```
  list p=16F84A
#include P16F84A.inc
   __CONFIG  _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

;**** --Definicion de variables-- ****
reset org 0x00          ;se define reset en la localidad 0
ORG 4
GOTO INTERRUPCIONWOO

CONTA_1 equ 12      ;se declaran las variables
CONTA_2 equ 33
CONTA_3 equ 21
LED_ACTIVO equ 34
PUERTOS equ 35

;CBLOCK ;Se definen las variables donde se guardarán los registros
Guarda_W equ 36
Guarda_STATUS equ 37
Guarda_CONTA_1 equ 38
Guarda_CONTA_2 equ 39
Guarda_CONTA_3 equ 40
Guardar_PORTA equ 41
Guardar_PORTB equ 42
 ;R_ContA
 ;R_ContB
;ENDC

;configuracion de puertos
            ;Vector de interrupción
```

```
goto inicio
org 0x05                    ;se inicia el programa en la localidad 5

inicio bsf STATUS, RP0
    movlw b'00000'     ; se declaran salidas en el puerto A
    movwf TRISA
    movlw b'00000001'  ; se declaran salidas en el puerto B y la entrada en el bit 0 del puerto B
    movwf TRISB;
    BCF OPTION_REG,6   ; INTERRUPCION POR FLANCO DESCENDENTE (CUANDO SE APRIETA EL
PB)
    bcf STATUS,RP0     ;
    MOVLW B'10010000'  ; HABILITAMOS INTERRUPCION, QUE SEAN LAS EXTERNAS INT Y GIE
    MOVWF INTCON       ; CONFIGURACION REG INTCON
    clrf PORTA         ; se limpia puerto A
    clrf PORTB         ; se limpia puerto B

PUHS
   ;BTFSS PORTB,0      ;si el bit 0 del puerto B esta en 1, salta
   GOTO  SECUENCIA     ;ir a SECUENCIA
   ;GOTO FIN           ; sino terminar programa

INTERRUPCIONWOO
   ; call retardo_20ms
   ; BTFSS PORTB,0
   ; return

  movwf Guarda_W               ; Guarda W y STATUS.
  swapf STATUS,W               ; Ya que "movf STATUS,W", modifica el bit Z.
```

```
movwf Guarda_STATUS
;bcf STATUS,RP0          ; Para asegurarse que trabaja con el banco 0.
movf CONTA_3,W           ; Guarda los registros del retando 0.5 utilizados
movwf Guarda_CONTA_3
movf CONTA_2,W
movwf Guarda_CONTA_2
movf CONTA_1,W
movwf Guarda_CONTA_1
movwf Guardar_PORTA
movf PORTA
movwf Guardar_PORTB
movf PORTB

 MOVF LED_ACTIVO,0
 MOVWF PORTB
 CALL Retardo_UnSeg
 CLRF PORTB
 CALL Retardo_UnSeg

 MOVF LED_ACTIVO,0
 MOVWF PORTB
 CALL Retardo_UnSeg
 CLRF PORTB
 CALL Retardo_UnSeg

 MOVF LED_ACTIVO,0
 MOVWF PORTB
 CALL Retardo_UnSeg
 CLRF PORTB
```

```
    CALL Retardo_UnSeg



FinInterrupcion
        swapf Guarda_STATUS,W       ;Restauramos valor de STATUS.
        movwf STATUS
        swapf Guarda_W,F        ;Restaura W como estaba antes de producirse
        swapf Guarda_W,W        ;interrupción.
        movf Guarda_CONTA_1,W       ;Restaura los registros utilizados en esta
        movwf CONTA_1               ;subrutina y también en la principal.
        movf Guarda_CONTA_2,W
        movwf CONTA_2
        movf Guarda_CONTA_3,W
        movwf CONTA_3
        bcf INTCON,INTF         ;Limpia flag de reconocimiento de la interrupción.
        retfie                  ;Retorna y vuelve a habilitar las interrupciones.



