USO DEL TECLADO PC-AT COMO PERIFÉRICO DEL MICROCONTROLADOR PIC16F84

Ing. Daniel Francisco Gómez Prado
dgomezp@unmsm.edu.pe

Profesor de la Facultad de Ingeniería Electrónica, Universidad Nacional Mayor de San Marcos
Lima - Perú

RESUMEN: El presente artículo explica un pequeño programa para el microcontrolador PIC16F84, que detecta las teclas presionadas de un teclado PC-AT. El tema es de utilidad para aquellos quienes deseen aclarar principios sobre el manejo de interrupciones, tablas y adquisición de datos y a partir de aquí poder realizar proyectos más elaborados.

ABSTRACT: This paper explains a small program for the microcontroller PIC16F84, which detects the pressed keys from a keyboard PC-AT. The topic is of utility for those who want to clarify concepts about the handling of interruptions, tables and acquisition of data.

Palabras Claves: Teclado, PC-AT, Microcontrolador PIC16F84.

I. TEORÍA DEL TECLADO PC-AT

El teclado es un periférico bidireccional que se conecta al ordenador por medio de un cable que contiene 4 hilos hábiles: uno de alimentación, uno de tierra, uno para datos y otro para reloj. Los conectores externos del teclado más conocidos se muestran en la Figura 1, a continuación:

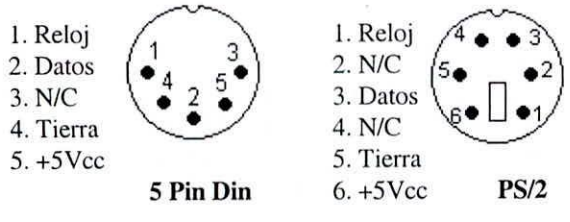

Figura 1. Conectores de Teclado PC-AT estándar

Donde las líneas del reloj y de datos son bidireccionales de E/S con colector abierto capaces de drenar una corriente de hasta de 300mA.

Después del encendido y al recibir alimentación, el teclado realiza un autodiagnóstico denominado BAT (*Basic Assurance Test)* donde chequea su ROM, RAM y enciende y apaga todos los LED. Esta operación emplea entre 600 y 900 milisegundos; al acabar el BAT y cuando sea posible establecer la comunicación con el host (líneas de reloj y datos en alto) envía un byte 0AAh si todo ha salido bien y un 0FCh si ha habido fallos; inicializando después los parámetros de auto repetición de las teclas.

1.1 Comandos del Teclado

Además de poder obtener el código de las teclas presionadas, también se puede enviar comandos desde y hacia el teclado, a continuación se detallan las funciones de estos comandos. [*Konzak, 1993*]

1.1.1 Comandos del Host al teclado

Los comandos desde el host hacia el teclado pueden ser enviados en cualquier momento, a menos que esté esperando por un byte de datos en el registro de entrada como consecuencia de un comando previo, cuando se envía un comando al teclado, éste responderá en menos de 20 milisegundos devolviendo una señal de reconocimiento por medio de un byte 0FAh. Los principales comandos (diferenciados de los datos por tener el bit 7 activo) son:

- Reset (0FFh): Al recibirlo se envía una señal de reconocimiento y se asegura que el host, se de por enterado poniendo en alto las líneas de reloj y datos un mínimo de 500 microsegundos; el teclado permanece inhibido hasta que el host acepte la señal de reconocimiento o envía otro comando que sobreescriba y anule éste. A continuación, el teclado ejecuta de nuevo el BAT, estableciendo valores por defecto para la autorepetición y limpiando su registro de salida.

- Reenvío (0FEh): El host puede enviar este comando cuando detecta un fallo en la recepción. Este comando sólo puede ser enviado después de una transmisión del teclado y antes de habilitar la comunicación para la siguiente recepción, el cual responde enviando de nuevo el dato anterior.
- Establecer valores por defecto (0F6h): Devuelve la autorepetición a los valores habituales, limpia su registro de salida y continúa rastreando las teclas si no estaba inhibido; es una especie de *reset en caliente*.
- Establecer valores por defecto y parar (0F5h): Similar al comando anterior, pero dejando de rastrear las teclas y permaneciendo inhibido hasta recibir más instrucciones.
- Habilitar (0F4): Reanuda el funcionamiento interrumpido por el comando anterior.
- Establecer el *ratio* y retardo de autorepetición (0F3h): tras este comando debe enviarse otro inmediatamente a continuación, que se interpretará como dato, estableciendo los valores de autorepetición. De este segundo byte, el bit 7 estará siempre en cero; el valor de los bits 5 y 6, al sumarles una unidad, indican el tiempo que ha de pasar desde que se pulsa una tecla hasta que comience autorepetirse, en unidades de 0,25 segundos (±20%). Los bits 2, 1 y 0 forman un número A; los bits 4 y 3 forman otro número B; por medio de la fórmula (1) se obtiene la tasa o ratio de autorepetición en teclas por segundo:

