cargo los valores en la RAM
                movwf     0x20
                movlw     b'00000001'     ;Así los traigo con el puntero
                movwf     0x21
                movlw     b'00000011'
                movwf     0x22
                movlw     b'00000010'
                movwf     0x23
                movlw     b'00000110'
                movwf     0x24
                movlw     b'00000111'
                movwf     0x25
                movlw     b'00000101'
                movwf     0x26
                movlw     b'00000100'
                movwf     0x27
                movlw     b'00001100'
                movwf     0x28
                movlw     b'00001101'
                movwf     0x29
                movlw     b'00001111'
                movwf     0x2A
                movlw     b'00001110'
                movwf     0x2B
                movlw     b'00001010'
                movwf     0x2C
                movlw     b'00001011'
                movwf     0x2D
                movlw     b'00001001'
                movwf     0x2E
                movlw     b'00001000'
                movwf     0x2f

;Seteo inicio del TIMER0 y otros valores
                movlw     0xF6
                movwf     TMR0

                movlw     0x10
                movwf     DATO

                movlw     ORIG
                movwf     PDAT

                bcf       INTCON,T0IF
                bsf       INTCON,GIE
LAZO

                GOTO      LAZO


                end
```

DIRECCIONAMIENTO INDEXADO

Este micro solo posee un puntero para memoria RAM, y no aceptar cargar datos en la memoria de programa. Pero se puede usar el contador de programa PC para obtener datos guardados en la ROM de programa. El PC siempre guarda la dirección de la próxima instrucción a ejecutar.
*El PC se pude alterar a través de 2 registros, PCL y PCLATH
PCL - Registro de lecto/escritura copia los bit 0 a bit 7 del PC.
PCLATH - Registro de solo escritura permite alterar del bit 8 al bit 12.
Cualquier operación sobre PCL altera al PC, o sea a la secuencia del programa, y vuelca PCLATH en los bit 8 a bit 13 del PC. Este salto es un GOTO COMPUTADO.
Un desborde o borrow en PCL no afecta a los bits altos del PC.

**La instrucción addwf VAR,F suma W al registro VAR y guarda el resultado en VAR.*
Si esta instrucción se aplica al PCL, se altera el PC según el resultado de esta instrucción del valor cargado en PCLATH. Lo cargado en PCLATH no se afecta a menos que sea escrito, pues no lee al PC sino que lo carga solamente.
Si W = 3F , PCLATH = 02 y PC=0110 por lo tanto PCL = 10

```
addwf   PCL,F
```

Da como resultado PCL = 4F entonces PC=024F
Se produce un salto hacia delante en el programa
Si W = 1A , PCLATH=01 y PC = 1F3 entonces PCL = F3

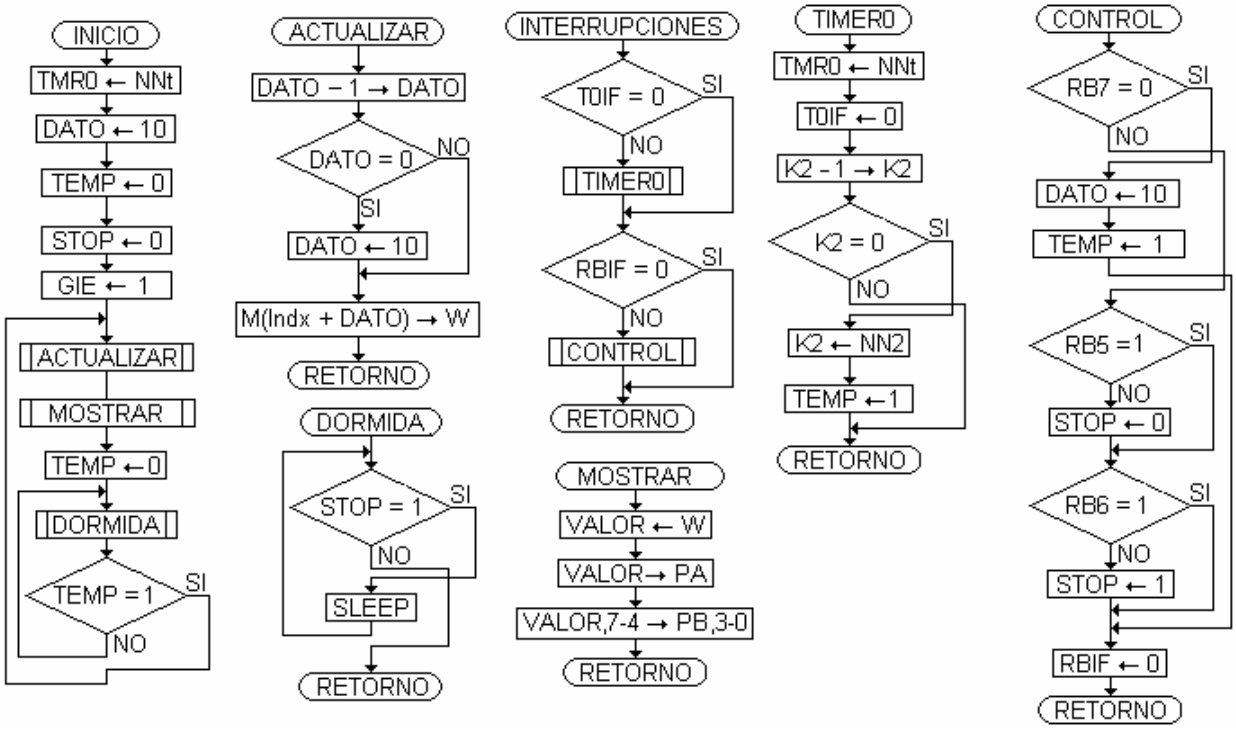
```
addwf   PCL,F
```

Da como resultado PCL = 0D entonces PC = 010D
Se produce un salto hacia atrás en el programa.

*La instrucción *retfie k* produce el retorno de una subrutina con el valor literal *k* cargado en W.

```
retfie      k       ; Pila → PC y K → W
```

Ejercicio:
Realizar un contador Johnson de 8 bit que el nible bajo aparezca en el nible bajo del puerto a y el nible alto en el nible bajo del puerto B. Cada elemento se mostrara durante 2 segundos. Mediante botones en el nible alto del puerto B se controlara su funcionamiento. Utilizar interrupción del Timer0 y del RB4-RB7.
RB7 reinicia el contador
RB6 detiene el contador y pone a dormir el chip
RB5 despierta el contador.



Se ajusta el pre-escalador en 1/64. El TMR0 parte del valor 06h, con lo cual el desborde se produce cada 16000 ciclos de la señal CK. Contando 125 desbordes del TMR0 transcurre 2 segundo.
En este programa se utiliza una palabra de estado propia del programador para que distintas partes (subrutinas) del programa se comuniquen que se esta haciendo.
Los bits STOP y TEMP nos indican si se esta escribiendo , despertando poniendo a dormir el micro.

```
;Ejercicio de Contador johnson de 8 salidas con teclas de control
; por interrupcion en el puerto B y temporizacion con interrupcion de TIMER0

                list            p=16f84a
                __config        0x3FF1
                __idlocs        0x0007

;-------Declaro las variables a usar en el programa fuente-----

W               EQU     .0              ;Destino el registro de trabajo
F               EQU     .1              ;Destino el propio registro direccionado

;----- Register Files------------------------------------------------------
INDF            EQU     H'00'           ;Registro auxiliar para direcc. Indirecto por registro
TMR0            EQU     H'01'           ;Registro de buffer del TIMER0
PCL             EQU     H'02'           ;Parte baja del PC
STATUS          EQU     H'03'           ;Palabra de estado del programa
FSR             EQU     H'04'           ;Puntero de la memoria RAM para direcc. Indirecto
PORTA           EQU     H'05'           ;Buffer de los pines RA0-RA4
PORTB           EQU     H'06'           ;Buffer de los pines RB0-RB7
INTCON          EQU     H'0B'           ;Registro de control de interrupciones
OPCION          EQU     H'81'           ;Registro de configuración de periféricos
TRISA           EQU     H'85'           ;Registro de control de dirección del puerto A
TRISB           EQU     H'86'           ;Registro de control de dirección del puerto B

;----- STATUS Bits --------------------------------------------------------
IRP             EQU     .7              ;Bit de selección bancos para el FSR
RP1             EQU     .6              ;Bit alto de selección de bancos RAM en direcc. directo
RP0             EQU     .5              ;Bit bajo de selección de banco RAM en direcc. directo
TO              EQU     .4              ;Bit de estado del micro
PD              EQU     .3              ;Bit de estado del micro
Z               EQU     .2              ;Bandera de resultado cero
DC              EQU     .1              ;Bandera de semiacarreo en sumas y restas
CY              EQU     .0              ;Bandera de acarreo en sumas y restas

;----- INTCON Bits --------------------------------------------------------
GIE             EQU     .7              ;Habilitación general de todas las interrupciones
EEIE            EQU     .6              ;Mascara de interrup final del ciclo de escritura EEPROM
T0IE            EQU     .5              ;Mascara de interrupción por desborde del TIMER0
INTE            EQU     .4              ;Mascara de interrupción por evento en RB0
RBIE            EQU     .3              ;Mascara de interrupción por cambios en RB4-RB7
T0IF            EQU     .2              ;Bandera de aviso de desborde de TIMER0
INTF            EQU     .1              ;Bandera de aviso de evento en RB0
RBIF            EQU     .0              ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits --------------------------------------------------------
RBPU            EQU     .7              ;Habilitación de pull-up en el PB
INTA            EQU     .6              ;Selector de flanco activo de RB0
T0CS            EQU     .5              ;Selector de fuente de incremento del TIMER0
T0SE            EQU     .4              ;Selector de flanco activo de RA4
PSA             EQU     .3              ;Selector de uso del pre-escalador
PS2             EQU     .2              ;Bit de configuración del pre-escalador
PS1             EQU     .1              ;Bit de configuración del pre-escalador
PS0             EQU     .0              ;Bit de configuración del pre-escalador


;Variables propias
```

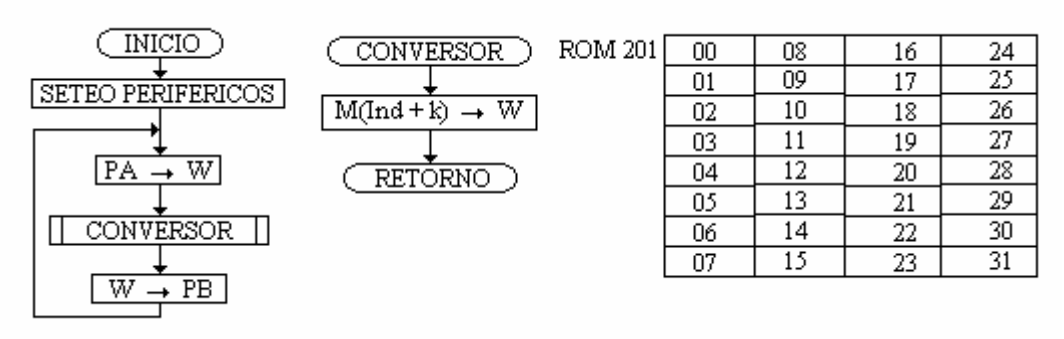
```
W_AUX       EQU   0x13
ST_AUX      EQU   0x12
DATO        EQU   0x11
VALOR       EQU   0x10
RB7         EQU   .7
RB6         EQU   .6
RB5         EQU   .5
ESTADO      EQU   0x16
STOP        EQU   .4
TEMP        EQU   .2
K2          EQU   0x15
NNt         EQU   0x06
NN2         EQU   0x7D

