INSTITUTO POLITÉCNICO NACIONAL

Centro de Estudios Científicos y Tecnológicos Nº 1

"Gonzalo Vázquez Vela"

Academia de Sistemas Digitales

Prácticas de Micro Electrónica Programable

NOMBRE DEL ALUMNO: _____
Apellido Paterno

_____
Apellido Paterno Nombre

N° DE BOLETA: _____ GRUPO: _____

ASIGNATURA: ***Micro Electrónica Programable***

HOJA	DE

FECHA		
DIA	MES	AÑO

EVALUACION

PROFESOR: _____

Práctica 7
Interrupciones externa y por cambio de nivel

Competencias de La Unidad:

- Emplea las interrupciones en el desarrollo de soluciones a problemas complejos

Resultado de Aprendizaje Propuesto (RAP):

- Usa interrupciones basadas en temporizadores para resolver problemas.

Objetivos de la Práctica:

1. Realizar la simulación de un programa para comprobar su funcionamiento utilizando herramientas computacionales
2. Utilizar las interrupciones por cambio de nivel en el puerto B para efectuar una opción determinada.
3. Efectuar programas en un circuito basado en microcontrolador, para comprobar su funcionamiento.
4. Configurar rutinas de servicio de interrupción utilizando el compilador de C.

Equipo Necesario	*Material Necesario*
Computadora (con el Software MPLAB IDE, IC-PROG o similar, compilador C, Simulador de circuitos electrónicos "Proteus") Programador tipo JDM o similar.	Instrucciones del PIC 16F887 u otro de gama media o alta. Manual de referencia de CCS

Introducción Teórica
Interrupciones

Esta familia de microcontroladores dispone de 14 fuentes de interrupción, de las cuales algunas de ellas se habilitan por medio del registro de control INTCON, otras, sin embargo, como la del convertidor A/D se encuentra en el registro PIE1, en este mismo registro está el bit del TIMER1 (denominado TMR1IE en el bit PIE1<0>) entre otros bits que habilitan interrupcciones. En el registro INTCON sólo se encuentran tres controles de interrupciones. La interrupción ocasionada en la terminal RB0/INT, aquellas relacionadas al cambio de estado en las terminales RB7:RB4 y el desbordamiento del temporizador TMR0, el resto de los controles de periféricos y funciones especiales se encuentran en los registros PIE1, PIE2.

Un bit importante en las interrupciones es el bit GIE (INTCON<7>) que habilita las interrupciones no enmascaradas, por otra parte cuando tiene lugar una interrupción, el valor del Contador de Programa (PC) se envía a la pila y se carga con la dirección del vector de interrupciones. Por tal motivo el microcontrolador comenzar a ejecutar las instrucciones de la subrutina de interrupciones y cuando se encuentre una instrucción RETFIE (En el caso de ensamblador) dará por finalizada la subrutina volviendo a la siguiente dirección de la última instrucción ejecutada antes de producirse la interrupción. Al atender una interrupción, el bit GIE se borra automáticamente con lo que se deshabilita la posibilidad de futuras interrupciones.

Para determinar qué periférico o función ha ocasionado la interrupción se deberán leer los bits de control de las correspondientes interrupciones. En los microcontroladores existen dos interrupciones ligadas a puerto B, las cuales son: en la terminal RB0/INT denomina interrupción externa y aquella que se produce con cualquier cambio de estado que se produzca en las terminales RB7:RB4 habilitadas mediante la bandera de interrupciones RBIE (INTCON<3>), ocasionando que dicho cambio genere uno en el bit RBIF (INTCON<0>) produzca la interrupción.

Algunos registros asociados a las interrupciones son:

REGISTRO OPTION u OPTION_REG

El Registro de OPTION_REG es un registro que puede ser leído o escrito y que contiene varios bits de control para configurar la asignación del preescaler al TMR0 o al WDT, la interrupción externa, el TMR0 y las resistencias de pull-up del PORTB.

REGISTRO INTCON

El registro INTCON es un registro de lectura y escritura que contiene los bit de habilitación de interrupciones por desbordamiento de TMR0 por cambio de nivel en el PORTB e interrupciones externas por la línea RBO/INT.

Otros detalles pueden ser consultados en la hoja de especificación del microcontrolador.