$$\frac{1}{0.00417(8+A)2^{B}} \qquad (1)$$

  Una vez recibido este comando, el teclado envía la acostumbrada señal de reconocimiento, deja de rastrear las teclas y espera por el parámetro de autorepetición, respondiendo al mismo con otra señal de reconocimiento y volviendo a ejecutar lo anterior. Si en lugar de recibir el parámetro recibe otro comando (bit 7 activo) dejará inalterados los valores de autorepetición y procesará dicho comando, aunque permanecerá inhibido hasta que se le habilite con el comando 0F4h. Por defecto, el sistema establece una tasa de 10 caracteres por segundo y 0,5 segundos de espera.
- Sin operación (0F7h a 0FDh y 0EFh al 0F2h): Son códigos reservados; el teclado al recibirlos envía la señal de reconocimiento de siempre y no realiza ninguna acción.
- Eco (0EEh): Si el teclado recibe este comando, lo reenvía a continuación. Es una ayuda al diagnóstico.
- Encender/apagar los LED (0EDh). Tras este comando se ha de enviar otro byte de datos, cuyos bits 0, 1 y 2 están ligados al estado de los LED de Scroll Lock, Num Lock y Caps Lock, respectivamente; los demás están reservados. Al recibir el comando envía la correspondiente señal de reconocimiento y deja de rastrear las teclas, esperando por el dato. Si en vez de un dato recibe otro comando, dejará intactos los LED, procesará dicho comando y continuará rastreando las teclas (sin quedar inhibido en esta ocasión).

1.1.2 Comandos del Teclado al Host

A continuación se listan los comandos que el teclado puede enviar al host en un momento dado.

- Reenvío (0FEh): El teclado puede enviar este comando al host para solicitar el reenvío cuando detecta un fallo en la recepción (normalmente de paridad) o una entrada incorrecta.
- Reconocimiento ó *ACK* (0FAh): El teclado devuelve este valor para indicar al host que ha recibido algún dato (excepto en el caso de los comandos Eco y Reenvío del host).
- Desbordamiento (0): Cuando el host intenta leer el teclado directamente sin haber códigos en el buffer interno del propio teclado.
- Fallo en el diagnóstico (0FDh): El teclado periódicamente se auto chequea y envía este código si detecta algún fallo. Si el fallo sucede durante el BAT, dejará de rastrear las teclas en espera de un comando del host; en cualquier otro momento continuará rastreando las teclas.
- Código de tecla soltada ó *break code* (0F0h): El teclado envía este código al host para indicar que el siguiente código que enviará a continuación corresponderá a una tecla soltada.
- BAT completado (0AAh): Después de realizar el BAT el teclado envía un 0AAh para indicar que ha salido bien, o un 0FCh (u otro valor) si ha habido fallos.
- Respuesta al eco (0EEh): El teclado reenvía este valor al host en respuesta de haberlo recibido.

1.2 Código de Escaneo

Al pulsar una tecla se genera un único código, denominado también código de rastreo, el cual identifica la tecla pulsada y la envía a través de su línea de data serial. Al soltar la tecla, los teclados de AT generan dos códigos que se envían consecutivamente (0F0h y después el mismo código que al pulsarla). Por ejemplo, si se pulsa la tecla 'A' aparecerá en la línea de dato serial del teclado el byte 1Ch, y al soltar la 'A' se generaran el código F0h y luego 1Ch. Si una tecla lleva cierto tiempo pulsada, el teclado se encarga de repetir su código, en el conocido mecanismo autorepetición de la mayoría de los teclados.

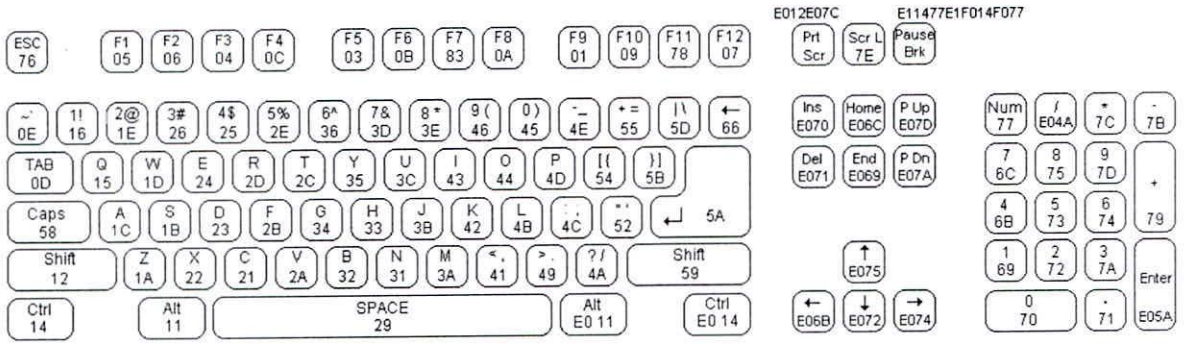

Figura 2. Teclado Extendido del PC-AT

Las teclas CTRL, ALT y SHIFT pueden ser pulsadas para modificar el resultado de la pulsación de otras, y estas teclas al igual que Num Lock, Caps Lock, Scroll Lock e Ins no poseen la característica de autorepetición de las demás teclas.

En la figura 2, se muestra el teclado básico del PC-AT de 83/84 teclas y sus códigos de escaneo, junto con el juego de teclas expandidas las cuales han sido añadidas al teclado estándar y tienen un comportamiento especial ya que pueden generar hasta 4 interrupciones consecutivas con objeto de emular ciertas combinaciones de las teclas no expandidas.

