LABORATORIO Nº3
INTERRUPCIONES, TECLADO MATRICIAL, LCD

I. OBJETIVOS

- ✓ Utilizar las interrupciones del microcontrolador para solucionar problemas prácticos.
- ✓ Utilizar adecuadamente dispositivos de interfaz de usuario tales como, LCD, y Teclado.

II. MATERIAL DE APOYO

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

III. EQUIPOS NECESARIOS

- ➢ PC y Software de simulación.

IV. EJEMPLO LCD Y TECLADO MATICIAL.

Para el siguiente ejemplo se maneja la misma configuración de bits para el pic 16F887 que se estableció en las guías anteriores.

- LCD (HD44780U)

Para este ejemplo, será necesario hacer dos diferentes librerías para cargar datos y para cargar comandos a la LCD. Es tema de consulta como crear un símbolo y cargarlo a la LCD y la extracción de datos de la DDRAM de ésta.

Para la carga de un comando o dato específico se deben tener en cuenta los tiempos de configuración de espera que se requieren y el manejo de los cambios de estado de los bits de control.

Los bits de control son: RS - Selecciona registros, R/W - Selecciona estado de lectura o escritura, E - Inicia lectura o escritura de un dato o comando.

- ➢ Carga de comandos.

Para la creación de esta librería se debe tener en cuenta el estado de los bits de control y modificarlos como se desea para luego hacer la carga del comando seleccionado (**El estudiante debe consultar que tipos de comandos existen**). Para esto, se desactivaran los bits RS y R/W, luego se cargara el valor deseado y se activara el bit E, se deberá tener en cuenta un tiempo de espera-escritura, y por último se desactivara el bit E. Para asegurar la escritura se esperara un tiempo mínimo 3 ms. **El estudiante debe consultar de donde vienen esos tiempos de espera.**

```
LCD_CMD:
    BCF     RS
    BCF     RW
    MOVWF   BUS_DATA
    BSF     E
    NOP
    NOP
    BCF     E
    CALL    RETARDO_3ms
    RETURN
```

De esta manera se podrá cargar cualquier comando deseado de forma genérica solo con asignarle un valor al registro W.

- ➢ Carga de datos.

Para la carga de datos en la LCD se sigue el comando que esta predefinido en el datasheet de la ésta.

```
LCD_DAT:
    BSF     RS
    BCF     RW
    MOVWF   BUS_DATA
    BSF     E
    NOP
    NOP
    BCF     E
    CALL    RETARDO_50us
    RETURN
```

De esta manera se podrá cargar cualquier dato deseado de forma genérica solo con asignarle un valor al registro W.

- Configuración de LCD y puertos asociados.

Para la configuración de la LCD, en este ejemplo, se manejara una *fosc* de 8Mhz, y se trabajaran los bits de control por PORTE y el bus de datos de salida o comando será por PORTD.

<u>Nota:</u> Para bautizar o anclar un nombre a un puerto o sus respectivos bits se puede manejar la sentencia #DEFINE, la cual nos ayuda a cambiar de puertos de manejo sin necesidad de alterar el código del programa.

```
#DEFINE BUS_DATA    PORTD
#DEFINE RS          PORTE,0
#DEFINE RW          PORTE,1
#DEFINE E           PORTE,2
```

```
LCD_PORT_CONF:
    BANKSEL OSCCON
    MOVLW   0XF1
    MOVWF   OSCCON
    BANKSEL ANSEL
    CLRF    ANSEL
    BANKSEL TRISD
    CLRF    BUS_DATA
    BCF     RS
    BCF     RW
    BCF     E
    BANKSEL PORTA
    CLRF    BUS_DATA
    BCF     RS
    BCF     RW
    BCF     E
    RETURN
```

El código inmediatamente anterior establece el valor del oscilador a trabajar, limpia los diferentes TRIS y PORT que se manejan y el ANSEL para trabajar con digital I/O.

- Inicialización de LCD.