En este caso el lenguaje tiene directivas específicas para cada una de las fuentes de interrupciones definidas como sigue:

#INT_xxx función_de_interrupción

Estas directivas especifican que la función que le sigue es una función de interrupción. Las funciones de interrupción no pueden tener ningún parámetro. Como es natural, no todas las directivas pueden usarse con todos los dispositivos. Algunas de las directivas son:[1]

#INT_EXT INTERRUPCIÓN EXTERNA
#INT_TIMER0 DESBORDAMIENTO DEL TIMER0
#INT_RTCC DESBORDAMIENTO DEL TIMER0(RTCC)
#INT_RB CAMBIO EN UNO DE LOS PINES B4,B5,B6,B7
#INT_AD CONVERSOR A/D
#INT_EEPROM ESCRITURA EN LA EEPROM COMPLETADA
#INT_TIMER1 DESBORDAMIENTO DEL TIMER1
#INT_TIMER2 DESBORDAMIENTO DEL TIMER2
#INT_CP1 MODO CAPTURA DE DATOS POR CCP1
#INT_CCP2 MODO CAPTURA DE DATOS POR CCP2
#INT_SSP PUERTO DE SERIE INTELIGENTE(SPI, I2C)
#INT_PSP PUERTO PARALELO
#INT_TBE SCI DATO SERIE TRANSMITIDO
#INT_RDA SCI DATO SERIE RECIBIDO
#INT_COMP COMPARADOR DE INTERRUPCIONES
#INT_ADOF DESBORDAMIENTO DEL A/DC
#INT_RC CAMBIO EN UN PIN Cx
#INT_I2C I2C DEL 14000
#INT_BUTTON PULSADOR DEL 14000
#INT_LCD LCD 92x

El compilador salta a una función cuando se detecta una interrupción. Es el propio compilador el encargado de generar el código para guardar y restaurar el estado del procesador.

También es el compilador quien borrará la interrupción (la bandera). Sin embargo, nuestro programa es el encargado de llamar a la función para activar previamente la interrupción junto con la bandera global de interrupciones mediante la siguiente directiva

ENABLE_INTERRUPTS(level)

Esta función activa la interrupción de acuerdo al tipo de interrupción. Y la definición del procedimiento o rutina de atención, al producirse la interrupción indicada se define en el programa. El nivel GLOBAL permite a todas las interrupciones que estén habilitadas de forma individual.

ACTIVIDADES TEÓRICAS PREVIAS

Investigar los siguientes:

- ***Investigar que son interrupciones***
- ***Investigar las interrupciones por cambio de nivel***
- ***¿Qué tipo de interrupciones tienen los Microcontroladores PIC?***
- **¿Cuáles son las terminales a utilizar para interrupción en Puerto B?**

[1] Para consultar las directivas existen revisar el manual de referencia de CCS.

- ***Menciona las características de la terminal RB0/INT del puerto B***
- ***Menciona las características de la terminal RB4 a RB7 del puerto B***

ACTIVIDADES PREVIAS

- ***Crear un proyecto de nombre pra7 en la carpeta c:\MEPIC\practica7 en MPLAB o PIC C Compiler. Los programas de cada ejercicio deben ser guardados con el nombre practica7X.c con X= 1, 2, 3…,A.***

- ***En el caso de utilizar MPLAB, realizar los siguientes pasos:***

    a. ***Utilizar Project wizard y seleccionar el compilador de c***
    b. ***Agregar al proyecto los archivos adecuados con extensión c y h.***
    c. ***Habilitar Simulador MPLAB SIM y modificar la frecuencia del simulador a 4 Mhz.***
    d. ***Utilizaremos la herramienta de stopwatch, para obtener la elija Debugger >> Stopwatch.***
    e. ***Obtener la herramienta de watch, de la siguiente manera View>> watch.***
    f. ***Y seleccione los registros PORTA, PORTB, PORTC, PORTD, PORTE, TRISA, TRISB, TRISC, TRISD, TRISE y W***
- ***Si usa PIC C compiler crear el proyecto únicamente.***

ACTIVIDADES PRÁCTICAS

Parte 1

1. Armar los siguientes circuitos correspondientes y probar los siguientes programas, además verificar el funcionamiento en el simulador Proteus.

Ejemplo 1

El siguiente programa ejemplifica el funcionamiento de la interrupción externa por la terminal RB0/INT. Simularlo en el circuito de la figura 7.1

```
#include <16F887.h>
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use delay(clock= 4000000)
#use standard_io(b)
#use standard_io(d)
#INT_EXT //Atención a interrupción por cambio en RB0
void ext_isr(){ //Función de interrupción
output_toggle(pin_d1);
output_toggle(pin_d4);
}
void main() {
output_low(PIN_d1); //Apaga LED
output_high(PIN_d4); // enciende LED
enable_interrupts(int_ext); //Habilita int. RB0…
ext_int_edge(H_TO_L); //por flanco de bajada
enable_interrupts(GLOBAL); //Habilita int. general
while (true){ //Bucle infinito de espera
```

```
}
}
```

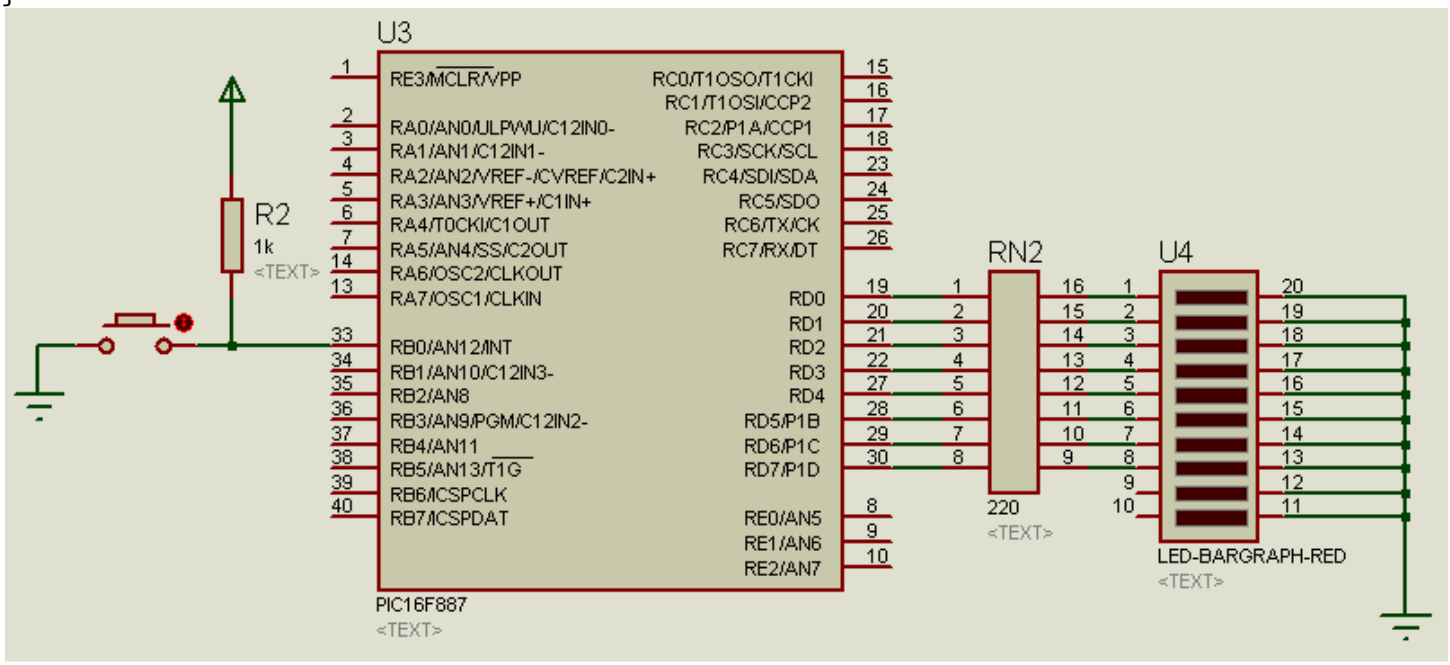

Figura 7.1

Ejemplo 2

El siguiente programa ejemplifica el funcionamiento de la interrupción por cambio de estado en las terminales RB4:RB7. Simularlo en el circuito de la figura 7.2

```
#include <16F887.h>
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use delay(clock=4000000)
#use fast_io(b)
#use standard_io(d)
#byte WPUB=0X95
#byte IOCB=0X96
#byte OPTION=0X81
#byte ANSEL=0X188
#byte TRISB=0X86
#byte ANSELH=0X189
int datoRB; //almacena el último dato
byte led=0b00000001;
#INT_RB //Atención a interrupción por cambio en RB4-RB7
void RB_isr(){ //Función de interrupción
datoRB=input_b();
output_d(0);
output_high(pin_d4);
delay_ms(2000);
output_low(pin_d7);
}
void main() {
wpub=0xff;
iocb=0xff;
option=0x0f;
ansel=0x00;
```

```
anselh=0x00;
trisb=0xf0;
output_d(0x00);
datoRB=input_b();
enable_interrupts(int_RB); //Habilita int. en puerto RB
enable_interrupts(GLOBAL); //Habilita int. general
do{led=0b00000001;