Se observa que el código ASCII 0E0h sólo es generado en los teclados expandidos por las teclas expandidas. Así, por ejemplo, cuando está inactivo NUM LOCK y se pulsa el cursor derecho, expandido se generan dos interrupciones consecutivas: en la primera aparece un valor 0E0h en el puerto del teclado que indica que es una tecla expandida; en la segunda interrupción aparece el valor 74h: el mismo que hubiera aparecido pulsando el '6' del teclado numérico. Así, las teclas exclusivas de teclados expandidos generan los mismos códigos de rastreo que sus correspondientes teclas «no expandidas», aunque precedidos de un código de rastreo adicional 0E0h como mínimo.

1.3 El Protocolo del Teclado PC-AT

Como se ha mencionado anteriormente, el teclado tiene implementado un protocolo bidireccional, es decir puede enviar datos al host así como recibir datos del host. Teniendo el host la ultima prioridad sobre la dirección del bus [Interfacing, 2001].

1.3.1 Teclado a Host

El teclado es libre de transmitir hacia el host mientras ambas líneas del reloj del teclado y Datos permanezcan en alto (*idle*). En este momento la transmisión de datos hacia el host se realiza a través de un protocolo de comunicación en serie que en el AT consta de un marco de 11 bits, el primer bit de inicio, los 8 siguientes de datos (el LSB primero), 1 bit de paridad impar[1] y el último de fin ó parada. En el diagrama de la Figura 3 se representa un byte de datos transmitidos desde el teclado. El teclado no cambia necesariamente el dato en la línea en el flanco de subida del

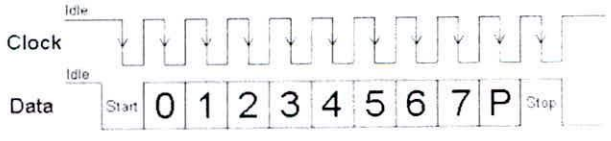

Figura 3. Transmisión del teclado al host

Reloj como se muestra aquí, pero el dato en la línea debe ser siempre valido en el flanco de bajada del reloj. Así, cada bit debe ser leído en el flanco de bajada del reloj. Teniendo en cuenta que el bit menos significativo siempre es enviado primero. El teclado genera la señal de reloj que por lo general esta entre 20khz y 30khz.

1.3.2 Host a Teclado

El protocolo de host a teclado es iniciado tomando la línea de datos a cero lógico, sin embargo para evitar que ambos teclado y host traten de utilizar la línea de datos al mismo tiempo, la línea de reloj del teclado se lleva a cero lógico y se mantiene en cero por más de 60us, el cual es un tiempo mayor al ancho de un bit, asegurando así que el teclado no transmita datos. Después la línea de datos del teclado es llevado a nivel bajo mientras que la línea del reloj es soltada y el teclado se prepara para aceptar comandos del host.

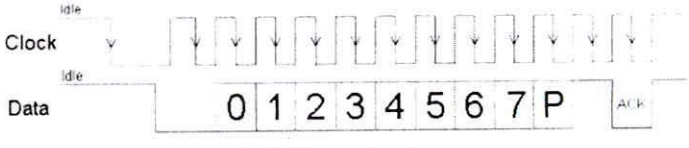

Figura 4. Transmisión del host al teclado

[1] El bit de paridad impar, es aquel agregado de forma tal que en los 9 bits totales (8bits dato y el bit agregado) hay un número impar de unos.

Cuando el teclado está listo para aceptar comandos del host empieza a generar una señal de reloj, cuya generación puede demorar hasta 10ms. Cuando se genera la señal de reloj después de que el primer flanco de bajada es detectado se puede poner el primer bit (el bit menos significativo) en la línea de datos, el cual será leído por el teclado en el próximo flanco de bajada. Este proceso se repite para los 8 bits de datos, después de esto el teclado se prepara para recibir el bit de paridad impar[1]; luego del cual la línea de datos debe mantenerse en 1 durante un ciclo de reloj. Al recibir este último bit 1 de parada, el teclado confirma que ha recibido el comando llevando la línea de datos a cero en el siguiente flanco de bajada del reloj.

Si la línea de datos no se lleva a 1 después del bit de paridad, el teclado continuará enviando la señal de reloj hasta que la línea de datos sea llevada a 1.

Si se está trasmitiendo el código de rastreo de una tecla, el host mantiene un cero en la línea de datos, el teclado guardará el valor del código en transmisión en un buffer hasta que la línea de datos sea soltada (llevada a 1), reanudando luego su transmisión.

II. PROGRAMA DE COMUNICACIÓN CON EL MICROCONTROLADOR PIC

El programa, en este caso, se debe encargar de detectar las interrupciones generadas por el teclado, determinar la tecla pulsada y hallar su valor equivalente en código ASCII.

Este programa se limitará a recibir los códigos de las teclas pulsadas. Para ello debemos leer el código binario producido en los flancos de bajada del reloj, esto se realiza colocando el reloj del teclado como la entrada de la interrupción RB0, y cada vez que esta se activa leemos en RB1 un bit del dato. El análisis de esta etapa se puede realizar en los módulos que se detallan a continuación.