;##########   comienzan las instrucciones   ###
            ORG         0x0000
RESET
            goto        PRINCIPIO

;-------------------------------------------------------------------------------
;_____________________________________________________________________


            ORG         0x0004
;VECTOR DE INTERRUPCIÓN
INTERRUPCIONES
            movwf       W_AUX       ;Salvo el valor de W
            movf        STATUS,W    ;Salvo el valor del STATUS
            movwf       ST_AUX
;Termine el ingreso a las interrupciones

            btfsc       INTCON,T0IF ;Verifico si la interrupcion
                                    ;es por desborde del TMR0
            call        TIMER0
            btfsc       INTCON,RBIF ;Verifico si la interrupcion
                                    ;es por cambio en el puerto b
            call        CONTROL             ;Llamo a la rutina de atencion
                                    ; de las teclas
IRNOS
;Debo volver con recuperacion de los datos de W y STATUS
            swapf       ST_AUX,w    ;Lo complicado es recuperar
            movwf       STATUS              ;STATUS intacto, pues algunos
            swapf       W_AUX,F             ;movimientos alteran Z
            swapf       W_AUX,W

            retfie                  ;Retorno de interrupciones
                                    ;y las habilito

;!!!!!!  Subrutina de atencion a los desborde del TIMER0  !!!!!!
TIMER0
            movlw       NNt
            movwf       TMR0        ;Reinicio el contador TMR0
            bcf         INTCON,T0IF ;Bajo la bandera de interrupcion

            decfsz      K2,F
            goto        CUENTA
            movlw       NN2
            movwf       K2          ;Recargo el valor de K2

            bsf         ESTADO,TEMP;Marco la finalizacion de 1 seg.
CUENTA
            return                  ;Retorno al programa en la
```

```
                                        ;seccion de interrupciones

;!!!!!  Subrutina de atencion a las teclas en RB7-RB5  !!!!!!!!!!
CONTROL
                btfsc           PORTB,RB7     ;Verifico si RB7 esta activo
                goto            SEGUIR
RESETEAR
                movlw           0x10
                movwf           DATO          ;Reinicio el contador
                bsf             ESTADO,TEMP;Activo aviso de escritura
                goto            VUELTA
SEGUIR
                btfss           PORTB,RB5     ;Verifico si RB5 esta activo
                bcf             ESTADO,STOP
PARAR
                btfss           PORTB,RB6     ;Verifico si RB6 esta activo
                bsf             ESTADO,STOP ;Por ser la ultima verificación tiene prioridad sobre
RB5

VUELTA
                bcf             INTCON,RBIF   ;Limpio la bandera de interrupción por puerto B
                return                        ;Termine de verificar teclas
;_____________________________________________________________________________________
____
;!!!!! Terminaron las subrutinas de atención a interrupciones  !!!!!!
;-------------------------------------------------------------------------------

;Subrutina propias
ACTUALIZAR                              ;Se encarga de ajustar el
                                        ;próximo símbolo a exhibir
                decfsz          DATO,F                ;Actualizo el valor del orden el valor a
mostrar
                goto            BUSCAR_VALOR
                movlw           0x10
                movwf           DATO          ;Reinicio el valor de DATO
BUSCAR_VALOR
                movf            DATO,W
                addwf           PCL,F
                nop
                retlw           b'00000000'
                retlw           b'00000001'
                retlw           b'00000011'
                retlw           b'00000111'
                retlw           b'00001111'
                retlw           b'00011111'
                retlw           b'00111111'
                retlw           b'01111111'
                retlw           b'11111111'
                retlw           b'11111110'
                retlw           b'11111100'
                retlw           b'11111000'
                retlw           b'11110000'
                retlw           b'11100000'
                retlw           b'11000000'
                retlw           b'10000000'

DORMIDA                                         ;Pone a dormir el micro
                                        ;hasta nuevo aviso por RB5
                btfss           ESTADO,STOP
                goto            DESPERTAR
                sleep
                nop                           ;La línea nop es para asegurarme que
```

```
                                        ;atiende la interrupción antes del salto
          goto          DORMIDA
DESPERTAR
          return


MOSTRAR
          movwf         VALOR           ;El noble bajo aparece en el PA
          movwf         PORTA
          swapf         VALOR,W
          movwf         PORTB           ;EL nible alto en RB0-RB3
          return

;________________________________________________________________________________
_
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

          ORG           0x0100
PRINCIPIO

;Seteo de perifericos-  PUERTOS
          bsf           STATUS,RP0      ;Cambio al banco 1 para setear los perifericos
          movlw         b'00000000'
          movwf         TRISA
          movlw         b'11110000'
          movwf         TRISB

;Seteo TIMER0
          bcf           OPCION,PSA
          bcf           OPCION,T0SE
          bsf           OPCION,PS2
          bsf           OPCION,PS1
          bcf           OPCION,PS0

;Seteo Interupciones
          bcf           INTCON,RBIF
          bcf           INTCON,T0IF
          bSf           INTCON,RBIE
          bsf           INTCON,T0IE

          BCF           STATUS,RP0      ;cAMBIO AL BANCO 0



;Fin del seteo de perifericos
;*************************************************************************************
;

INICIO
          movlw         NNt
          movwf         TMR0            ;Cargo valor inicial para el TIMER0
          movlw         0x10
          movwf         DATO                    ;CARGO el primer elemento del contador
          bcf           ESTADO,STOP             ;Seteo para arrancar sin detenerme
          bcf           ESTADO,TEMP             ;Seteo indicador de un segundo
          bsf           INTCON,GIE              ;Habilito las interrupciones
LOOP
          call          ACTUALIZAR      ;LLamo a la subrutina que cambia de elemento
          call          MOSTRAR                 ;Muestro el elemento en los puertos

          bcf           ESTADO,TEMP;Bajo la bandera de 1 segundo
ESPERA
          call          DORMIDA                 ;Paso a verificar si debo detener el micro
```

```
btfss       ESTADO,TEMP
goto        ESPERA

goto        LOOP

END
```

Ejercicio:
Realizar un conversor binario a BCD. Por los 5 pin del puerto A entra un valor binario el cual aparece codificado en BCD por el puerto B. Utilizar la obtención de datos mediante direccionamiento indexado.



00	08	16	24
01	09	17	25
02	10	18	26
03	11	19	27
04	12	20	28
05	13	21	29
06	14	22	30
07	15	23	31

El uso del direccionamiento indirecto supone el uso de una tabla. En este micro en particular solo puede ser una tabla en la memoria RAM y su puntero el FSR. Se dedicara una rutina a cargar la tabla en la memoria RAM antes de comenzar el programa.

```
;Conversor de Binario a BCD del puertoa al puerto B condireccionamiento indexado

; primero seteamos Palabra de Configuración
                LIST            p=16f84         ;Indico compilar para el PIC16F84A
                __CONFIG        0x3FF1          ;Selecciono XT, PWTE, No WDT  y No CP
                __IDLOCS        0x0008          ;Identifico el software

;Declaro los nombres de todos los SFR (registros de uso especial)
;Como el micro trata a los puertos como registros  internos se le puede aplicar cualquier
; operación valida para los SFR
;-------Declaro las variables a usar en el programa fuente-----

W               EQU     .0              ;Destino el registro de trabajo
F               EQU     .1              ;Destino el propio registro direccionado

;----- Register Files----------------------------------------------------------
INDF            EQU     H'00'           ;Registro auxiliar para direcc. Indirecto por registro
TMR0            EQU     H'01'           ;Registro de buffer del TIMER0
PCL             EQU     H'02'           ;Parte baja del PC
STATUS          EQU     H'03'           ;Palabra de estado del programa
FSR             EQU     H'04'           ;Puntero de la memoria RAM para direcc. Indirecto
PORTA           EQU     H'05'           ;Buffer de los pines RA0-RA4
PORTB           EQU     H'06'           ;Buffer de los pines RB0-RB7
INTCON          EQU     H'0B'           ;Registro de control de interrupciones
OPCION          EQU     H'81'           ;Registro de configuración de periféricos
TRISA           EQU     H'85'           ;Registro de control de dirección del puerto A
TRISB           EQU     H'86'           ;Registro de control de dirección del puerto B

;----- STATUS Bits --------------------------------------------------------------
IRP             EQU     .7              ;Bit selector bancos del FSR, no se usa en el PIC16F84
RP1             EQU     .6              ;Bit alto de selección de bancos RAM en direcc. directo
RP0             EQU     .5              ;Bit bajo de selección de banco RAM en direcc. directo
TO              EQU     .4              ;Bit de estado del micro
PD              EQU     .3              ;Bit de estado del micro
Z               EQU     .2              ;Bandera de resultado cero
DC              EQU     .1              ;Bandera de semiacarreo en sumas y restas
CY              EQU     .0              ;Bandera de acarreo en sumas y restas

;----- INTCON Bits --------------------------------------------------------------
GIE             EQU     .7              ;Habilitación general de todas las interrupciones
```

```
EEIE            EQU     .6              ;Mascara de interp. final del ciclo de escritura EEPROM
T0IE            EQU     .5              ;Mascara de interrupción por desborde del TIMER0
INTE            EQU     .4              ;Mascara de interrupción por evento en RB0
RBIE            EQU     .3              ;Mascara de interrupción por cambios en RB4-RB7
T0IF            EQU     .2              ;Bandera de aviso de desborde de TIMER0
INTF            EQU     .1              ;Bandera de aviso de evento en RB0
RBIF            EQU     .0              ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits ---------------------------------------------------
RBPU            EQU     .7              ;Habilitación de pull-up en el PB
INTA            EQU     .6              ;Selector de flanco activo de RB0
TOCS            EQU     .5              ;Selector de fuente de incremento del TIMER0
TOSE            EQU     .4              ;Selector de flanco activo de RA4
PSA             EQU     .3              ;Selector de uso del pre-escalador
PS2             EQU     .2              ;Bit de configuración del pre-escalador
PS1             EQU     .1              ;Bit de configuración del pre-escalador
PS0             EQU     .0              ;Bit de configuración del pre-escalador


;------------ Declaro todas las constantes, etiquetas y variables a usar--------------
;No se usa variables propias
;Comienzo el programa colocando el  programa en la posición ROM donde se buscara la primera
;instrucción al salir de reset
                ORG         0x000
COMIENZO
                goto        PRINCIPIO

                ORG         0x004
VECTOR_DE_INTERRUPCION

SUBRUTINAS
CONVERSOR
                goto            BINBCD          ;Salto a la pagina 02 de la ROM


;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
                ORG         0x100
PRINCIPIO
;Lo primero es setear los puertos
                bsf             STATUS,RP0      ;Cambio al Banco 1
                movlw           0x00            ;Cargo valor para que el puerto B
                movwf           TRISB           ; sea salidas
                movlw           0xFF            ;Cargo valor para que el PA sea todo entradas
                movwf           TRISA

                bcf             STATUS,RP0      ;Cambio al Banco 0
;Seteo el valor de PCLATH para ajustar el valor de la suma con PCL
                movlw           0x02
                movwf           PCLATH          ;02h es la pagina en que esta la subrutina

;-*-*-*-*-*-*-*-*-*-*--*-*-*-*-*-*-*-*-*-*-*---*-*-*-*-*-*-*-*-*-*-*---*-*-*-*-*-*-*-*-*-
;Termino de setear los periféricos y comienza el programa propiamente dicho

INICIO

                movf            PORTA,W         ;Leo el binario de 5 dígitos del puerto A
                call            CONVERSOR       ;Cambio de binario a BCD
                movwf           PORTB
```