do{
output_d(led);
rotate_left(&led,1);
delay_ms(300);
}while(bit_test(led,4)==0);
datoRB=input_b();
}
while (true); //Bucle infinito de espera
}
```

Ejemplo 3

El siguiente programa ejemplifica el funcionamiento de la interrupción por cambio de estado en las terminales RB4:RB7, realizando una función específica al existir un cambio de nivel. Simularlo en el circuito de la figura 7.2

```
#include <16F887.h>
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use delay(clock= 4000000)
#use fast_io(B)
#use standard_io(c)
#byte WPUB=0X95
#byte IOCB=0X96
#byte OPTION=0X81
#byte ANSEL=0X188
#byte TRISB=0X86
#byte ANSELH=0X189
void intermitente(void);
byte datoRB; //almacena el último dato
#INT_RB //Atención a interrupción por cambio en RB4-rB7
void RB_isr()
{ //Función de interrupción
byte conta=0;
byte cambio;
output_high(pin_B0);
output_low(pin_B1);
cambio= datoRB^input_b();
datoRB=input_b();
//if(bit_test(cambio,4))
if(input(PIN_B4)==0)
{intermitente();}
else if(bit_test(cambio,5))
{
do{intermitente();
conta++;}
while(conta<2);
}
```

```
conta=0;
output_low(PIN_B0);
output_high(PIN_B1);
}
void main() {
byte led=0b00000001;
wpub=0xff;
iocb=0xff;
option=0x0f;
ansel=0x00;
anselh=0x00;
trisb=0xf0;
output_low(PIN_e0);
output_high(PIN_e1);
datoRB=input_b();
enable_interrupts(int_RB); //Habilita int. en puerto RB
enable_interrupts(GLOBAL); //Habilita int. general
do{
led=0b00000001;
do{
output_d(led);
rotate_left(&led,1);
delay_ms(300);
}while(bit_test(led,4)==0);
datoRB=input_b(); //obtiene valor si interrupcion
}
while (true); //Bucle infinito de espera
}
void intermitente(void)
{
output_d(0);
delay_ms(1000);
output_d(255);
delay_ms(1000);
output_d(0);
delay_ms(1000);
}
```