2.1 Módulo de Detección de la Tecla Pulsada

Este modulo se encarga de atender las interrupciones generadas por el teclado a través de las líneas KBD_DATA y KBD_CLK. De esta manera se tendrá una interrupción por cada bit transmitido por el teclado, incrementando un contador de Nbits, cuando este contador llega a 9, los últimos 8 bits recibidos son almacenados en un registro KeyRtn y se esperan las dos ultimas interrupciones generadas por el protocolo de comunicación del teclado correspondientes al bit de paridad impar y bit de *stop*. Una vez recibido estos dos últimos bits el registro contador Nbits llega a 11 y se activa el registro Nbytes, indicando con ello que se a obtenido el código completo de una tecla pulsada.

En el diagrama de la Figura 5, se observa el uso de un registro No_Rotar, este es utilizado por que los bits que se van leyendo del teclado empiezan por el menos significativo y son almacenados en el bit más significativo de Datain, por ello mientras No_Rotar esta activo, 1 lógico,

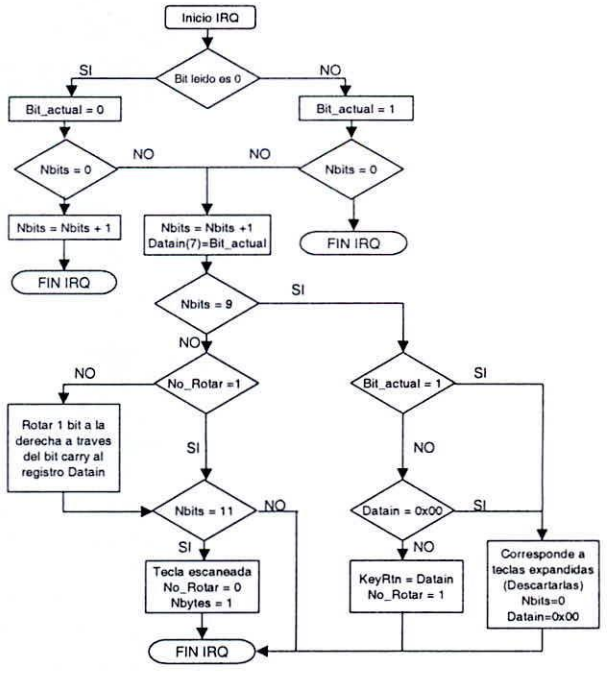

Figura 5. Diagrama de Flujo de la Interrupción

se almacena en el bit actual de Datain (7) el cual se desplaza 1 bit a la derecha. De esta manera al leer el último bit del dato se han realizado 7 rotaciones al bit menos significativo y el dato almacenado en Datain es correcto. El código completo del teclado se almacena en Datain en el 9no bit transmitido, es decir cuando Nbits = 9, por ello en este punto se guarda el registro Datain en KeyRtn y se desactiva la rotación del registro Datain, No_Rotar = 0.

2.2 Módulo de determinación del equivalente ASCII de la Tecla Pulsada

Cuando Nbytes se activa a 1, se sale del lazo de espera de tecla presionada y se procede a hallar el equivalente ASCII del código guardado en KeyRtn. Para simplicidad del programa se descarta todas las teclas que no pertenezcan al teclado base, es decir aquellas menores de 0x0E y mayores de 0x67. Si la tecla presionada esta entre ese rango entonces se revisa el estado del bit CAPS, si éste está activo se utiliza la tabla de mayúsculas para hallar el equivalente ASCII de la tecla pulsada, en caso contrario se utiliza la tabla de minúsculas. Una vez determinado el equivalente ASCII de la tecla su valor es almacenado en el registro ASCII.

Si se está determinando el equivalente ASCII de una tecla se pulsará nuevamente el teclado, entonces se activará la interrupción, y los registros *Status* y W son salvados al inicio de la IRQ, por lo que al final de cada IRQ estos registros no

han variado continuando con la ejecución del programa donde habían sido interrumpido.