```
                goto            INICIO          ;El programa no tiene fin
;El programa principal termino pero después de él se coloca una subrutina que viene
;de la página 00

                ORG             0x200
BINBCD
                addwf           PCL,F
                retlw           0x00
                retlw           0x01
                retlw           0x02
                retlw           0x03
                retlw           0x04
                retlw           0x05
                retlw           0x06
                retlw           0x07
                retlw           0x08
                retlw           0x09
                retlw           0x10
                retlw           0x11
                retlw           0x12
                retlw           0x13
                retlw           0x14
                retlw           0x15
                retlw           0x16
                retlw           0x17
                retlw           0x18
                retlw           0x19
                retlw           0x20
                retlw           0x21
                retlw           0x22
                retlw           0x23
                retlw           0x24
                retlw           0x25
                retlw           0x26
                retlw           0x27
                retlw           0x28
                retlw           0x29
                retlw           0x30
                retlw           0x31
                retlw           0x32

;Fin  de la tabla ROM
                END
```

CONTROL DE MECANISMOS

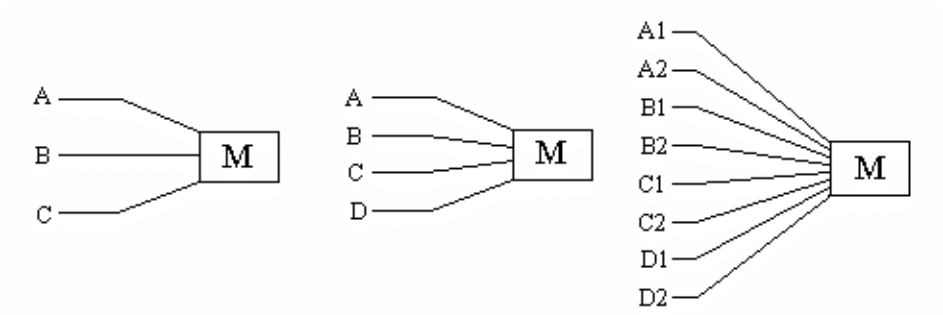
Control de Motores Paso a Paso

Los motores paso a paso son motores de movimiento discreto. Necesitan un control logico para alimentar sus bobinas. Los microcontroladores permiten hacerlo con mucha exactitud y flexibilidad al momento de modificar la secuencia.

Los parámetros mas importantes de estos motores son:

Angulo de paso. Variación angular mínima al cambiar el valor de entrada.
Torque de paso. Fuerza que desarrolla para girar el eje.
Fuerza de frenado. Oposición a ser sacado de una posición estable.
Velocidad máxima. La mayor rapidez de cambio de posición sin perder el paso.

Se pueden controlar dentro de cierto límite la velocidad y aceleración angular. Se fabrican con 3, 4 u 8 hilos de control.



Los motores de 4 hilos son similares a los motores de 8 hilos. En estos últimos, si sus bobinas se conectan de a dos en anti-paralelo, se controla como de 4 hilos. En general pueden quedar estáticos. Es decir mantener velocidad cero. E inclusive se les define un torque necesario para cambiar de su posición angular.

Los motores de tres líneas, tienen un valor máximo para la duración del pulso a aplicar. No aceptan velocidad cero, debido a que sus bobinas son de baja resistencia óhmica. Se usan en situaciones donde no es necesario controlar posición angular, solo velocidad y aceleración angular.

Todos tienen en común que usan en sistemas digitales para controlarlos.

Motor de 4 bit

Consideremos el caso de un motor de 4 hilos de control. Se le van dando valores que activen en orden las bobinas del motor. Hecho en la secuencia correcta el motor va dando pasos de **n** grados (un paso) o de **n/2** grados (medio paso) en cada cambio. Hoy en día la tecnología permite hacer n = 7,5.

Consideremos el caso del sistema con cuatro hilos de control. Podemos tener 2 secuencia según se desee mover de a paso o de a medio paso.

Se le van dando estados al sistema. Al cambiar de un estado a otro podemos hacerlo avanzar o retroceder un paso o medio paso.

Secuencia para moverse un paso

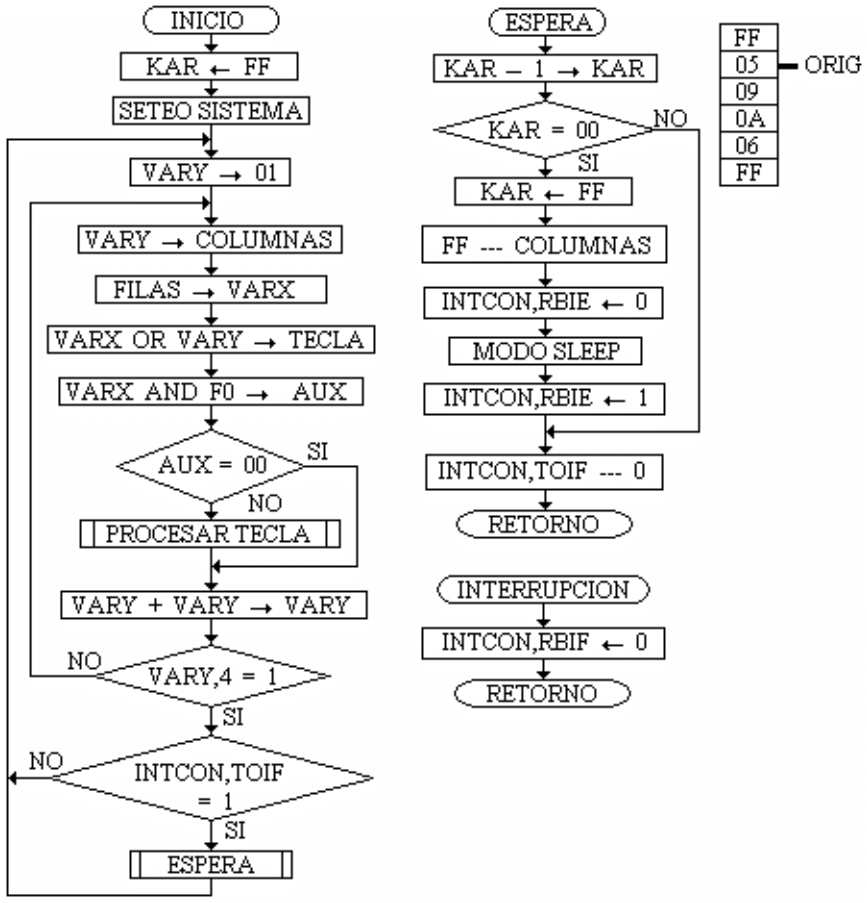
D	C	B	A
0	1	0	1
1	0	0	1
1	0	1	0
0	1	1	0

Para mover medio paso

D	C	B	A
0	1	0	1
1	1	0	1
1	0	0	1
1	0	1	1
1	0	1	0
1	1	1	0
0	1	1	0
0	1	1	1

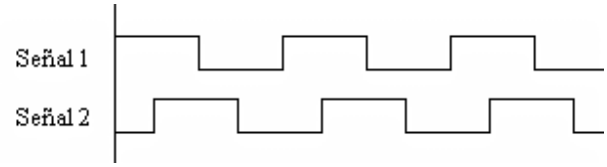
Trataremos el caso de un motor de paso conectado al nible bajo del puerto MOTOR. Y que es comandado por los byte 0 y 1 del puerto TECLADO, para hacer avanzar y retroceder el motor respectivamente.
Debemos recordar cual es el estado en que estamos para decidir cual será el siguiente. Si es el mismo el motor estará quieto y frenado. El microcontrolador es capaz de variar el comando mas rápido de lo que el motor es capaz de seguir. Se hace una pausa entre un dato presentado y otro, cuanto mas cortas sean mas rápido variara su posición el motor. El limite lo dispone las características físicas del motor. Si se excede se dice que el motor pierde el paso.
Una técnica es usar una tabla con los valores ordenados de la secuencia y tener un vector que lo recorra. Según el vector aumente o disminuya recorrerá la secuencia hacia adelante o atrás.
Veremos el caso con avances de un paso.
La tabla seria

Ptabla	FF	Primer valor de la tabla es un marcador del inicio de secuencia
+1	05	
+2	09	
+3	0A	
+4	06	
+5	FF	Último valor de la tabla es un marcador del fin de secuencia

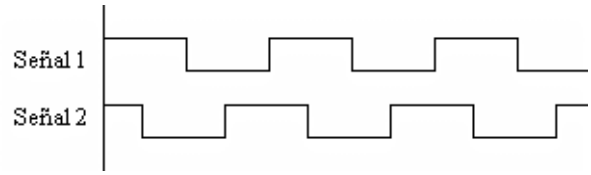


Decoder, Shift Decoder o Decodificador Angular
Al variar la posición angular de su eje, el decodificador genera dos ondas rectangulares en cuadratura. O dos trenes de pulsos desfasados 90º grados.
Habiendo más de una tecnología para fabricarlos, solo nos interesa interpretar y manejar sus señales de salida.

Puede darse dos casos para la relación entre las señales.
Caso a:

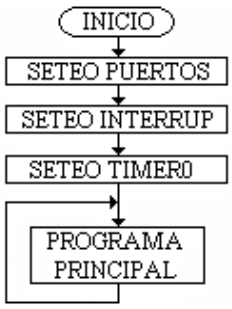


Caso b:



La diferencia entre ambos casos es el valor de la señal 1 al ocurrir el flanco ascendente de la señal 2. En el caso A vale 1 y en el caso B vale 0. En el caso A el sentido de giro es uno, y en el caso B el sentido es el opuesto.

Contando el número de pulsos de cualquiera de las dos señales se puede determinar el ángulo variado. El fabricante suministra el número de grados por pulso o el número de pulsos por vuelta.



Consideremos un ejemplo de programa para leer un sensor de esta clase. Usaremos la interrupción por flanco en RB7 para detectar los flancos de la señal S1. Detectar el flanco de S1 mediante una interrupción nos permite reducir el software o sea simplificar el programa. Todo lo hecho por software disminuye el hardware y viceversa.
Cuando es necesario determinar la velocidad de giro y el sentido de giro.
Se presenta el limite de cuando consideramos velocidad cero.
Consideramos que después de cierto tiempo (velocidad mínima) el decoder esta detenido. Se utiliza la interrupción de TMR0 para contar el tiempo que dura cada pulso. Debido a no saber donde quedara quieto el decoder se debe medir el tiempo para el nivel bajo y alto de S1.
Consideramos que a la mínima velocidad la duración del pulso será máxima. Nuestro contador de tiempo debe ser lo suficientemente grande como para no desbordarse. Alcanza con utilizar más registros para aumentar el tamaño del contador
A partir de que ocurre el flanco en S1 determinamos el sentido de giro. Activamos un contador decreciente.
El contador disminuye con cada desborde del TMR0.
Se reinicia cada vez que S1 cambia de valor. Debido a que la duración depende de la velocidad angular del eje y esta puede variar, no siempre será una señal simétrica.
Cuando nuestro contador sea cero decimos que el decoder esta detenido. Nuestro otro limite es la velocidad máxima. La duración del TMR0 nos debe permitir detectar la máxima velocidad.

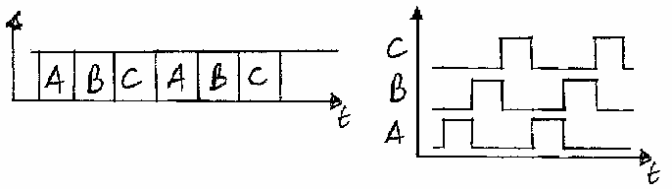
```
 ;Programa para el manejo de un decoder
;Primero elegimos la clase de micro a usar y seteamos la palabra de configuración.
                LIST        p=16f84
                __CONFIG    0x3FF1          ;Seteo XT, No CP, No WDT, PWRTE
                __IDLOCS    0x00C0          ;Le asigno un número arbitrario