Para la inicialización de la LCD se debe cargar el comando *Fuction Set* en un orden de tiempos de espera específico. **El estudiante debe consultar de donde vienen esos tiempos de espera y el por qué se debe hacer esa carga consecutiva del comando *Fuction Set*.** Para la demás configuración como, pixeles y cantidad de líneas, display y estado del cursor, dirección del cursor y borrado de pantalla se siguen los comandos establecidos en el datasheet de la LCD.

```
LCD_INIT:
    CALL    LCD_PORT_CONF
    CALL    RETARDO_25ms
    MOVLW   0X30
    CALL    LCD_CMD
    CALL    RETARDO_5ms;
    MOVLW   0X30
    CALL    LCD_CMD
    CALL    RETARDO_100us
    MOVLW   0X30
    CALL    LCD_CMD
    MOVLW   0X38
    CALL    LCD_CMD
    MOVLW   0X0F
    CALL    LCD_CMD
    MOVLW   0X06
    CALL    LCD_CMD
    MOVLW   0X01
    CALL    LCD_CMD
    RETURN
```

El código inmediatamente anterior establece una configuración de funcionamiento que dicta:

PIXELES DE 5X8 Y DOS LINEAS.
DISPLAY PRENDIDO, CURSOR ACTIVO Y TITILANDO.
CURSOR INCREMENTA.
BORRAR PANTALLA LCD.

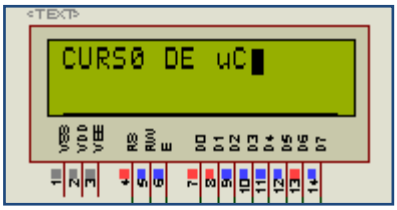
- Implementación de LCD

Finalmente para enviar un mensaje a la LCD se usan las librerías anteriormente creadas.

En el orden de llamado se cumple de la siguiente manera, primero el llamado a la librería de inicialización y luego el llamado de la librería de carga de datos.

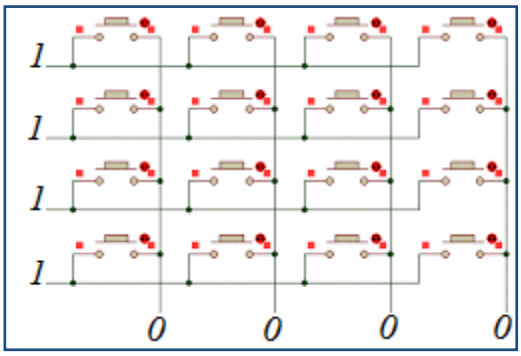
```
ORG 0X00
CALL    LCD_INIT
MOVLW   'C'
CALL    LCD_DAT
MOVLW   'U'
CALL    LCD_DAT
        .
        .
        .
CALL    LCD_DAT
MOVLW   'u'
CALL    LCD_DAT
MOVLW   'C'
CALL    LCD_DAT
GOTO    $
```

El mensaje de salida es “CURSO DE uC”.



- Teclado Matricial

Teniendo en cuenta que el teclado matricial es un arreglo de pulsadores, los cuales dependen de los estados lógicos de las filas y columnas, se cargará para este ejemplo un estado lógico alto (1) para las filas y un estado lógico bajo (0) para las columnas.



Dado que es necesario tener en cuenta en cualquier momento del funcionamiento del programa los estados del teclado, es decir, si se ha oprimido una tecla, se usarán los registros de interrupción del pic.

- Configuración de Puertos e Interrupciones

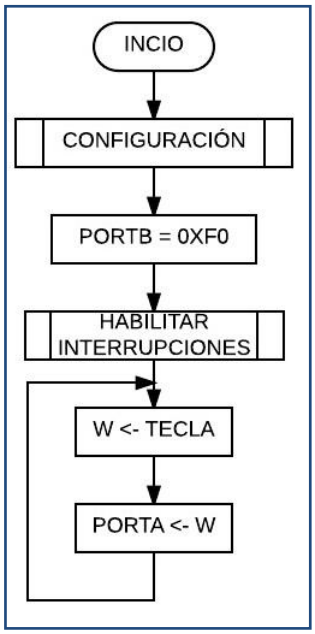
Se sigue manejando una *fosc* de 8MHz. Se configura el PORTB como entradas y salidas. Y se usará el PORTA para visualizar por medio de un display 7-seg que tecla fue oprimida.