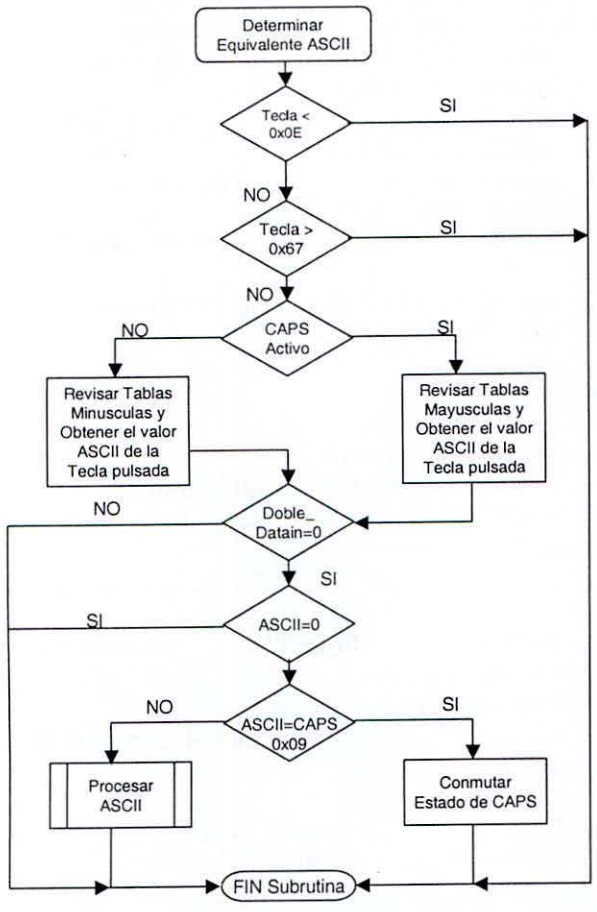

Figura 6. Diagrama de Flujo del equivalente ASCII

En el manejo de las tablas Mayúsculas y Minúsculas hay que tener en cuenta que los microcontroladores PIC16F84 poseen 1K de memoria de programa, agrupada en 4 bancos de 0x0FF a 0x3FF. Así cada banco de memoria es direccionado internamente mediante el registro PC que corresponde al byte inferior de direcciones; y la selección del banco se realiza con los dos bits superiores mediante el registro PCLATH. Por ello, cuando se trabaja con tablas hay que tener especial cuidado debido a que los saltos sólo varían el registro PC, pero no el PCLATH, pudiendo ocasionar un error en la secuencia de ejecución que no es detectado en la compilación del programa. El programa evita este problema (ver Figura 6).

2.3 Circuito

A continuación se muestra el circuito esquemático a implementar (ver Figura 7). En donde los 4 bits RB4-RB7 pueden ser utilizados para enviar el código ASCII en dos *nibles* a una pantalla LCD con RB2 y RB3 como sus señales de control; y los 4 bits restantes RA1-RA5 para activar o desactivar relés.

