(1) Interrupts

Interrupt Routines are program segments which are executed when an interrup source is activated.

PIC16f84 has the following interrupt sources

1) External hardware interrupt

RB0/INT is used as interrup signal source

2) Software interrupt or internal interrupt.

when timer register TMR0 makes a full count; i.e, counter (TMR0) starts from 0 counts upto FF and back to 0 again, a software interrupt or internal interrupt occurs.

3) PORTB bits RB4, RB5, RB6, RB7 can be used as hardware interrupt sources.

(2) INTCON Register (Interrupt Control Register)

7	6	5	4	3	2	1	0
GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

GIE → Global interrupt Enable bit
1 → Enables all unmasked interrupts
0 → Disables all interrupts.

TOIE → TMR0 Overflow Interrupt Enable bit
1 → Enables the TMR0 Interrupt
0 → Disables the TMR0 Interrupt

INTE → RB0/INT External Interrupt Enable bit
1 → Enables the RB0/INT external interrupt
0 → Disables the RB0/INT external interrup!

INTF → RB0/INT External Interrupt Flag bit
1 → The RB0/INT external interrupt occured (must be cleared in software)
0 → The RB0/INT external interrupt did not occur.

(3)

When an interrupt signal is received the following set of events occur

1) Program counter value is saved in to the stack register. That is, return address is placed on top of the stack.

2) Program counter is loaded by 0x04 i.e, program execution goes with the command at address location 0x04

3) At address 0x04 goto interrupt-subroutine Command is executed.

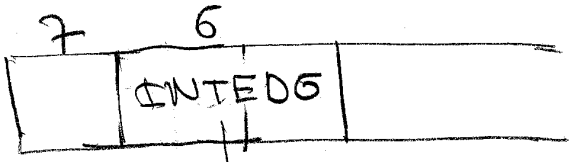
4) After execution of interrupt-subroutine PC is loaded by stack register value

5) Program execution goes from the point it stayed.

(4) An assembly program with interrupt subroutine is written in general as

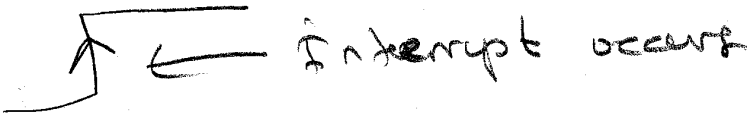
```
list P=16f84A
include "p16f84A.INC"
 org 0x000;
 goto main;
 org 0x004;
 goto interrupt_subroutine

main bsf INTCON, GIE;   -> Enable Global Interrupts.
     bsf INTCON, INTE;  -> Enable RB0/INT interrupt source
     !
     !
loop goto loop;

interrupt_subroutine
  bcf INTCON, INTF;   clear interrupt flag
  !
  !
  retfie;  -> return from interrupt
  end
```

(5) OPTION Register

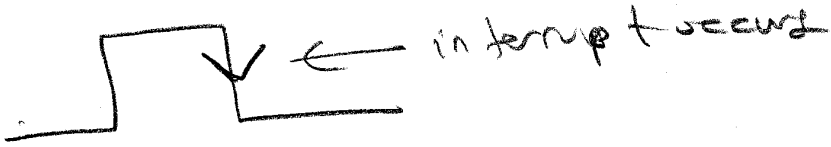


↑ Interrupt Edge Select Bit

1 = Interrupt on rising edge of RB0/INT pin



0 = Interrupt on falling edge of RB0/INT pin



When an interrupt occurs, the program execution continues with interrupt subroutine. To disable new interrupts while the executing the interrupt subroutine INTE should be reset.

↓ bcf INTCON, INTE

and interrupt flag should also be cleared inside interrupt subroutine.

(6) During an interrupt if the content of W register and STATUS register is to be protected then the interrupt subroutine should be written as follows.

```
org   0x004;
goto  interrupt-subroutine
      .
      .
      .
interrupt-subroutine
 movwf  Temp-W;
 swapf  STATUS, W;
 movwf  Temp-S;
        .
        .
        .
swapf  Temp-S, W
movwf  STATUS;
swapf  Temp-W, F;
swapf  Temp-W, W
 retfie;
```

movf → changes the zero flag in STATUS register. Hence swapf is used to save register contents.

(7) Ex: Write a program such that when a falling pulse occurs at RB0/INT RBL is turned ON.

Sln: