Se recomienda consultar el siguiente enlace y el datasheet del PIC18F2550.
http://picfernalia.blogspot.com.es/2012/06/interrupciones-conceptos-basicos.html

El PIC18F2550 dispone de 10 registros para configurar los distintos tipos de interrupciones: RCON, INTCON, INTCON2, INTCON3, PIR1, PIR2, PIE1, PIE2, IPR1, IPR2.

Y las posibles fuentes de interrupción son:

- Externa por flanco (RB0/INT0, RB1/INT1, RB2/INT2).
- Externa por cambio lógico en el puerto B (pines RB4 a RB7).
- Desbordamiento del timer (TMR0, TMR1, TMR2).
- Conversor analógico digital (A/D).
- Transmisión y recepción del puerto serie (USART).
- Módulo CCP (captura, comparación, PWM).
- Memoria EEPROM/Flash.
- Fallo del oscilador.
- Bus USB.
- Alta tensión.

En estos apuntes nos centraremos en las interrupciones externas, las únicas no periféricas.

El trabajar por interrupciones nos permite estar haciendo nuestra tarea, y que sea el PIC el que nos avise de que ha sucedido un evento. Es decir, nuestro programa principal se verá interrumpido para realizar una subtarea, denominada *subrutina de interrupción*, *ISR*.

Hasta ahora la consulta del estado de los puertos de entrada se ha hecho por *polling*, es decir, preguntando cada cierto tiempo qué valor tenemos a la entrada. Si utilizamos las interrupciones externas del puerto B, se ejecutará nuestro programa principal (*void main()*), y cuando se produzca un cambio en el puerto B, se interrumpirá el programa en el punto donde se encuentre en ese momento para ejecutar la *subrutina de interrupción*. Una vez ejecutada, el PIC volverá a la instrucción donde se quedó del programa principal.

Pasos a seguir:

1. Nivel alto el bit de habilitacion global de interrupciones, *GIE* del registro *INTCON*.
   ```
   INTCONbits.GIE = 1;
   ```

2. En caso de ser necesario, nivel alto el bit de habilitación de instrucciones periféricas, *PEIE* del registro *INTCON* (para las externas del puerto B no es necesario).
   ```
   INTCONbits.PEIE = 1;
   ```

3. Habilitar el bit correspondiente a las interrupciones que se vayan a usar (…EI).
   ```
   INTCONbits.INT0IE = 1;      // RB0/INT0
   INTCON3bits.INT1IE = 1;     // RB1/INT1
   INTCON3bits.INT2IE = 1;     // RB2/INT2
   INTCONbits.RBIE = 1;        // RB4-RB7
   ```

4. Configurar la prioridad, alta o baja (…IP). Tratándose sólo del puerto externo, se recomienda usar una sola subrutina de interrupción, y comprobar dentro en qué pin se ha producido el cambio.

5. Para las que son por flanco, se puede configurar si flanco de subida o de bajada.
   ```
   INTCON2bits.INTEDG0 = 1;        // INT0 flanco de subida
   INTCON2bits.INTEDG1 = 1;        // INT1 flanco de subida
   INTCON2bits.INTEDG2 = 1;        // INT2 flanco de subida
   ```

6. Limpiar la bandera correspondiente a la interrupción habilitada para evitar falsas interrupciones. Estas instrucciones se usarán también al final de la subrutina de interrupción, para indicar que ya se ha ejecutado, y de esta manera permitir que se produzca otra.

```
INTCONbits.INT0IF = 0;      // flag deshabilitado RB0/INT0
INTCON3bits.INT1IF = 0;     // flag deshabilitado RB1/INT1
INTCON3bits.INT2IF = 0;     // flag deshabilitado RB2/INT2
INTCONbits.RBIF = 0;        // flag deshabilitado RB4-RB7
```

7. Dentro de la subrutina, comprobar cual ha saltado, por ejemplo con un *if*.

8. Deshablitar el *flag* al final de la subrutina. Cada vez que salta la interrupción, el PIC lo pone a nivel alto.

NOTA: La función subrutina de interrupción se indica con *void interrupt*. Por ejemplo:

void interrupt ISR()