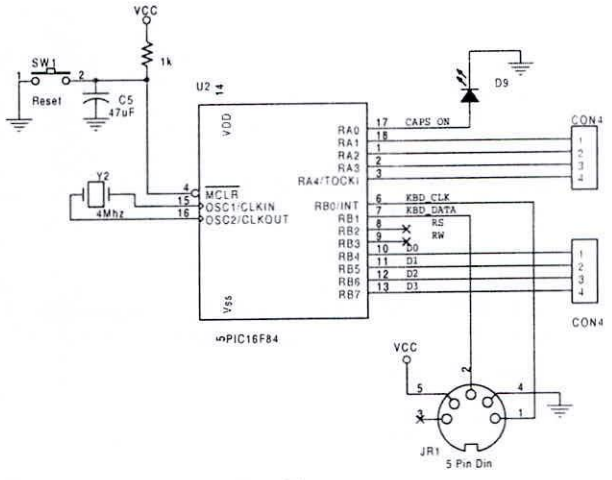

Figura 7. Circuito Esquemático

2.5 *Programa del Microcontrolador*

A continuación se muestra el código fuente del programa, obsérvese que además de la instrucción List para indicar al compilador el microcontrolador a utilizar, se debe añadir una palabra de configuración que indique al microcontrolador entre otras cosas el tipo de reloj que se esta utilizando, en nuestro caso de cristal. Este último es necesario si se desea grabar el archivo *hex* generado en el microcontrolador.

```
        LIST P=16F84

;*************************************************
; Control del teclado PC AT
;
;
;
;    PORTB  PORTB  PORTB  PORTB  PORTB  PORTB
;
; Salida de Datos:   RB7 - RB4
;
; :                  RB3
;
; :                  RB2
;
; KBD_DATA:      RB1  Datos sereal del teclado
;
; KBD_CLK:       RB0  Irq generada por el teclado
;
;
;    PORTA  PORTA  PORTA  PORTA  PORTA  PORTA
;
; :                  RA4
;
```

```
*  :                    RA3
*
*  :                    RA2
*
*  :                    RA1
*
*  CAPSON :     RA0  Led indicador de Caps
*
*
*
*  Por:     Daniel Gomez Prado, dgomezp@unmsm.edu.pe*
************************************************
        include <p16f84.inc>

Palabra de Configuracion

__CONFIG   _CP_OFF & _WDT_OFF & _PWRTE_ON &
_XT_OSC

PUERTOS
Doble_DATAIN equ            0x14
ASCII        EQU            0X15
BANDERAS     EQU            0X16
SALTO        EQU            0X17
CONTADOR1    EQU            0X18
CONTADOR2    EQU            0X19
#DEFINE      CAPSON         PORTA,0    ;RA0
#DEFINE      BANDERAS_CAPS      BANDERAS,0
             ;BIT0
Registros del Teclado PC
Nbits        equ    0x0c    ;guarda numero de bits
de la tecla pulsada
Nbytes       equ    0x0d    ;guarda numero de bytes
NO_ROTAR     equ    0x0e    ;guarda          #bytes,
esperando bit de P y Stop
KeyRtn       equ    0x12    ;regresa  el  valor  a
visualizar
DATAIN       equ    0x13    ;almacena        valor
ingresado por KBD DATA
#DEFINE      KBD_CLK        PORTB,0    ;RB0

#DEFINE      KBD_DATO       PORTB,1    ;RB1
#DEFINE      BIT_ACTUAL     DATAIN,7
Registros temporales para PUSH y POP
STATUS_TEMP equ     0x0f    ;\ de la rutina
W_TEMP       equ    0x11    ;/ de IRQ
INICIO DE LA MEMORIA DE PROGRAMA
        org     0x00    ;Vector de reset
        goto    INICIO
        org     0x04    ;Inicio del Vector de     IRQ
MOVWF       W_TEMP              ;\
SWAPF STATUS,W      ; > PUSH, SALVA REGISTRO
MOVWF STATUS_TEMP       ;/   STATUS Y W
CALL     LEER_TECLADO
bcf      INTCON,INTF   ; Limpia bandera de   IRQ
SWAPF STATUS_TEMP,W     ;\
MOVWF       STATUS          ;\ POP, RECUPERA
SWAPF W_TEMP,F      ;/ REGISTROS STATUS Y W
SWAPF W_TEMP,W      ;/
RETFIE              ; Retorno de IRQ
INICIO ;******>>
        clrf    PORTB           ; Borra salidas
        clrf    PORTA           ; Borra salidas
        bsf     STATUS,RP0      ; Selecciona banco 1
        movlW   b'11111111'
        movWf   TRISB           ; ´1´ -> In , ´0´ -> Out
        movlW   b'00000000'
        movWf   TRISA           ; Todo RA  Out
        movlW   b'10111111'
        movWf   OPTION_REG  ;Establecer  IRQ  flanco
bajada
        bcf     STATUS,RP0      ;Selecciona banco 0
        clrf    BANDERAS
        clrf    Nbits
        clrf    Nbytes
        clrf    DATAIN
        clrf    NO_ROTAR
        movlW   b'10010000'     ;IRQ RB0/INT
        movWf   INTCON          ;Activa GIE global
LAZO_LECTURA ;******>>
        CLRWDT              ;Espera  que  se  genere
IRQ
        DECFSZ      Nbytes,W        ; Debido a la
pulsacion de una tecla
GOTO    LAZO_LECTURA
MOVF    KeyRtn,W        ;Tecla Precionada al registro W
sublW   0x0D            ;\
btfsc   STATUS,C        ; |
goto    LIMPIAR         ; < 0Eh | Desactiva
movf    KeyRtn,W        ;\ Teclas
sublW   0x67            ; / no
btfsc   STATUS,C        ;| necesarias
goto    SIGUE           ;|
goto    LIMPIAR         ;>= 67h /
                        ;Solo interesa las teclas entre 0EH
                        Y 67H
MOVLW       0X0E
SUBWF KeyRtn,W      ; el salto a realizar en la tabla esta
en W
BTFSC  BANDERAS_CAPS        ;  ¿¿  ESTA  CAPS
ACTIVO ??
GOTO   TBL_MAY              ; CAPS=1
TBL_MIN ;******>>
MOVWF       SALTO           ; AQUI SALTO=W
MOVLW       high TABLA1     ;TABLA              DE
MINUSCULAS   CAPS=0
MOVWF       PCLATH          ; RUTINA TABLAS EN
LA FRONTERA
MOVLW       TABLA1+1        ; 0X0FF -> 0X100
ADDWF       SALTO,W
BTFSC  STATUS,C
INCF   PCLATH,F
CALL   TABLA1               ;TABLA              DE
MINUSCULAS CAPS=0
goto    FIN_TABLA
TBL_MAY ;******>>
```

```
MOVWF   SALTO       ;AQUI SALTO=W
MOVLW   high TABLA0 ;TABLA DE
MAYUSCULAS CAPS=1
MOVWF   PCLATH      ;RUTINA TABLAS EN
LA FRONTERA
MOVLW   TABLA0+1    ;0X0FF -> 0X100
ADDWF   SALTO,W
BTFSC STATUS,C
INCF  PCLATH,F
CALL  TABLA0        ;TABLA DE MAYUSCULAS
CAPS=1

FIN_TABLA ;******>>

incf    Doble_DATAIN,1      ;\
btfsc   Doble_DATAIN,0      ;> Para eliminar Break
Code
goto    LIMPIAR     ;/