;-------Declaro las variables a usar en el programa fuente-----
W               EQU         H'0'
F               EQU         H'1'

;----- Register Files----------------------------------------------------
INDF            EQU         H'0000'
TMR0            EQU         H'0001'
PCL             EQU         H'0002'
STATUS          EQU         H'0003'
FSR             EQU         H'0004'
PORTA           EQU         H'0005'
PORTB           EQU         H'0006'
INTCON          EQU         H'000B'
OPCION          EQU         H'0081'
TRISA           EQU         H'0085'
TRISB           EQU         H'0086'

;----- STATUS Bits --------------------------------------------------------
IRP             EQU         H'0007'
RP1             EQU         H'0006'
RP0             EQU         H'0005'
TO              EQU         H'0004'
PD              EQU         H'0003'
Z               EQU         H'0002'
DC              EQU         H'0001'
CY              EQU         H'0000'

;----- INTCON Bits --------------------------------------------------------
GIE             EQU         H'0007'
EEIE            EQU         H'0006'
T0IE            EQU         H'0005'
INTE            EQU         H'0004'
RBIE            EQU         H'0003'
T0IF            EQU         H'0002'
INTF            EQU         H'0001'
RBIF            EQU         H'0000'

;----- OPTION Bits --------------------------------------------------------
RBPU            EQU         H'0007'     ;Habilitación de resistencias pull-up en PORTB
INTA            EQU         H'0006'     ;Selector de flanco activo de RB0
TOCS            EQU         H'0005'     ;Selector de fuente para TMR0
TOSE            EQU         H'0004'     ;Flanco de activación de RA4/TOCKI
PSA             EQU         H'0003'     ;Destino del pre-escalador
PS2             EQU         H'0002'     ;Selectores del valor del pre-escalador
PS1             EQU         H'0001'
PS0             EQU         H'0000'

;------------Declaro mis propias variables, constantes y etiquetas-----------

                ORG         0x000
INICIO
                goto        PRINCIPAL       ;Salto por encima del comienzo
                                            ;de las subrutinas e interrupciones
                ORG         0x004
INTERRUPCIONES
                movf        AUX1            ;Guardo W
```

```
                movwf   STATUS,W
                movf    AUX2            ;Guardo STATUS
                movwf   FSR,W
                movf    AUX3            ;Guardo FSR
DETERMINO_INTERRUPCION
                btfsc   INTCON,RBIF
                goto    FLANCO_S1       ;Detecto si es por cambio en S1

                btfsc   INTCON,T0IF
                goto    TIMER0          ;Detecto si es por desborde del Timer0
VOLVER
                movwf   AUX3,W
                movf    FSR             ;Recupero FSR
                movwf   AUX2,W          ;Recupero
                movf    STATUS          ;Recupero STATUS
                movwf   AUX1,W          ;Recupero W

                retfie

TIMER0                                  ;Proceso desborde del timer0
                decfsz  CONT1,F         ;Decremento contador bajo
                goto    SALTO1
                decfsz  CONT2,F         ;Decremento contador alto
                goto    SALTO1
                bcf     INTCON,T0IE     ;Deshabilito TMR0 por estar detenido
                                        ;el decoder y llegar a cero el contador
SALTO1
                movlw   0xFA
                movf    TMR0            ;Reinicio el timer0
                bcf     INTCON,T0IF     ;Limpio el aviso de desborde
                goto    VOLVER          ;Voy a retornar correctamente

;Proceso flanco en S1. Según el flanco , el valor de S2 significa un sentido u otro de giro
FLANCO_S1                               ;Elijo según sea el flanco de S1
                btfsc   PORTB,S1        ;Si S1 vale cero busco detectar flanco
                goto    F_SUBIDA        ;de subida,  y si es 1el de bajada
F_BAJADA                                ;Si S2 vale 0 horario, si vale 1 antihorario
                movlw   HORA
                btfsc   PORTB,S2        ;Determino valor de S2
                movlw   ANTIHORA
                movf    SENTIDO
;---- Ajusto INTA para el próximo flanco
                bsf     STATUS,RP0      ;Cambio de Banco
                bsf     OPCION,INTA     ;Debo detectar ascendentes
                bcf     STATUS,RP0      ;Cambio de Banco
                goto    DECODER
F_SUBIDA                                ;Si S2 vale 1 Horario, si vale 0 antihorario
                movlw   HORA
                btfss   PORTB,S2        ;Determino valor de S2
                movlw   ANTIHORA
                movf    SENTIDO
;---- Ajusto INTA para el próximo flanco
                bsf     STATUS,RP0      ;Cambio de Banco
                bcf     OPCION,INTA     ;Debo detectar descendentes
                bcf     STATUS,RP0      ;Cambio de Banco

;-----Guardo ultima velocidad del decoder----------
DECODER
                movwf   CONT1,W
                movf    VELO1           ;Guardo velocidad baja
                movwf   CONT2,w
                movf    VELO2           ;Guardo velocidad alta
```

```
                movlw       0xFF                ;Reinicio contadores de velocidad
                movf        CONT1
                movf        CONT2
                movlw       0xFA
                movf        TMR0

                bsf         INTCON,T0IE         ;Habilito TMR0 por si estaba enmascarado
                bcf         INTCON,INTF         ;Limpio aviso de flanco en S1
;------------Termine subrutinas de interrupción ----------------

PRINCIPAL
                bsf         STATUS,RP0  ;Cambio al banco 1
                movlw       0x82        ;S1 entra por RB7 y S2 por RB1
                movf        TRISB       ;Solo me interesan dos líneas del puerto B
                movlw       '10110000'B ;
                movf        INTCON      ;Seteo interrupciones.
                movlw       '11000110'B
                movf        OPCION      ;Seteo Timer0 con pre-escalador 256
                bcf         STATUS,RP0  ;Cambio al banco 0
                movlw       0xFA
                movf        TMR0        ;Inicializo Timer0

LOOP
                nop                     ;No me interesa el programa principal
                                        ;Este procesa los datos obtenidos de S1 y S2
                goto        LOOP