Se utilizara el PORTB dado que tiene bandera de interrupción por cambio de estado. Le asignaremos el nibble bajo a las columnas estableciéndolas como salidas y el nibble alto a las filas estableciéndolas como entradas. Es necesario también, activar las resistencias internas de pull-up. **El estudiante debe consultar que tipos de interrupciones existen para el PORTB así como también revisar los registros OPTION_REG, INTCON, PIE1, PIE2.**

```
CONFIGURACION:
    BANKSEL OSCCON
    MOVLW   0XF1
    MOVWF   OSCCON
    MOVLW   0XF0
    MOVWF   TRISB
    CLRF    TRISA
    BANKSEL ANSELH
    CLRF    ANSELH
    CLRF    ANSEL
    BSF     INTCON,7
    BSF     INTCON,3
    BANKSEL OPTION_REG
    BCF     OPTION_REG,7
```

```
BANKSEL OPTION_REG
BCF     OPTION_REG,7
BANKSEL WPUB
MOVLW   0XF0
MOVWF   WPUB
MOVWF   IOCB
BANKSEL PORTA
MOVLW   0XF0
MOVWF   PORTB
CLRF    PORTA
CLRF    TECLA
RETURN
```

El siguiente diagrama de flujo describe el funcionamiento del programa a realizar.

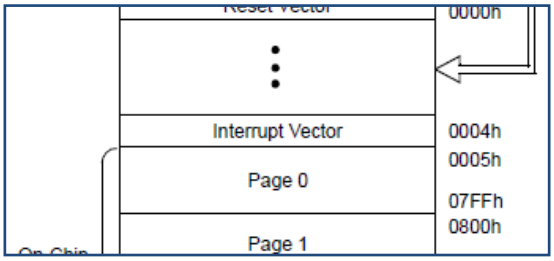


Teniendo en cuenta que el teclado matricial es de la siguiente forma:

```
INICIO:
    CALL    CONFIGURACION
LOOP:
    MOVF    TECLA,W
    MOVWF   PORTA
    GOTO    LOOP
```

- Interrupción generada.

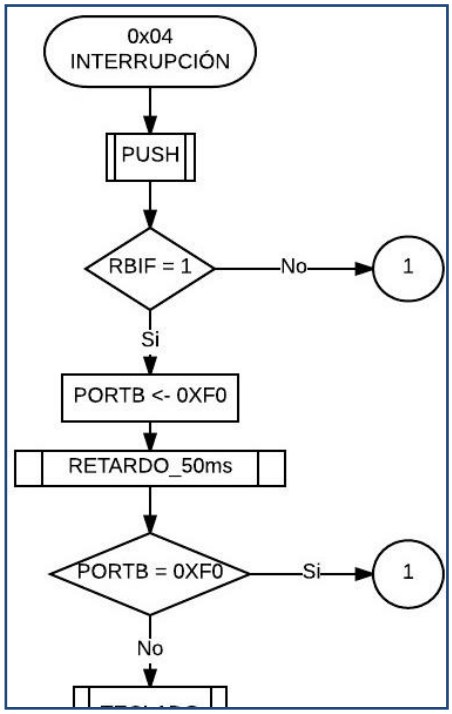
Cuando se genera una interrupción y el bit GIE esta activado, el puntero del programa se dirige a la posición 0x04 la cual se encarga de dar dirección del programa que se quiere correr dada la interrupción.

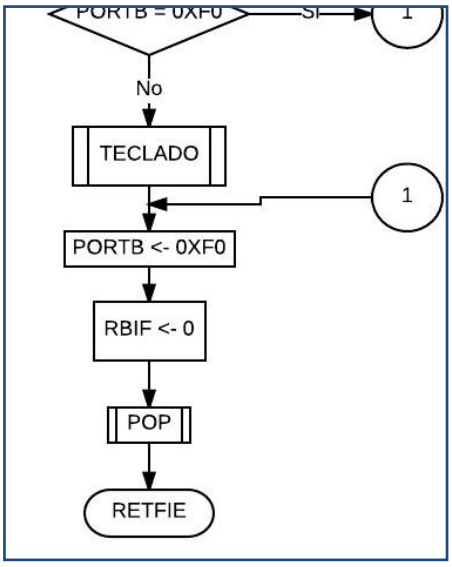


Es necesario salvar el valor del registro STATUS y el registro W dado que la interrupción se puede generar en cualquier momento del programa. Guardar el contexto es necesario para que una vez terminado el programa de la interrupción el programa siga corriendo normalmente.

En el programa de interrupción es necesario preguntar qué clase de interrupción ha ocurrido dado que hay diferentes razones por las cuales se puede generar. Para esto se tiene en cuenta el estado de la bandera de interrupción respectiva. Es este caso la bandera de interrupción por el PORTB dado un cambio de estado está dada por el bit RBIF del registro INTCON.

El siguiente diagrama de flujo describe el funcionamiento del programa dada una interrupción.



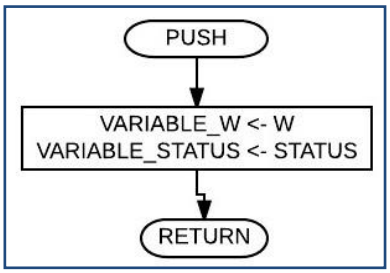


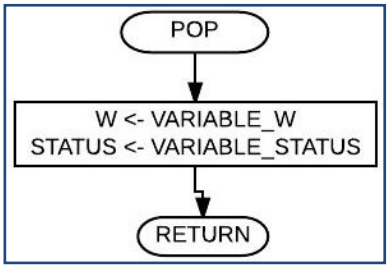
El código del diagrama de flujo inmediatamente anterior es el siguiente:

```
  ORG 0X00
  GOTO    INICIO
  ORG 0X04
  GOTO    INTERRUPT
```

```
INTERRUPT:
    CALL    PUSH
    BTFSS   INTCON,0
    GOTO    UNO
    MOVLW   0XF0
    MOVWF   PORTB
    CALL    RETARDO
    MOVLW   0XF0
    XORWF   PORTB,W
    BTFSS   STATUS,Z
    CALL    TECLADO
UNO:
    MOVLW   0XF0
    MOVWF   PORTB
    BCF     INTCON,0
    CALL    POP
    RETFIE
```

Las librerías que se usan para guardar el contexto y para cargarlo las llamaremos PUSH y POP respectivamente.

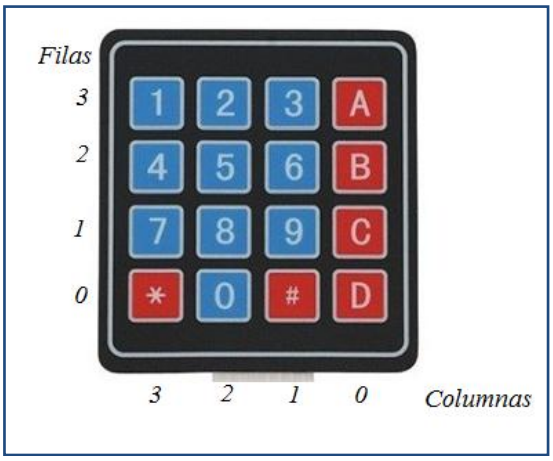




El código del diagrama de flujo inmediatamente anterior es el siguiente:

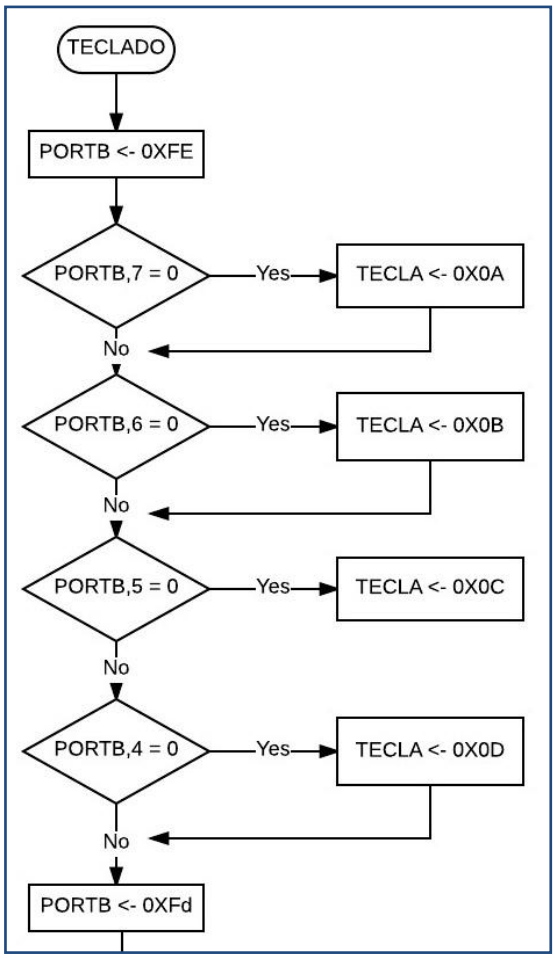
```
    CBLOCK
    W_V
    STATUS_V
    ENDC
PUSH:
    MOVWF   W_V
    MOVF    STATUS,W
    MOVWF   STATUS_V
    RETURN
POP:
    MOVF    W_V,W
    MOVF    STATUS_V,W
    MOVWF   STATUS
    RETURN
```

Se asignará un valor diferente para cada combinación de PORTB con la cual se establecerá que tecla ha sido oprimida. El siguiente diagrama de flujo explica el comportamiento de asignación de valor a una variable TECLA creada previamente, que será la encargada de llevar este valor al PORTA.



El diagrama de flujo solo contiene la combinación de las columnas 0, 1, 2 y 3 con la fila 3, esta secuencia se repetirá para todas las filas.

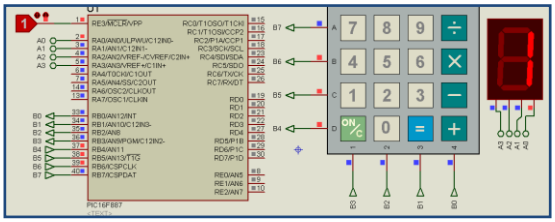
De ahí, el PORTB tomara valores de 0XFE, 0XFD, 0XFB y 0XF7.



El código del diagrama de flujo inmediatamente anterior es el siguiente:

```
TECLADO:
     MOVLW    0XFE
     MOVWF    PORTB

     BTFSC    PORTB,7
     GOTO     $+4
     MOVLW    0X0A
     MOVWF    TECLA
     RETURN

     BTFSC    PORTB,6
     GOTO     $+4
     MOVLW    0X0B
     MOVWF    TECLA
     RETURN

     BTFSC    PORTB,5
     GOTO     $+4
     MOVLW    0X0C
     MOVWF    TECLA
     RETURN

     BTFSC    PORTB,4
     GOTO     $+4
     MOVLW    0X0D
     MOVWF    TECLA
     RETURN

     MOVLW    0XFD
     MOVWF    PORTB
```

De esta manera el programa está listo para ser implementado. El teclado de simulación tiene las teclas cambiadas, por esto cuando es oprimida la tecla del 7 se visualizara un 1.