SECUENCIA

    MOVLW b'00000010'
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg

    MOVLW b'00000100'
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg
```

```
    MOVLW b'00001000'
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg

    MOVLW b'00010000'
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg

    MOVLW b'00100000'
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg

    MOVLW b'01000000'
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg

    MOVLW b'10000000'
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg

VUELTA

    MOVLW b'00000000'
```

```
MOVWF PORTB
CALL Retardo_MedioSeg

MOVLW b'10000000'
MOVWF LED_ACTIVO
MOVWF PORTB
CALL Retardo_MedioSeg

MOVLW b'01000000'
MOVWF LED_ACTIVO
MOVWF PORTB
CALL Retardo_MedioSeg

MOVLW b'00100000'
MOVWF LED_ACTIVO
MOVWF PORTB
CALL Retardo_MedioSeg

MOVLW b'00010000'
MOVWF LED_ACTIVO
MOVWF PORTB
CALL Retardo_MedioSeg

MOVLW b'00001000'
MOVWF LED_ACTIVO
MOVWF PORTB
CALL Retardo_MedioSeg

MOVLW b'00000100'
```

```
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg

    MOVLW b'00000010'
    MOVWF LED_ACTIVO
    MOVWF PORTB
    CALL Retardo_MedioSeg

    GOTO PUHS       ;regresa a la subrutina PUHS


Retardo_MedioSeg                    ;etiqueta Retardo_MedioSeg (Retardo_MedioSegardo de 0.5
segundos)
    movlw d'6';movlw d'6'          ;se da el valor de 6 al registro w
    movwf CONTA_3       ;se mueve lo que hay en el registro w a CONTA_3
d     movlw d'101'        ;se da el valor de 101 al registro w
    movwf CONTA_2       ;se mueve el registro w a CONTA_2
c     movlw d'250'        ;se da el valor de 250 a w
    movwf CONTA_1       ;se mueve el registro w a CONTA_1
a     nop               ;no opera
    decfsz CONTA_1      ;CONTA_1 <- CONTA_1 - 1 y  salta si llego a 0 (249 veces)
    goto $-.2           ;no es 0, entonces salta a nop
    decfsz CONTA_2      ;si es 0 entonces CONTA_2 <- CONTA_2 -1 y salta si llega a 0 (repite 100
veces)
    goto $-.6           ;no es 0, entonces a caragar CONTA_1
    decfsz CONTA_3      ; si es 0, entonces CONTA_3 <- CONTA_  y salta si llega a 0
    goto $-.10          ;no es 0, entonces regresa a cargar CONTA_3(repite 5 veces)
    return              ;si es 0, entonces fin del retrdo y regresa
```

```
Retardo_UnSeg


      movlw d'11'         ;etiqueta RET3 (retardo de 1  segundo)
      movwf CONTA_3       ;se da el valor de 6 al registro w
      movlw d'101'        ;se mueve lo que hay en el registro w a CONTA_3
      movwf CONTA_2       ;se da el valor de 101 al registro w
      movlw d'250'        ;se da el valor de 250 a w
      movwf CONTA_1       ;se mueve el registro w a CONTA_1
      nop               ;no opera
      decfsz CONTA_1      ;CONTA_1 <- CONTA_1 - 1 y  salta si llego a 0 (249 veces)
      goto $-.2           ;no es 0, entonces salta a nop
      decfsz CONTA_2      ;si es 0 entonces CONTA_2 <- CONTA_2 -1 y salta si llega a 0 (repite 100
veces)
      goto $-.6           ;no es 0, entonces a caragar CONTA_1
      decfsz CONTA_3      ; si es 0, entonces CONTA_3 <- CONTA_  y salta si llega a 0
      goto $-.10          ;no es 0, entonces regresa a cargar CONTA_3(repite 5 veces)
      return            ;si es 0, entonces fin del retrdo y regresa


;


retardo_20ms


;      movlw d'11'         ;etiqueta RET3 (retardo de 1  segundo)
;      movwf CONTA_3       ;se da el valor de 6 al registro w
;      movlw d'101'        ;se mueve lo que hay en el registro w a CONTA_3
;      movwf CONTA_2       ;se da el valor de 101 al registro w
      movlw d'25'         ;se da el valor de 250 a w
      movwf CONTA_1       ;se mueve el registro w a CONTA_1
      nop               ;no opera
```

```
        decfsz CONTA_1      ;CONTA_1 <- CONTA_1 - 1 y  salta si llego a 0 (249 veces)
        goto $-.2           ;no es 0, entonces salta a nop
;       decfsz CONTA_2      ;si es 0 entonces CONTA_2 <- CONTA_2 -1 y salta si llega a 0 (repite 100 veces)
;       goto $-.6           ;no es 0, entonces a caragar CONTA_1
;       decfsz CONTA_3      ; si es 0, entonces CONTA_3 <- CONTA_  y salta si llega a 0
;       goto $-.10          ;no es 0, entonces regresa a cargar CONTA_3(repite 5 veces)
        return

FIN     BTFSC PORTB,0       ;etiqueta fin, si el bit 0 del puerto B esta en 0, salta.
        ;GOTO PUHS
        END                 ;terminar programa
```