                END                     ;Termino el programa con un enter
```

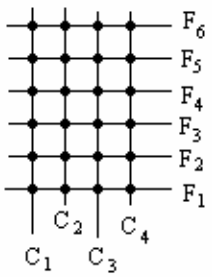
CONTROL DE ELEMENTOS MEDIANTE MULTIPLEXION EN EL TIEMPO

Multiplexión en el tiempo:
Significa atender varios elementos sucesivamente dedicando un periodo pequeño de tiempo a cada uno. El resultado es similar a atender todos los elementos al unísono.
Dado un intervalo de tiempo *Δt* y *n* elementos; cada elemento es atendido durante un tiempo *Δt* y vuelto a atender en un período *n.Δt* llamado período de multiplexión.
Cada elemento o dispositivo es leído o escrito durante Δt, que se llama ranura (slot) o ventana de tiempo. Después de atender a todos los elementos vuelve a atender al primero. Esto se logra utilizando una línea del bus de control para cada dispositivo. Mediante estas líneas de control, una per capita, se selecciona con cual elemento se esta actuando.
En la grafica del tiempo a pequeña escala se ve como se trabaja con cada uno de los dispositivos en forma individual. A gran escala se ve a todos los dispositivos actuando juntos.



Si se encendiera sucesivamente los bit 0, 2 y 4 de un puerto, y si la duración de la temporización es grande se observa la secuencia de encendido de los led. Si la temporización es pequeña se ven los 3 led brillar juntos.
Si la velocidad es muy alta brillan poco y si es muy lenta se nota el parpadeo. Se debe buscar una velocidad que permita verlos cómodamente, ni muy rápida ni muy lenta.
También hay un límite en el número de elementos que se pueden multiplexar. Cuantos más elementos tengamos más pequeña será la relación de la ranura de tiempo con el periodo del multiplexado [Δt/(n.Δ)t] . Este límite depende del hardware de los periféricos a controlar.

Matriz de Elementos
Si se tienen muchos dispositivos que controlar se recurre a las matrices de elementos.
Se acomodan los elementos en subgrupos o columnas. A cada columna se le asigna una línea de control. Todos los elementos de una columna tienen conexión con la misma línea de control de dicho subgrupo. Dentro de cada columna se ordenan los dispositivos para poderles asignar un cardinal (1, 2, 3, etc.)
A su vez todos los elementos de igual cardinal (por ejemplo 2) de cada columna (uno por columna) comparten otra línea de control. O sea forman una fila. Así tendrán tantas filas como el orden mayor de cada columna.
Cada elemento tiene dos líneas de control, una debido a la columna a la que pertenece, y otra por la fila a la que pertenece. Se identifica cada dispositivo por la combinación de fila y columna por ejemplo: E_{12} , E_{22} o E_{31}
Estos ejemplos son de una matriz bidimensional, pero puede extrapolarse a n-dimensiones. Cada nueva dimensión agrega una línea de control.



$$\begin{matrix} E_{16} & E_{26} & E_{36} & E_{46} \\ E_{15} & E_{25} & E_{35} & E_{45} \\ E_{14} & E_{24} & E_{34} & E_{44} \\ E_{13} & E_{23} & E_{33} & E_{43} \\ E_{12} & E_{22} & E_{32} & E_{42} \\ E_{11} & E_{21} & E_{31} & E_{41} \end{matrix}$$

Multiplexado de Matrices

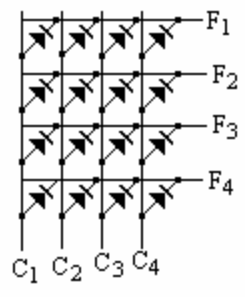
Aunque se puede multiplexar en el tiempo a todos los elementos de un grupo extenso, resulta una ventana muy angosta en relación al periodo. Además de necesitar muchas líneas de control, una por cada elemento.

Al configurar la matriz bidimensional cada elemento tiene 2 o más líneas de control, las que comparte con otros elementos. En cada ventana de tiempo se atiende una columna entera. Es decir a todos los elementos de la columna al unísono. De esta manera se puede hacer una ranura más ancha en proporción al período.

La atención a cada columna es multiplexada en el tiempo. Se tiene una línea de activación (control) por cada columna, la cual es común a todos los elementos de este subgrupo llamado columna.

Manejo de Matrices por Multiplexión en el Tiempo

Matriz de salida:



En el caso de una matriz de led o sea una matriz de elementos de salida o display de datos, tanto F_1-F_4 como C_1-C_4 son salidas. Las filas entregan datos para encender y las columnas para seleccionar.

Lo escrito en cada puerto se mantendrá hasta que se re-escriba nuevamente o se reinicie el sistema. Inclusive se mantiene la información en modo SLEEP.

Si para las filas y las columnas se usan puertos diferentes, el PIC16F84A no puede escribir en ambos al mismo tiempo, al menos hay dos o tres ciclos de reloj entre la escritura de un puerto y el otro.

Se debe coordinar el valor entregado a las filas con el dado a las columnas. Para la columna C_1 se presenta el dato D_0 en las filas, y así sucesivamente. Para C_2 es D_1, para C_3 es D_2 y para C_4 es D_3.

Aunque la velocidad para escribir en un registro, y después en otro, es muy alta los led son dispositivos muy rápidos para brillar. Esto plantea el siguiente problema llamado solapamiento de información.

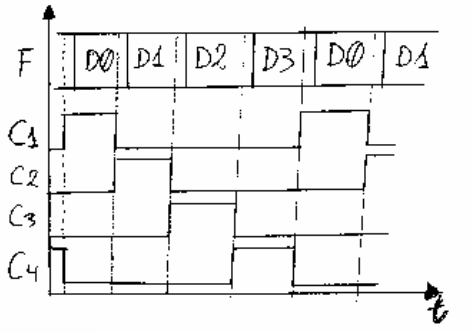
Se escribe en las columnas, o sea se activa una de ellas. Un par de ciclos maquina después (algunos microsegundos) se escribe el dato que le corresponde en las filas. El tiempo que se activa una columna es mucho mayor que un ciclo de instrucción, pasado el mismo se desactiva esa columna y se activa la siguiente. Este cambio de columna se realiza en una sola instrucción de escritura en el puerto que controla las COLUMNAS.

Como aún no se ha cambiado lo escrito en las filas, esta presente el dato de la columna anterior. Inmediatamente después (un par de microsegundos) se corrige el dato de las filas al correspondiente con la actual columna.

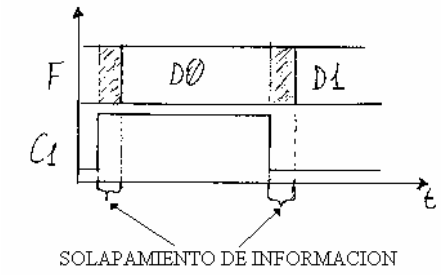
Esto implica que cada columna tendrá presente los datos de la columna anterior durante los primeros microsegundos de su activación.

Se debe recordar que los led son muy rápidos para encender, por lo tanto se verá el dato de la columna anterior con un tenue brillo de los led correspondientes. No quedan completamente oscuros. Esto es el solapamiento de información entre las columnas. Se observa como lo escrito en una columna aparece tenue en la siguiente.

En un diagrama de tiempo seria:



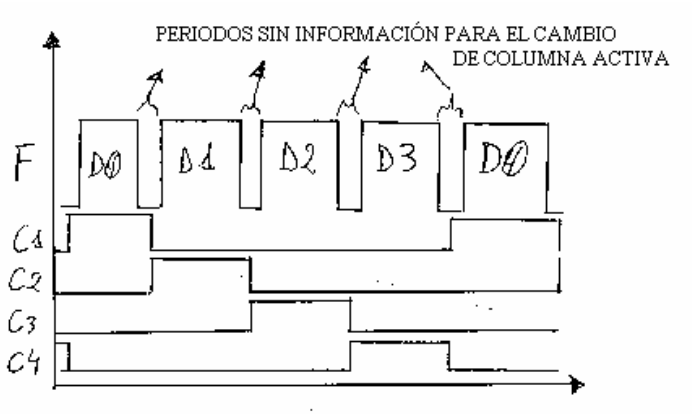
Considerando un acercamiento a la activación de una columna



El solapamiento esta exagerado en el diagrama. Pero por pequeño que sea puede ser detectado por los dispositivos de salida, dependiendo de la inercia de respuesta.

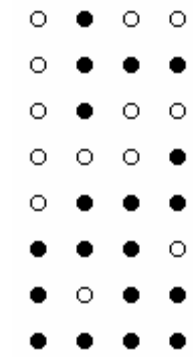
A este problema se le llama solapamiento de información. La forma de solucionar esta situación es no enviar información a las filas antes de cada cambio de columna. Es decir momento antes del cambio de columna dejar las filas sin datos.

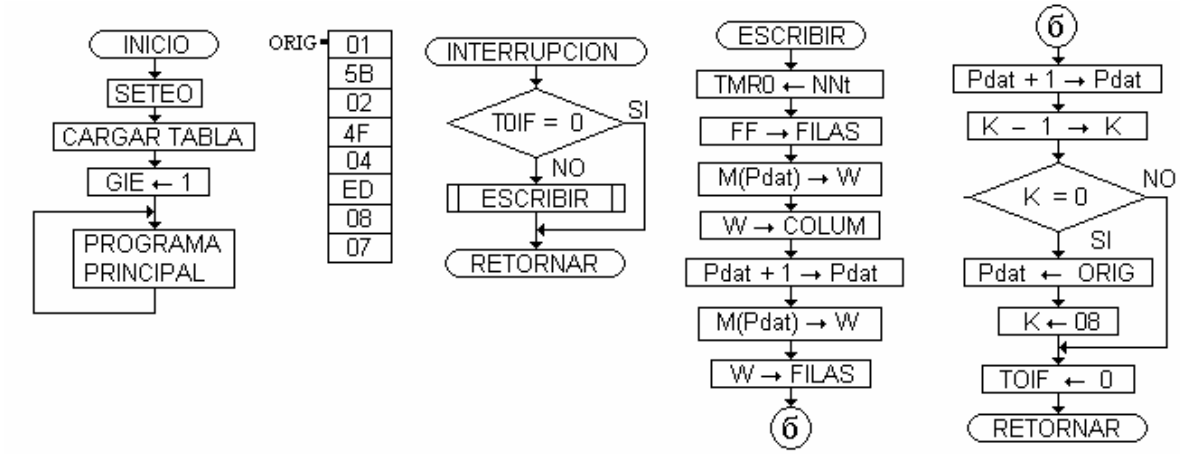
En este caso seria apagar los led antes del cambio de columna activa. El diagrama de tiempo entonces queda así:



Por lo tanto antes de activar una columna (o sea un grupo de dispositivos), se deben apagar (dejar sin información) a todos los elementos de la columna. Luego se activa la nueva columna y después sí presentar los datos correspondientes a la columna actual.

Los datos para las filas se deben coordinar con las columnas, se tratan como una estructura y se le asocia en una serie. Por lo tanto se les invoca con un puntero.
El usar un puntero permite modificar lo escrito en la matriz de led más fácilmente.
Para rotar hacia la izquierda un valor en binario también se le puede sumar a si mismo o sea multiplicarlo por 2.
Considerar una matriz de 4 columnas x 8 filas. Realizar un programa que muestre en la matriz de led esta forma
Dato 1: E0
Dato 2: B7
Dato 3: F2
Dato 4: DA





```
;Programa de manejo de una matriz de 4x8 puntos formada por LEDs
;La manejo por multiplexion en el tiempo. A intervalos regulares
; mediante interrupcion de TIMER0 escribo en la matriz
;año 2005
        LIST        p=16f84
    __CONFIG     0x3FF1
    __IDLOCS     0x0009

;Declaro nombres de los SFR

W               EQU   .0        ;Destino el registro de trabajo
F               EQU   .1        ;Destino el propio registro direccionado

;----- Register Files----------------------------------------------------
INDF            EQU   H'00'      ;Registro auxiliar para direcc. Indirecto por registro
TMR0            EQU   H'01'      ;Registro de buffer del TIMER0
PCL             EQU   H'02'      ;Parte baja del PC
STATUS          EQU   H'03'      ;Palabra de estado del programa
FSR             EQU   H'04'      ;Puntero de la memoria RAM para direcc. Indirecto
PORTA           EQU   H'05'      ;Buffer de los pines RA0-RA4
PORTB           EQU   H'06'      ;Buffer de los pines RB0-RB7
INTCON          EQU   H'0B'       ;Registro de control de interrupciones
OPCION          EQU   H'81'      ;Registro de configuración de periféricos
TRISA           EQU   H'85'      ;Registro de control de dirección del puerto A
TRISB           EQU   H'86'      ;Registro de control de dirección del puerto B

;----- STATUS Bits --------------------------------------------------------
IRP             EQU   .7        ;Bit de selección bancos para el FSR
RP1             EQU   .6        ;Bit alto de selección de bancos RAM en direcc. directo
RP0             EQU   .5        ;Bit bajo de selección de banco RAM en direcc. directo
TO              EQU   .4        ;Bit de estado del micro
PD              EQU   .3        ;Bit de estado del micro
Z               EQU   .2        ;Bandera de resultado cero
DC              EQU   .1        ;Bandera de semiacarreo en sumas y restas
CY              EQU   .0        ;Bandera de acarreo en sumas y restas

;----- INTCON Bits --------------------------------------------------------
GIE             EQU   .7        ;Habilitación general de todas las interrupciones
EEIE            EQU   .6        ;Mascara de interrup final del ciclo de escritura EEPROM
T0IE            EQU   .5        ;Mascara de interrupción por desborde del TIMER0
INTE            EQU   .4        ;Mascara de interrupción por evento en RB0
RBIE            EQU   .3        ;Mascara de interrupción por cambios en RB4-RB7
T0IF            EQU   .2        ;Bandera de aviso de desborde de TIMER0
INTF            EQU   .1        ;Bandera de aviso de evento en RB0
RBIF            EQU   .0        ;Bandera de aviso de cambio en el nible alto del puerto B

;----- OPTION Bits --------------------------------------------------------
RBPU            EQU   .7        ;Habilitación de pull-up en el PB
INTA            EQU   .6        ;Selector de flanco activo de RB0
T0CS            EQU   .5        ;Selector de fuente de incremento del TIMER0
T0SE            EQU   .4        ;Selector de flanco activo de RA4
PSA             EQU   .3        ;Selector de uso del pre-escalador
PS2             EQU   .2        ;Bit de configuración del pre-escalador
PS1             EQU   .1        ;Bit de configuración del pre-escalador
PS0             EQU   .0        ;Bit de configuración del pre-escalador

;Defino las variables y constantes propias
```

```
ORIG        EQU   0X20   ;Defino inicio de la tabla
NNt         EQU   0x06
PDAT        EQU   0x11
FILAS       EQU   PORTB
COLUM       EQU   PORTA
AUX_W       EQU   0x12
AUX_ST      EQU   0X13
AUX_F       EQU   0X14
K           EQU   0X10


        ORG   0x000
RESET
        goto  INICIO

        ORG   0x004
INTERRUPCIONES
;Primero salvar los registros importantes
; W, STATUS, FSR, ETC
        movwf       AUX_W       ;SAlvo el W
        swapf       STATUS,W
        movwf       AUX_ST      ;Salvo el STATUS
        movf        FSR,W
        movwf       AUX_F       ;Salvo el FSR para liberar el programa principal

;Verifico si la interrupcion fue por el TMR0
        btfsc       INTCON,T0IF
        call        ESCRIBIR    ;En caso de ser llamo a la subrutina adecuada

FIN_INTERRUPCION
;Recupero los registros
        movf        AUX_F,w
        movwf       FSR         ;Recupero el FSR
        swapf       AUX_ST,W
        movwf       STATUS      ;Recupero el STATUS
        swapf       AUX_W,F
        swapf       AUX_W,W     ;Recupero W sin alterar STATUS
        retfie


SUBRUTINAS
ESCRIBIR                ;Manejo los datos de la matriz como una serie de datos dobles.
        movlw       NNt
        movwf       TMR0        ;Recargo el TMR0
        movlw       0xFF
        movwf       FILAS       ;Apago las filas antes de cambiar de columna
        movf        PDAT,W
        movwf       FSR         ;Cargo el puntero lógico en el puntero físico.
        movf        INDF,W
        movwf       COLUM       ;Cambio de columna

        incf        PDAT,F      ;Actualizo el puntero lógico
        incf        FSR,F       ;Actualizo el puntero físico
        movf        INDF,W
        movwf       FILAS       ;Cargo el nuevo valor de filas para la actual columna
        incf        PDAT,F      ;Actualizo el puntero lógico
                                ;pero no el físico pues no lo utilizo
```

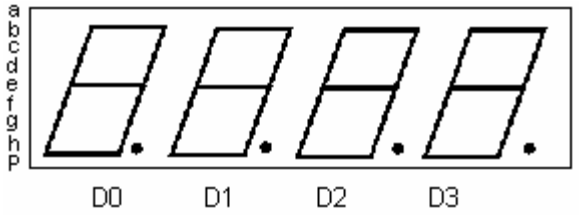
```
        decfsz      K,F
        goto        VOLVER      ;Verifico si ya escribe en las 4 columnas

        movlw       ORIG
        movwf       PDAT        ;Reinicio los contadores de la matriz
        movlw       0x04
        movwf       K