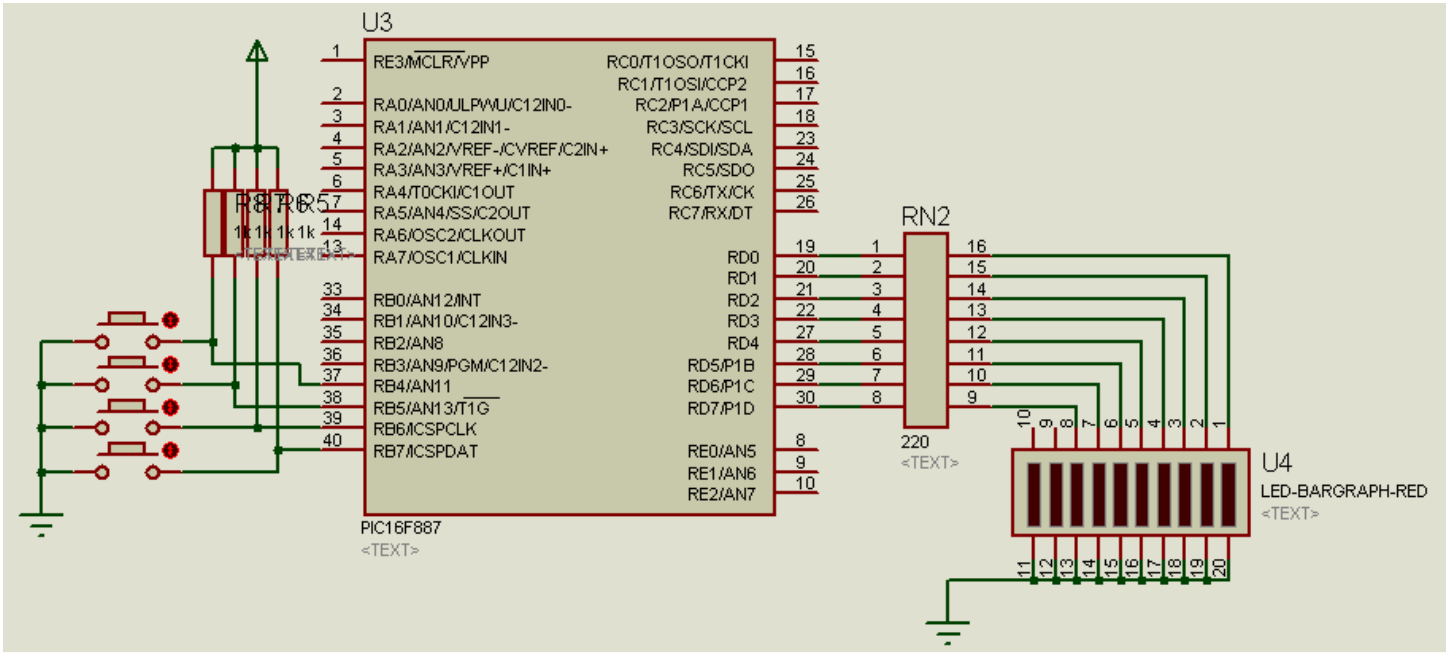

Figura 7.2

Parte 2

Realizar los siguientes programas y sus respectivas simulaciones en Proteus

A. Proponga un programa que efectué el parpadeo de dos led conecta en RE0 y RE1; y al producirse y una interrupción en RB0/INT cuenta en los bits RB4:RB7 las veces que se producen la interrupción al llegar a 15 reinicie la cuenta en 0. Utilice el circuito de la figura 7.1.

B. Utilizando la interrupción de cambio de nivel en RB4-RB7, suponga que tiene alternado dos led en RE1y RE0, indicando que está en espera de oprimir el botón realizara la función siguiente:

Oprimo RB4: los leds de puerto D rotan a la derecha y la izquierda 1 veces,
Oprimo RB5: los leds de puerto D rotan a la derecha y la izquierda 2 veces,
Oprimo RB6: los leds de puerto D parpadean bits RD0 y RD3, y luego RD1 y RD2 3 veces
Oprimo RB7: los leds de puerto D parpadean bits RD1 y RD2, y luego RD0 y RD3 4 veces

Modifique el circuito de la figura 7.2 como lo requiera, además, tome en cuenta que la acción realizada al oprimir cada botón deben de representarse en funciones.

2. Conclusiones

A. Realizar conclusiones de manera individual.

3. Cuestionario

a) ¿Qué es una interrupción por cambio de nivel?
b) ¿Cómo se debe de configurar el puerto B para poder ser utiliza la interrupción por cambio de nivel o externa?
c) *¿Qué ventaja presenta utiliza interrupción a diferencia de la técnica de consulta o polling?*
d) *¿Cuál es la función de la directiva* `enable_interrupts()`*?*
e) *Investiga, ¿Cómo se realizarían en lenguaje en ensamblador un interrupción externa mediante RB0/INT?*
f) *Investiga ¿Cómo se realizarían en lenguaje en ensamblador una interrupción por cambio de nivel?*
g) *¿Qué ventajas observa al programar interrupciones mediante lenguaje comparado con lo investigado en el inciso e) y f)?*

Comentarios Finales

- **El alumno entrega un reporte de la práctica, como el profesor lo indique.**
- **El reporte debe contener el diagrama de flujo o algoritmo (Seudo código) de cada uno de los programas.**
- **Además, en el reporte deben anexarse las conclusiones y cuestionario contestado.**