btfsc   STATUS,Z    ;\ ASCII es cero => NONE
goto    LIMPIAR     ;/
MOVWF   ASCII   ;'W' tiene VALOR 'ASCII' DE
TABLAS
xorlW 0xA9          ;ES ASCII CAPS o SHIFT?
btfss   STATUS,Z
GOTO  CARACTER_ASCII  ; No es CAPS NI Shift
COMF  BANDERAS,W
andlW 0x01          ;W,BIT 0 IGUAL A 1 SI CAPS
activo
MOVWF   BANDERAS
BTFSC BANDERAS_CAPS     ;¿¿   ESTA   CAPS
ACTIVO ??
BSF     CAPSON          ;CAPS=1     ACTIVO
PRENDE LED
BTFSC BANDERAS_CAPS     ;
goto    LIMPIAR         ;si era tabla mayusculas
BCF   CAPSON            ; CAPS=0 APAGA LED
GOTO  LIMPIAR

CARACTER_ASCII ;******>>

NOP             ;aquí ya se tiene la tecla presionada en
ASCII
NOP             ;pudiéndose mostrar en una pantalla LCD,
o con
NOP             ;registros tipo FIFO se pueden almacenar
todo
NOP             ;el texto deseado.

LIMPIAR ;******>>

NOP
CLRF  KeyRtn
CLRF  ASCII
CLRF  SALTO
CLRF  Nbytes
GOTO  LAZO_LECTURA      ;Aquí termina el
procesamiento de una; tecla y se inicia de nuevo la espera de
que se pulse otra tecla  Lazo__Cerrado

*************************************************
* FUNCION DE IRQ DEL TECLADO                    *
* REALIZA LA LECTURA SERIAL DE LA TECLA PULSD   *
*************************************************

LEER_TECLADO ;******>>
BTFSC KBD_DATO
goto    KBD_ES1     ; dato de KBD DATA es uno
bcf     BIT_ACTUAL  ; ES CERO, --> DATAIN<7> ==
0
goto    SEGUIR
KBD_ES1 ;******>>
bsf     BIT_ACTUAL  ; ES UNO, ---> DATAIN<7> ==
1
SEGUIR ;******>>
incf    Nbits,F     ; 1ro: W = 1, Nbits = 0
DECFSZ      Nbits,W
goto    BIT_REAL    ; diferente a 1, no es BIT START
BTFSS BIT_ACTUAL    ; revisa si bit START es cero
RETURN              ; es cero
goto    CLEANUP     ; limpiar contador
BIT_REAL
movf    Nbits,W
XORLW       0x09            ;si W = '9', Z=1
btfsc   STATUS,Z    ;si W <> '9', Z=0
goto    ES_9NO_BIT
NO_ES_9NO_BIT ;******>>
decfsz  NO_ROTAR,W  ;decf   NO_ROTAR,W
RRF   DATAIN,1
movf    Nbits,W
XORLW       0x0B            ;si W = '11', Z=1
btfsc   STATUS,Z    ;si W <> '11', Z=0
goto    aceptar_byte    ;Nbits = 11
RETURN
ES_9NO_BIT ;******>> ;Ahora tenemos 8 bits del Scan
Code,
QUE TENEMOS?
BTFSC BIT_ACTUAL    ;va a clenaup
goto    CLEANUP     ;si es break code.
movf    DATAIN,W    ;TECLA ACEPTADA
btfsc   STATUS,Z    ;si W <> '0', Z=0
goto    CLEANUP     ;son iguales
MOVWF   KeyRtn      ;No es 0 Retornar caracter
MOVLW       0x01
MOVWF       NO_ROTAR    ;BYTE           SIN
COMPLETAR
RETURN
aceptar_byte ;******>>
MOVLW       0X01
MOVWF   Nbytes      ;DATAIN to process
clrf    NO_ROTAR
CLEANUP ;******>>
CLRF  Nbits         ;clear # bits
CLRF  DATAIN
RETURN
*************************************************
Tabla MAYÚSCULAS                                *
```

```
*********************************************
TABLA0 ;******>>
MOVWF       PCL     ;PARA  TABLAS  mAYusculas
EN LA FRONTERA
    retlW   '~'       ; code 0E, CON mayus
    retlW    0        ; code 0F
    retlW    0        ; code 10
    retlW    0        ; code 11 ALT  IZQ
    retlW   0xA9      ; code 12 SHIFT IZQ
    retlW    0        ; code 13
    retlW    0        ; code 14 CTRL  IZQ
    retlW   'Q'       ; code 15
    retlW   '!'       ; code 16
    retlW    0        ; code 17
    retlW    0        ; code 18
    retlW    0        ; code 19
    retlW   'Z'       ; code 1A
    retlW   'S'       ; code 1B
    retlW   'A'       ; code 1C
    retlW   'W'       ; code 1D
    retlW   '@'       ; code 1E
    retlW    0        ; code 1F
    retlW    0        ; code 20
    retlW   'C'       ; code 21
    retlW   'X'       ; code 22
    retlW   'D'       ; code 23
    retlW   'E'       ; code 24
    retlW   '$'       ; code 25
    retlW   '#'       ; code 26
    retlW    0        ; code 27
    retlW    0        ; code 28
    retlW   ' '       ; code 29
    retlW   'V'       ; code 2A
    retlW   'F'       ; code 2B
    retlW   'T'       ; code 2C
    retlW   'R'       ; code 2D
    retlW   '%'       ; code 2E
    retlW    0        ; code 2F
    retlW    0        ; code 30
    retlW   'N'       ; code 31
    retlW   'B'       ; code 32
    retlW   'H'       ; code 33
    retlW   'G'       ; code 34
    retlW   'Y'       ; code 35
    retlW   '^'       ; code 36
    retlW    0        ; code 37
    retlW    0        ; code 38
    retlW    0        ; code 39
    retlW   'M'       ; code 3A
    retlW   'J'       ; code 3B
    retlW   'U'       ; code 3C
    retlW   '&'       ; code 3D
    retlW   '*'       ; code 3E
    retlW    0        ; code 3F
    retlW    0        ; code 40
    retlW   '<'       ; code 41
    retlW   'K'       ; code 42
    retlW   'I'       ; code 43
    retlW   'O'       ; code 44
    retlW   ')'       ; code 45
    retlW   '('       ; code 46
    retlW    0        ; code 47
    retlW    0        ; code 48
    retlW   '>'       ; code 49
    retlW   '?'       ; code 4A
    retlW   'L'       ; code 4B
    retlW   ':'       ; code 4C
    retlW   'P'       ; code 4D
    retlW   '_'       ; code 4E
    retlW    0        ; code 4F