VOLVER
        bcf         INTCON,T0IF

        return
;Termine con las subrutinas
;Comienza el programa principal
        ORG         0x0100
INICIO
;Seteo perifericos, ambos puertos como salidas
        bsf         STATUS,RP0  ;Cambio al Banco 1
        clrf        TRISA
        clrf        TRISB

;Seteo TIMER0 entre cada 128 microseg y 32 mseg
        bcf          OPCION,T0CS  ;FUente del TIMER0 es CK
        bcf         OPCION,PSA  ;Utiliza el pre-escalador
        bsf         OPCION,PS2  ;Pre-escalador en 1/128
        bsf         OPCION,PS1
        bcf         OPCION,PS0  ;

        bcf         STATUS,RP0  ;CAmbio al Banco 0
        movlw       NNt
        movwf       TMR0        ;Seteo el TMR0

;Seteo interrupciones
        bcf         INTCON,T0IF ;Por las dudas limpio aviso de interrupcion
        bsf         INTCON,T0IE ;Habilito la interrupcion por TMR0

;Cargar tabla en la memoria RAM de los datos a enviar por las columnas y las filas
        movlw       0x01
        movwf       0x20
        movlw       0xE0
        movwf       0x21
        movlw       0x02
        movwf       0x22
        movlw       0xE7
        movwf       0x23
        movlw       0x04
        movwf       0x24
        movlw       0xF2
        movwf       0x25
        movlw       0x08
        movwf       0x26
        movlw       0xDA
        movwf       0x27

;Seteo las variables propias
        movlw       ORIG
        movwf       PDAT
        movlw       0x04
        movwf       K
```

```
        bsf     INTCON,GIE      ;Habilito las interrupciones

LAZO
        ;Desarrollo del programa principal
        goto    LAZO

        END
```

Display de Segmentos:

La presentación de información a un operador humano implica que esta debe ser inteligible por el mismo. Para sistemas pequeños una forma es a través de un display de números. Lo estándar es usar display que forma los símbolos con 7 segmentos, pero los hay de 8 y 16 segmentos.

Trataremos el caso de un display de 4 dígitos formados por 7 segmentos y un punto para cada digito. Un display de 4 dígitos de 7 segmentos con punto implica tener 32elementos para activar. Podemos utilizar un micro con 32 salidas o multiplexarlas y encender por grupos de 8 elementos.
Encendemos un digito por vez, eso se llama multiplexar en el tiempo al display. Si lo hacemos suficientemente rápido veremos los 4 símbolos encendidos sin percibir que parpadean.
A cada digito se le asocia con un valor para los segmentos, guardados en una tabla a partir de la posición ORIG. Con variar el contenido del puntero (Pdat) modificamos los segmentos a encender. Y si variamos el contenido de la tabla cambiamos lo escrito en el display, pero la rutina de escritura en el display es la misma.
El procedimiento es activar un dígito y encender los segmentos seleccionados para ese digito, guardado en M(Pdat). Para el display que consideramos con los dígitos designados D0, D1, D2 y D3 se le asocian M(ORIG +0), M(ORIG + 1), M(ORIG +2), M(ORIG + 3)



Antes de activar cada digito debemos apagar los segmentos, pues estos contienen información del digito anterior. Sino veríamos el símbolo anterior superpuesto con el actual. También se puede hacer al revés, es decir primero apagar todos los digito y seleccionar los segmentos y después encender el digito correspondiente.
Cada símbolo debe permanecer presentado un momento antes de seguir escribiendo en el digito siguiente, así se fija la figura de símbolo. A su vez el tiempo debe ser el mismo para todos los dígitos, sino presentarían diferencias en el brillo.
A los valores de los segmentos para cada digito se les trata como una serie, de 4 elementos, coordinada con el digito a activar o activado.
Considerando este caso:

Cada segmento se activa con 0
Cada digito se selecciona con 1

Asociamos cada digito con un bit de la variable DIGITOS y cada segmento con un bit de la variable SEGMENTOS

DIGITO,0	selecciona D0
DIGITO,1	selecciona D1
DIGITO,2	selecciona D2
DIGITO,3	selecciona D3
SEGMTO,0	activa a
SEGMTO,1	activa b
SEGMTO,2	activa c
SEGMTO,3	activa d
SEGMTO,4	activa e
SEGMTO,5	activa f

SEGMTO,6 activa g
SEGMTO,7 activa p

Se pueden representar todos los símbolos decimales y la mayoría de las letras en este display. Cada símbolo tendrá un valor que encenderá los segmentos que formaran su figura.

Símbolo	Valor
0	3F
1	06
2	5B

etc..

Si quisiera ver escrita la palabra "HOLA" en el display seria así: se enciende "H" en D3, se apaga y enciende "O" en D2, se apaga y enciende "L" en D1, se apaga y enciende "A" en D0, se apaga y enciende "H" en D3, y así continuamente y si es rápido nuestro ojo no notara que los caracteres parpadean.

Carácter "A" valor 10001000_B 88_H

Carácter "L" valor 11000111_B $C7_H$

Carácter "O" valor 11000000_B $C0_H$

Carácter "H" valor 10001001_B 89_H

Mostrar la información en un display, en general es una función secundaria, por lo tanto la rutina del display debe ser una subrutina del programa principal. Lo más importante es escribir a intervalos iguales de tiempo, permitiendo que todos los segmentos tengan igual brillo. Un procedimiento es mediante una interrupción por desborde del TMR0.

Ejercicio:

Realizar un programa que muestre por el display la palabra HOLA y que cada 10 segundos rote las letras hacia la derecha. Cada vez que se aparezca un pulso en RB7 reinicie la posición de las letras

Al entrar en la interrupción lo primero es determinar cual es la causa de la misma. En el caso de no habilitar ninguna otra fuente es un paso innecesario. La determinación de cual es la fuente se hace verificando las banderas de aviso de interrupción. En el caso de desborde del TMR0 es verificando el bit T0IF del registro INTCON.

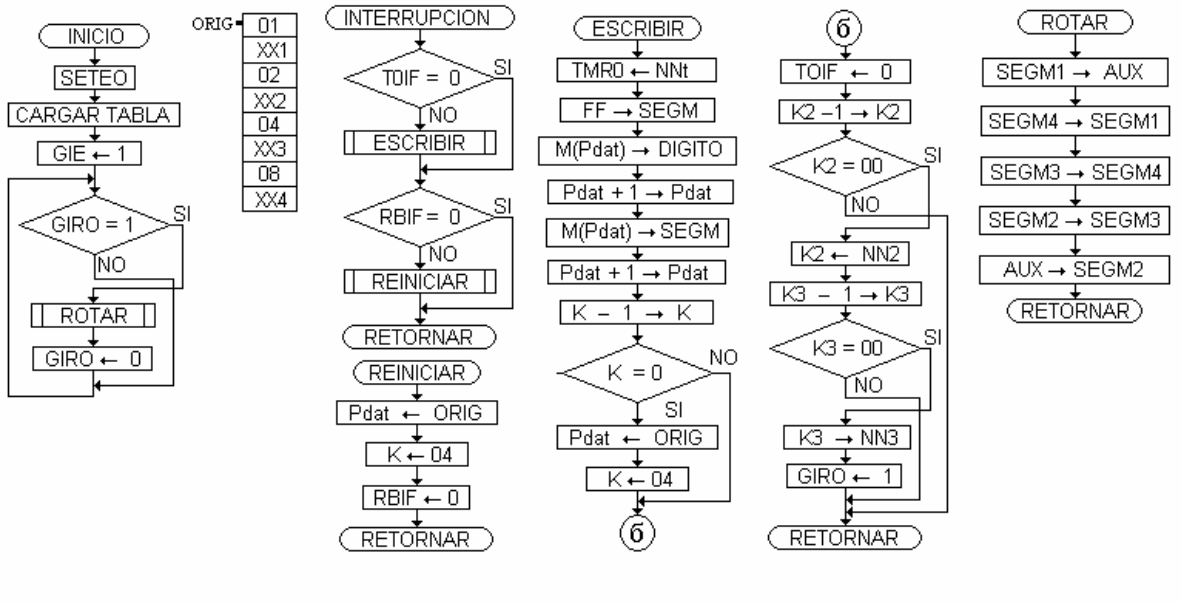
Como la interrupción no es predecible cuando ocurre o no interesa, se debe prever no alterar el programa principal. Los registros más importantes para el mantenimiento normal del programa son W, STATUS y FSR. Los guardamos en registros auxiliares, debido a que este micro no cuenta con una pila de uso general. En la pila solo se guarda el PC. Previo a salir de la subrutina restauramos el valor de estos registros.

Obsérvese como la subrutina no necesita el retardo, pues el tiempo que da el desborde del TMR0 es suficiente para activar el led.

Al retornar de la subrutina se habilita la interrupción por TMR0 y las interrupciones en general.

La señal en RB7 se trata como interrupción mediante el aviso en RBIF. Este aviso ocurre cada vez que hay un flanco en RB7, por lo tanto se debe discriminar cuando tiene un nivel lógico activo.

La rotación de las letras se sincroniza con la escritura del display, mediante un aviso pero forma parte del programa principal y no de la interrupción.




```
;Programa general con rutina de manejo de display de 4 dígitos de 7 segmentos.
;Primero elegimos la clase de micro a usar y seteamos la palabra de configuración.
                LIST        p=16f84
                __CONFIG    0x3FF1          ;Seteo XT, No CP, No WDT, PWRTE
                __IDLOCS    0x000A          ;
;-------Declaro las variables a usar en el programa fuente-----
W               EQU         H'0'
F               EQU         H'1'

;----- Register Files----------------------------------------------------
INDF            EQU         H'0000'
TMR0            EQU         H'0001'
PCL             EQU         H'0002'
STATUS          EQU         H'0003'
FSR             EQU         H'0004'
PORTA           EQU         H'0005'
PORTB           EQU         H'0006'
INTCON          EQU         H'000B'
OPCION          EQU         H'0081'
TRISA           EQU         H'0085'
TRISB           EQU         H'0086'

;----- STATUS Bits --------------------------------------------------------
IRP             EQU         H'0007'
RP1             EQU         H'0006'
RP0             EQU         H'0005'
TO              EQU         H'0004'
PD              EQU         H'0003'
Z               EQU         H'0002'
DC              EQU         H'0001'
CY              EQU         H'0000'
;----- INTCON Bits --------------------------------------------------------
GIE             EQU         H'0007'
EEIE            EQU         H'0006'
T0IE            EQU         H'0005'
INTE            EQU         H'0004'
RBIE            EQU         H'0003'
T0IF            EQU         H'0002'
INTF            EQU         H'0001'
RBIF            EQU         H'0000'
```

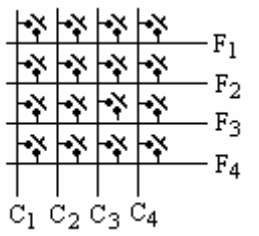
```
;----- OPTION Bits ----------------------------------------------------
RBPU        EQU         H'0007'     ;Habilitación de resistencias pull-up en PORTB
INTA        EQU         H'0006'     ;Selector de flanco activo de RB0
TOCS        EQU         H'0005'     ;Selector de fuente para TMR0
TOSE        EQU         H'0004'     ;Flanco de activación de RA4/TOCKI
PSA         EQU         H'0003'     ;Destino del pre-escalador
PS2         EQU         H'0002'     ;Selectores del valor del pre-escalador
PS1         EQU         H'0001'
PS0         EQU         H'0000'