V. DESCRIPCIÓN DEL LABORATORIO

LABORATORIO N°3: Cajero Automático. Este programa debe contar con:

- Ingreso de contraseña.
- Cambio de contraseña.
- Retiros.
- Consulta de saldo.

Las teclas A, B, C, y D tendrán que servir para:

- Enter
- Ir atrás
- Borrar carácter (Esta debe funcionar solo cuando se ingrese o cambie la clave)
- Salir de operación (Ir a Inicio)

La contraseña debe ser de 4 caracteres. Al cuarto intento de error al ingresar la clave el cajero se debe bloquear y tendrá que salir un mensaje avisando dicho bloqueo.

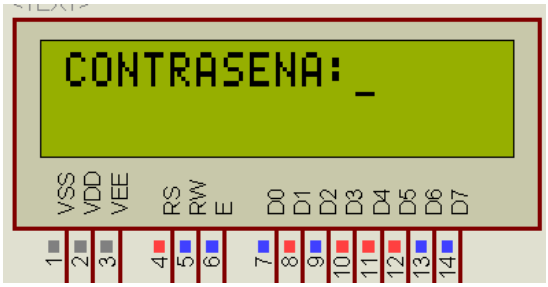
Los retiros del cajero serán:

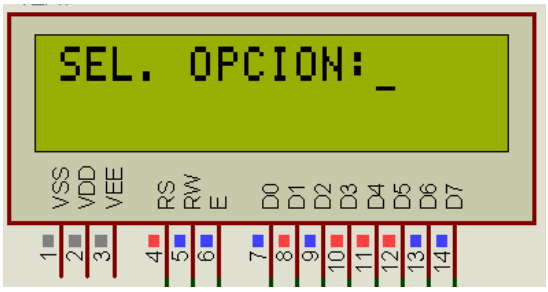
1) $5000
2) $10000
3) $15000
4) $20000

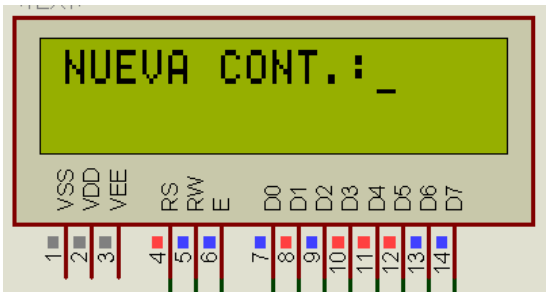
La cuenta debe tener un fondo base por defecto. Si en la cuenta no hay dinero suficiente, tendrá que salir un mensaje avisando la liquidez.

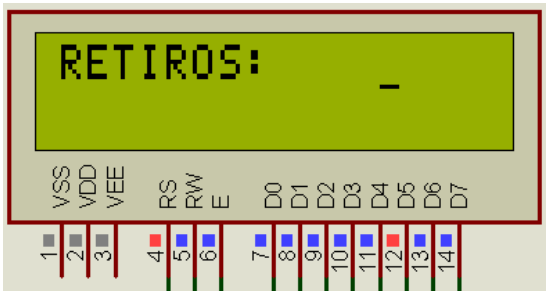
Usar teclado matricial y LCD para el desarrollo de la práctica.

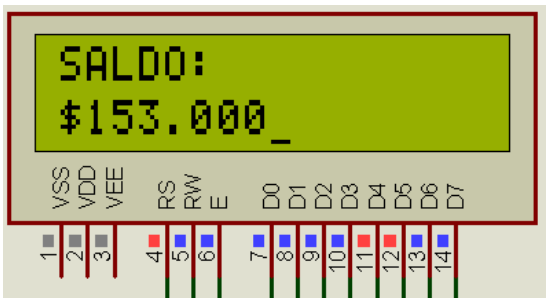
Sugerencia de MENU:











VI. BIBLIOGRAFIA

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

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

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

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

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

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