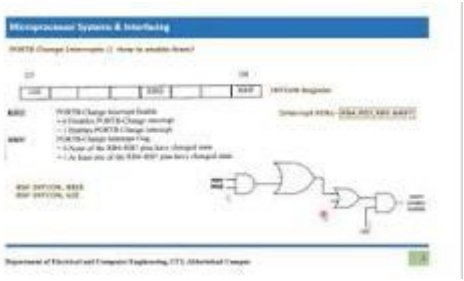
What is portb change interrupt?



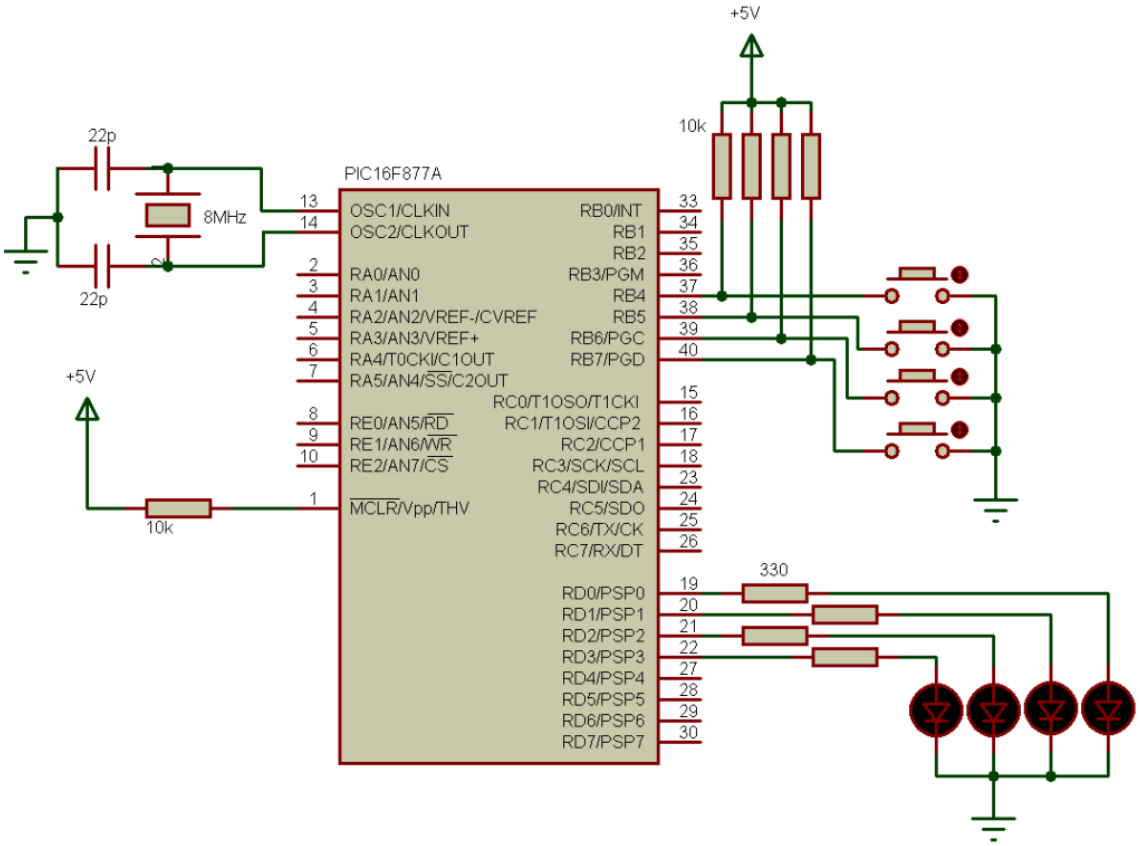
There are four pins including (RB7, RB6,RB5 and RB4) which may cause interrupt to microcontroller if their status changes from zero to one or high to low thats why these are called PORTB-Change interrupt because they interrupt on change and they are present in portb

Four of the PORTB pins, RB7:RB4, have an interrupt-on-change (IOC) feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB port change interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:
a) Any read or write of PORTB. This will end the mismatch condition.
b) Clear flag bit RBIF.
A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared. The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

PIC16F877A PORTB change interrupt example:

This is a small example shows how to use the interrupt-on-change feature of PIC16F877A. Circuit schematic is below:



In the circuit there are 4 buttons and 4 LEDs, when a button is pressed or depressed an interrupt occurred and the microcontroller executes the interrupt routine which turn ON or OFF an LED as shown in the video below.

PIC16F877A PORTB interrupt on change example C code:

The C code below was tested with CCS PIC C compiler version 5.051.

```
1  // PIC16F877A RB change interrupt
2  // http://simple-circuit.com/
3
4  #include <16F877A.h>;
5  #use delay(crystal=8000000)
6
7  byte i;
8  #INT_RB
9  void rb_isr(void)
10 {
11    i = input_b();
12    output_d(i >> 4);
```

```
13 }
14 void main()
15 {
16    set_tris_b(0xF0);
17    clear_interrupt(INT_RB);        // Clear PORTB IOC flag bit
18    enable_interrupts(INT_RB);      // Enable PORTB IOC
19    enable_interrupts(GLOBAL);      // Enable global interrupts
20
21    while(TRUE) ; // Endless loop
22 }
```