;------------Declaro mis propias variables, constantes y etiquetas-----------
ORIG        EQU         0x30        ;Defino el comienzo de la tabla
CX          EQU         0x20        ;Registro intermediario con los dígitos
Pdat        EQU         0x25        ;Puntero de la tabla
DIGITO      EQU         PORTA       ;Los dígitos se seleccionan por el puerto A
SEGM        EQU         PORTB       ;Los segmentos se conectan al puerto B
AUX1        EQU         0x21        ;Registro para guardar W durante la interrupción
AUX2        EQU         0x22        ;Registro para guardar STATUS durante interrupción
AUX3        EQU         0x24        ;Registro para guardar FSR durante la interrupción
DATO        EQU         0x00        ;Elemento direccionado por el FSR

;--------------Declaro el valor de los posibles símbolos a representar en el display.------------------
;                                     pgf edcba    activos en cero
S_0         EQU         b'11000000'
S_1         EQU         b'11111001'
S_2         EQU         b'10100100'
S_3         EQU         b'10110000'
S_4         EQU         b'10011001'
S_5         EQU         b'10010010'
S_6         EQU         b'10000010'
S_7         EQU         b'11111000'
S_8         EQU         b'10000000'
S_9         EQU         b'10011000'

S_GUION     EQU         b'10111111'
S_A         EQU         b'10001000'
S_B         EQU         b'10000011'
S_C         EQU         b'11000110'
S_D         EQU         b'10100001'
S_E         EQU         b'10000110'
S_F         EQU         b'10001110'
S_G         EQU         b'10010000'
S_H         EQU         b'10001001'
S_I         EQU         b'11001111'
S_J         EQU         b'11100000'
S_L         EQU         b'11000111'
S_N         EQU         b'10101011'
S_O         EQU         b'11000000'
S_P         EQU         b'10001100'
S_R         EQU         b'10101111'
S_S         EQU         b'10100100'
S_T         EQU         b'10001111'
S_U         EQU         b'11000001'
S_V         EQU         b'11100011'


;Comienzo el programa en la posición 0x000
            ORG         0x00
PRINCIPIO
            goto        INICIO      ;Salto por encima de la subrutina de interrupción
;------------Subrutina de atención a interrupciones-------------
            ORG         0x004       ;posición de inicio indicada por el fabricante
```

```
INTERRUPCION
                btfss           INTCON,T0IF   ;Verifico el origen de la interrupción
NO_TIMER0
                retfie                        ;Si no es TMR0 regreso o ejecuto otra subrutina
;------------Sección dedicada al desborde del Timero--------------
SI_TIMER0
                movwf           AUX1          ;Guardo una copia de lo principales registros
                movf            STATUS,W
                movwf           AUX2
                movf            FSR,W
                movwf           AUX3
ESCRIBIR_DISPLAY
                movf            Pdat,W
                movwf           FSR
                clrf            SEGM          ;Apago todos los segmentos
                movf            CX,w
                movwf           DIGITO        ;Activo siguiente digito
                movf            DATO,W                ;Traigo el próximo valor a encender
                movwf           SEGM          ;Enciendo los segmentos correspondientes
                incf            Pdat,F        ;Apunto al próximo valor a desplegar
                movf            CX,W
                addwf           CX,F          ;Desplazo los bit hacia la izquierda
                btfss           CX,.4         ;Verifico si ya active todas las columnas
                goto            SIGUIENTE     ;Si no es así sigo con la próxima
AL_PRIMERO
                movlw           ORIG          ;Si ya active todas vuelvo a la primera
                movwf           Pdat          ;Reinicio el puntero de la tabla
                movlw           0x01
                movwf           CX            ;Selecciono de nuevo el primer digito
SIGUIENTE
                ;Si aun queda alguna no necesito reiniciar ni Pdat  ni CX
;---------Termine la Subrutina  de Interrupción y vuelvo correctamente al programa principal------
                movf            AUX3,W        ;Recupero el FSR
                movwf           FSR
                movf            AUX2,W
                movwf           STATUS        ;Recupero el registro STATUS. Los bits PO y PD
                                              ;no son modificables por software
                movf            AUX1,W        ;Recupero el valor del W

                bcf             INTCON,T0IF   ;Reseteo el aviso de interrupción por TMR0
                retfie                        ;Retorno habilitando las interrupciones

;/////************** PROGRAMA PRINCIPAL ******************////////////////
INICIO
                ORG             0x0100        ;Si no le asigno una posición especifica el
                                              ; ensamblador lo hace en forma automática.
                                              ;Colocándolo a continuación de la subrutina.
;------Primero seteo los puertos e interrupciones-----
                bsf             STATUS,RP0    ;Cambio al banco 1
                movlw           0x00
                movwf           TRISA
                movwf           TRISB         ;Todos los pines de los puertos serán salidas
                movlw           b'00100000'   ;
                movwf           INTCON        ;Seteo interrupciones. Solo habilito Timer0
                movlw           b'10000111'
                movwf           OPCION        ;Seteo Timer0 con pre-escalador 256
                bcf             STATUS,RP0    ;Cambio al banco 0
CARGO_TABLA
                movlw           S_A
                movwf           0x30          ;Primer dato de la tabla
                movlw           S_L
                movwf           0x31          ;Segundo dato de la tabla
```