    retlW    0        ; code 50
    retlW    0        ; code 51
    retlW   0x22      ; code 52  "
    retlW    0        ; code 53
    retlW   '{'       ; code 54
    retlW   '+'       ; code 55
    retlW    0        ; code 56
    retlW    0        ; code 57
    retlW   0xA9      ; code 58       CAPS
    retlW   0xA9      ; code 59       SHIFT DER
    retlW   0x0A      ; code 5A       ENTER
    retlW   '}'       ; code 5B
    retlW    0        ; code 5C
    retlW   '|'       ; code 5D
    retlW    0        ; code 5E
    retlW    0        ; code 5F
    retlW    0        ; code 60
    retlW    0        ; code 61
    retlW    0        ; code 62
    retlW    0        ; code 63
    retlW    0        ; code 64
    retlW    0        ; code 65
    retlW   0x08      ;ES BACK SP   code 66
    retlW    0        ; FIN
*********************************************
* Tabla MINUSCULAS                           *
*********************************************
TABLA1 ;******>>
MOVWF       PCL     ;PARA  TABLAS  minusculas
EN LA FRONTERA
    retlw   0x60      ; 0E code
    retlw    0        ; 0F code
    retlw    0        ; 10 code
    retlw    0        ; 11 code      ALT  IZQ
    retlw   0xA9      ; 12 code      SHIFT IZQ
    retlw    0        ; 13 code
    retlw    0        ; 14 code      CTRL  IZQ
    retlw   'q'       ; 15 code      q
    retlw   '1'       ; 16 code      1!
    retlw    0        ; 17 code
    retlw    0        ; 18 code
    retlw    0        ; 19 code
    retlw   'z'       ; 1A code            z
    retlw   's'       ; 1B code      s
    retlw   'a'       ; 1C code      a
    retlw   'w'       ; 1D code            w
```

```
retlw   '2'          ; 1E code                2@
retlw    0           ; 1F code
retlw    0           ; 20 code
retlw   'c'          ; 21 code                c
retlw   'x'          ; 22 code                x
retlw   'd'          ; 23 code                d
retlw   'e'          ; 24 code                e
retlw   '4'          ; 25 code                4$
retlw   '3'          ; 26 code          3#
retlw    0           ; 27 code
retlw    0           ; 28 code
retlw   ' '          ; 29 code          ESPACIO
retlw   'v'          ; 2A code          v
retlw   'f'          ; 2B code          f
retlw   't'          ; 2C code          t
retlw   'r'          ; 2D code          r
retlw   '5'          ; 2E code  5%
retlw    0           ; 2F code
retlw    0           ; 30 code
retlw   'n'          ; 31 code          n
retlw   'b'          ; 32 code          b
retlw   'h'          ; 33 code          h
retlw   'g'          ; 34 code          g
retlw   'y'          ; 35 code          y
retlw   '6'          ; 36 code          6&
retlw    0           ; 37 code
retlw    0           ; 38 code
retlw    0           ; 39 code
retlw   'm'          ; 3A code m
retlw   'j'          ; 3B code          j
retlw   'u'          ; 3C code  u
retlw   '7'          ; 3D code          7/
retlw   '8'          ; 3E code          8(
retlw    0           ; 3F code
retlw    0           ; 40 code
retlw   ','          ; 41 code          ,<
retlw   'k'          ; 42 code          k
retlw   'i'          ; 43 code          i
retlw   'o'          ; 44 code          o
retlw   '0'          ; 45 code          0=
retlw   '9'          ; 46 code          9)
retlw    0           ; 47 code
retlw    0           ; 48 code
retlw   '.'          ; 49 code          .>
retlw   '/'          ; 4A code          /?
retlw   'l'          ; 4B code          l
retlw   ';'          ; 4C code          ;:
retlw   'p'          ; 4D code          p
retlw   '-'          ; 4E code          -_
retlw    0           ; 4F code
retlw    0           ; 50 code
retlw    0           ; 51 code
retlw   0x27               ; 52 code    '"
retlw    0           ; 53 code
retlw   '['          ; 54 code          [{
retlw   '='          ; 55 code          =+
retlw    0           ; 56 code
retlw    0           ; 57 code
retlw   0xA9         ; 58 code  CAPS
retlw   0xA9         ; 59 code  SHIFT DER
retlw   0X0A         ; 5A code          ENTER
retlw   ']'          ; 5B code  ]
retlw    0           ; 5C code
retlw   0x5C         ; 5D code          \
retlw    0           ; 5E code
retlw    0           ; 5F code
retlw    0           ; 60 code
retlw    0           ; 61 code
retlw    0           ; 62 code
retlw    0           ; 63 code
retlw    0           ; 64 code
retlw    0           ; 65 code
retlw   0x08         ; 66 code  BACKSPACE
retlw    0           ; FIN
End
```

III. REFERENCIA

Microchip Technology INC; "PIC16F8X 18 pin Flash/EEPROM 8 bit Microcontrollers", Documento número DS·30430C, 68 pags.

Konzak gary j., Annabooks; "PC keyboard Design", 2da edición, San Diego, CA, 1993.

Messmer, Hans-Peter; "The indispensable PC hardware Book – Your Questions Answered"; Addison-Wesley Publishing Company, Reading, MA, 1994.

PC Keyboard FAQ, http://www.repairfaq.org/filipg/, 2001

Interfacing the PC's Keyboard
http://www.senet.com.au/~cpeacock/index.html, 2001