```
                movlw       S_O
                movwf       0x32        ;Tercer dato de la tabla
                movlw       S_H
                movwf       0x33        ;Cuarto dato de la tabla
SETEO_SERIE
                movlw       ORIG
                movwf       Pdat        ;Inicializo el puntero
                movlw       0x01
                movwf       CX          ;Inicializo selector de dígito
PROGRAMA_PRINCIPAL
                bsf         INTCON,GIE  ;Habilito todas las interrupciones
LAZO
                nop                     ;El programa principal no es importante
                                        ; en este ejemplo
                goto        LAZO
                END                     ;Termino el programa con este comando
```

Otra Clases de display:
El display de puntos es en el que los caracteres se forman sobre un grupo de 5x6 o 7x12 o cualquier otra cantidad de puntos. Es el mismo procedimiento que en el display de segmentos pero con mayor cantidad de elementos y por lo tanto de filas y columnas.
Esta clase de display y el de segmentos puede ser manejada por el propio micro sin necesidad de gran cantidad de hardware externo.
El display LCD (display de cristal líquido por sus siglas en ingles) también es matricial, pero necesita una interfase especial que puede o no estar dentro del micro. Inclusive puede estar integrada al propio display.
Otros modelos de display son más complejos y su uso no es tan extendido en pequeños sistemas con microcontroladores, por ejemplo tubo CRT.

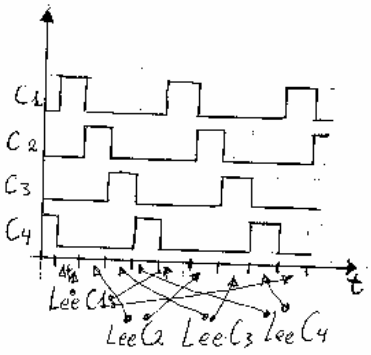
Matriz de Entrada: Matriz de Switch
En el caso de una matriz de switch, o sea una matriz de elementos de entrada o de adquisición de datos, cada interruptor se identifica con su orden de Columna y Fila (S_{CF})



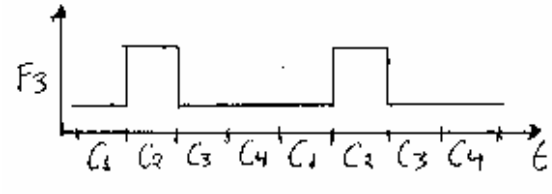
Para atender a cada columna las líneas de filas son entradas de datos.
La lectura se realiza cerca del comienzo de la ejecución de la instrucción que involucre al puerto.
Las señales de control (C_1, C_2, C_3, C_4) son activadas en forma secuencial.

En el diagrama del tiempo
Siempre se leen las líneas F_1-F_4, pero lo que significan cambia cada Δt.

En un momento representan los switch S_{11}, S_{12}, S_{13}, S_{14} y al siguiente Δt a los switch S_{21}, S_{22}, S_{23}, S_{24} y así sucesivamente. Se debe asociar cada lectura de F_1-F_4 con el valor de C_1-C_4 en ese momento, de esta manera si hay algún switch presionado podemos identificarlo. O sea nos suministra coordenadas de Filas y Columnas para cada switch.



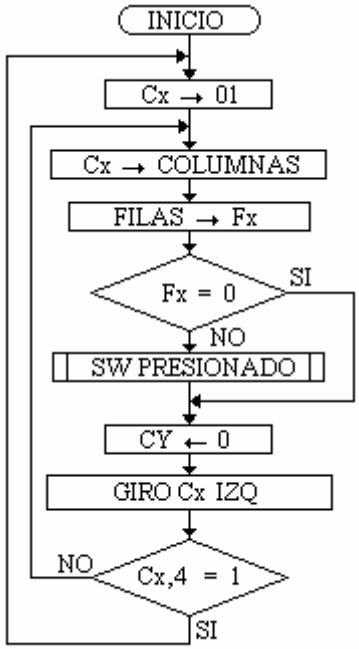
Por ejemplo, en el caso de tener presionado S_{23} , en F_1, F_2, F_4 se leen siempre en reposo para cualquier valor de C_1-C_4.
En cambio F_3, cuando C_2 sea activo copiara su nivel lógico, y valdrá lo opuesto para los momentos en que C_1, C_3 y C_4 estén activas.
Si no hubiera ningún switch presionado siempre F_1-F_4 estarían en reposo. Para el caso de S_{23} presionado el grafico de tiempo para F_3 será:
En este ejemplo consideramos activo al nivel alto.



Los valores a enviar por C1-C4 son

0001
0010
0100
1000
0001
…

Es decir la salida de un contador Johnson de 4 bit.

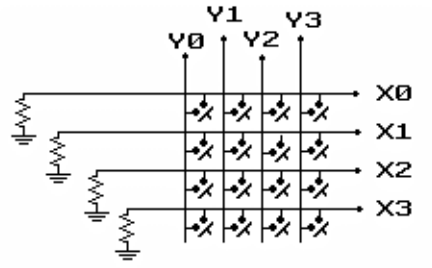


En este diagrama se usa la variable Cx como intermediaria con las COLUMNAS. Así se puede almacenar la información de con cual columna se está trabajando o leyendo.

Lo escrito en un puerto para el PIC16F84 se mantiene inalterado aunque se entre en modo SLEEP, solo se modifica si:

Se vuelve a escribir en el `puerto
Se resetea el sistema

Teclado Matricial:
Un sistema muchas veces necesita entradas de datos de un operador humano. Un método es a través de un teclado, en el cual cada tecla representa una función.



Así sea solo numérico necesitamos poder interpretar 10 teclas. Podemos intercalar un integrado que nos codifique estas 10 teclas en un numero de 4 bit. Pero lo mejor es usar el propio micro así ahorramos un IC.

Se arma la matriz de 16 switch. Colocamos 4 filas por 4 columnas.
Desde el punto de vista del PIC16F84 se nombran Y0, Y1, Y2 e Y3 a las columnas, y X0, X1, X2, X3 a las filas. Las filas tienen resistencia (pull-up) que en reposo las llevan a Vdd. Se activan las

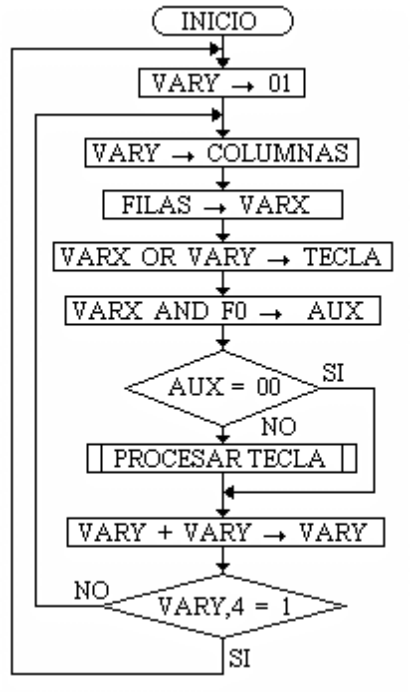
columnas llevándolas a 0 lógico. Si en esa columna hay alguna tecla presionada la fila correspondiente es llevada a 0 lógico.
La forma de escanear el teclado es activando de a una columna por vez y leyendo todas las filas. Si se aprieta de una tecla por vez solo al activar la columna correspondiente va a estar la fila de la tecla en nivel alto.
Es necesario aplicar a las columnas una secuencia parecida a un contador de desplazamiento e ir leyendo las filas cada vez que se modifica la secuencia.
Se puede hacer un reconocimiento rápido del teclado poniendo todas las columnas en alto y ver si están o no todas las filas en bajo. Si **no** están todas a 1, es que hay al menos una tecla presionada.
Para determinar cual es revisamos el teclado columna a columna.
Para ir activando las columnas de a una se genera esta secuencia

Y3	Y2	Y1	Y0
1	1	1	0
1	1	0	1
1	0	1	1
0	1	1	1

Cada vez que presentamos un valor de esta secuencia leemos las filas para detectar alguna tecla presionada.
Asociamos:

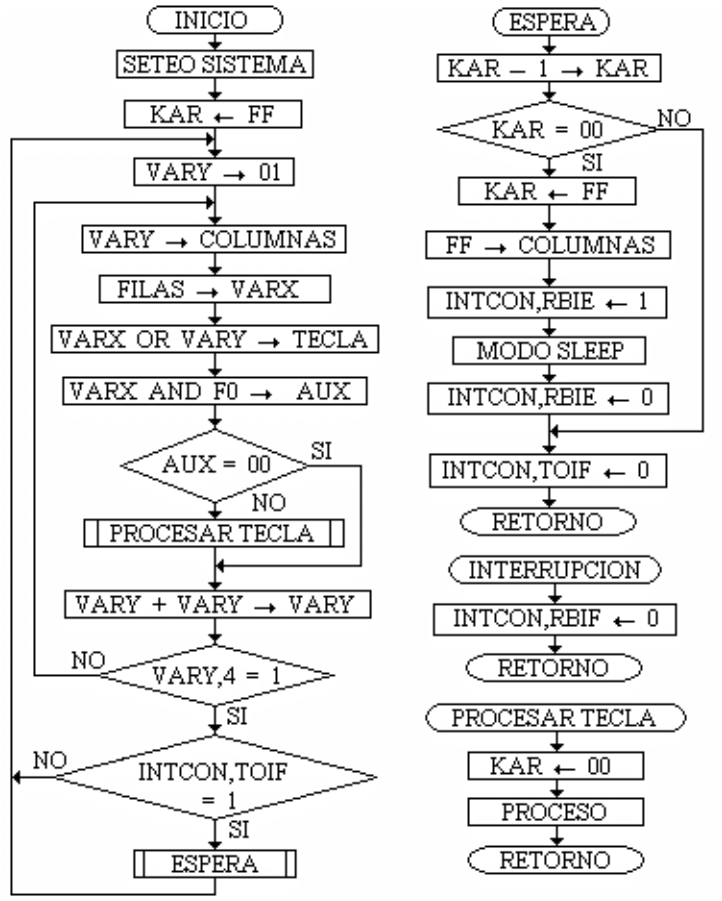
Y0 - Bit 0
Y1 - Bit 1
Y2 - Bit 2
Y3 - Bit 3
X0 - Bit 4
X1 - Bit 5
X2 - Bit 6
X3 - Bit 7

Cada coordenada de la tecla se expresa con 4 bit, o sea que podemos concatenar ambos valores en un solo byte que defina a la tecla. Siendo para cada tecla valor diferente.
Se debe tener cuidado de presionar un solo botón por vez. Si asociamos las columnas al nible bajo de un puerto y las filas al nible alto del mismo, podemos deducir este programa.



La descripción de la subrutina no es importante. Por eso no será desarrollada.
Para cambiar la columna activada, en vez de rotar los bit, sumamos el registro a si mismo. Eso nos evita usar los bits del registro STATUS. Cuando el bit 4 del registro VARY se hace 1, es que ya recorrimos todas las columnas.

Si nuestro sistema debe ahorrar energía modificaremos este programa.
Hasta que se presione alguna tecla, dejamos todas las columnas activadas y habilitamos la interrupción por cambios en RB4-RB7.
Lo ponemos en modo SLEEP y con solo presionar una tecla alcanza para sacarlo de este estado.
Antes de entrar en modo bajo consumo debemos esperar unos segundos por si se presiona alguna tecla.
Usando el timer0 podemos hacer un contador con un solo registro que nos permita esperar 10 segundos antes de entrar en modo SLEEP.
Cada vez que se detecta una tecla debemos resetear el registro KAR. Así cuando pasen 10 segundos, sin presionar una tecla, se activa la interrupción por cambios en RB4-RB7 y entramos en modo SLEEP.



Cada vez que se disminuye KAR se debe volver a habilitar el flag T0IF. Pero no se habilita su interrupción, así no genera dificultades con el resto del programa.
En cuanto se sale del modo de bajo consumo deshabilitamos la interrupción de RBIF, por no ser de utilidad hasta la próxima vez que entremos en SLEEP.

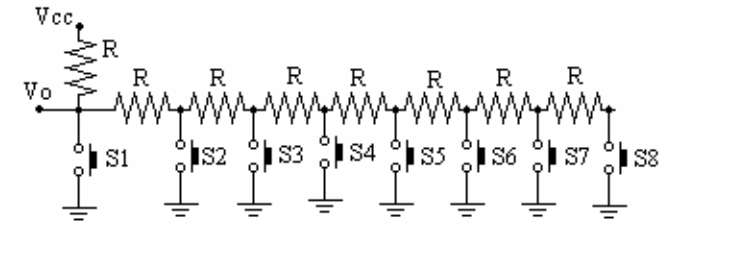
```
;Programa para el manejo de una matriz de switch
;Primero elegimos la clase de micro a usar y seteamos la palabra de configuración.
                LIST            p=16f84
                __CONFIG        0x3FF1          ;Seteo XT, No CP, No WDT, PWRTE
                __IDLOCS        0x00B0          ;Le asigno un número arbitrario
```

```
;-------Declaro las variables a usar en el programa fuente-----
W               EQU             H'0'
F               EQU             H'1'

;----- Register Files----------------------------------------------------
INDF            EQU             H'0000'
TMR0            EQU             H'0001'
PCL             EQU             H'0002'
STATUS          EQU             H'0003'
FSR             EQU             H'0004'
PORTA           EQU             H'0005'
PORTB           EQU             H'0006'
INTCON          EQU             H'000B'
OPCION          EQU             H'0081'
TRISA           EQU             H'0085'
TRISB           EQU             H'0086'

;----- STATUS Bits --------------------------------------------------------
IRP             EQU             H'0007'
RP1             EQU             H'0006'
RP0             EQU             H'0005'
TO              EQU             H'0004'
PD              EQU             H'0003'
Z               EQU             H'0002'
DC              EQU             H'0001'
CY              EQU             H'0000'

;----- INTCON Bits --------------------------------------------------------
GIE             EQU             H'0007'
EEIE            EQU             H'0006'
T0IE            EQU             H'0005'
INTE            EQU             H'0004'
RBIE            EQU             H'0003'
T0IF            EQU             H'0002'
INTF            EQU             H'0001'
RBIF            EQU             H'0000'

;----- OPTION Bits --------------------------------------------------------
RBPU            EQU             H'0007'         ;Habilitación de resistencias pull-up en PORTB
INTA            EQU             H'0006'         ;Selector de flanco activo de RB0
TOCS            EQU             H'0005'         ;Selector de fuente para TMR0
TOSE            EQU             H'0004'         ;Flanco de activación de RA4/TOCKI
PSA             EQU             H'0003'         ;Destino del pre-escalador
PS2             EQU             H'0002'         ;Selectores del valor del pre-escalador
PS1             EQU             H'0001'
PS0             EQU             H'0000'

;------------Declaro mis propias variables, constantes y etiquetas-----------
VARY            EQU             0x20
VARX            EQU             0x21
TECLA           EQU             0x22


                ORG             0x000
INICIO
                goto            PRINCIPAL               ;Salto por encima del comienzo
                                                        ;de las subrutinas e interrupciones
                ORG             0x004
INTERRUPCIONES
                bcf             INTCON,RBIF             ;Limpio aviso de interrupción
                retfie
;------Retorno al Programa principal------
```

```
PROCESAR_TECLA
                clrf        KAR             ;Al restarle uno queda en FF
                nop                         ;No la desarrollo pues no me interesa
                                            ;en este ejemplo
                return
;------------Retorno de procesar la tecla detectada

;----------------------Desarrollo la subrutina de 10 segundos sin apretar tecla
ESPERA
                decfsz      KAR,F           ;disminuyo contador de 10 segundos
                goto        VOLVER_ESP
                movlw       0xFF
                movwf       KAR             ;Reinicio contador de 10 segundos

VOLVER_ESP
                bcf         INTCON,T0IF     ;Reseteo aviso de desborde de TMR0
                return
PRINCIPAL
                bsf         STATUS,RP0      ;Cambio al banco 1
                movlw       0xF0
                movf        TRISB           ;Nible bajo salidas y nible alto entradas
;-------Seteo interrupciones y TIMER0---------

                bcf         STATUS,RP0      ;Cambio al banco 0

LOOP1
                movlw       0x01
                movf        VARY            ;Inicializo selector de columna
LOOP2
                movwf       VARY,W
                movf        PORTB           ;Activo la columna seleccionada
                movwf       PORTB,W         ;Leo las filas.
                movf        TECLA           ;Guardo la coordenada de tecla presionada
                andlw       0xF0            ;borro el nible bajo así queda VARX
                btfss       STATUS,Z        ;Verifico si VARX vale cero
TECLA
                call        PROCESAR_TECLA  ;Si no vale la proceso
NO_TECLA
                movwf       VARY,W
                addwf       VARY,F          ;Desplazo los bit hacia la izquierda
                btfss       VARY,.4         ;Verifico si active todas las columnas
                goto        LOOP2

                btfss       INTCON,T0IF     ;Si desbordo el TMR0 voy a ESPERA
                goto        LOOP1
                call        ESPERA          ;Cuento si pasaron 10 segundos
                goto        LOOP1

                END                         ;Al terminar debo incluir una línea vacía.
```

Otra clase de Teclado

Otra clase de matriz de switch, aunque no se ajuste al nombre es la asociada con un conversor A-D. Se usa en microcontroladores con dicho dispositivo incluido en el chip.
Si contamos con una línea con conversor Analógico-Digital podemos armar este esquema



Cada tecla presionada entrega un valor diferente de tensión. El conversor A-D entrega por lo tanto un valor binario diferente.