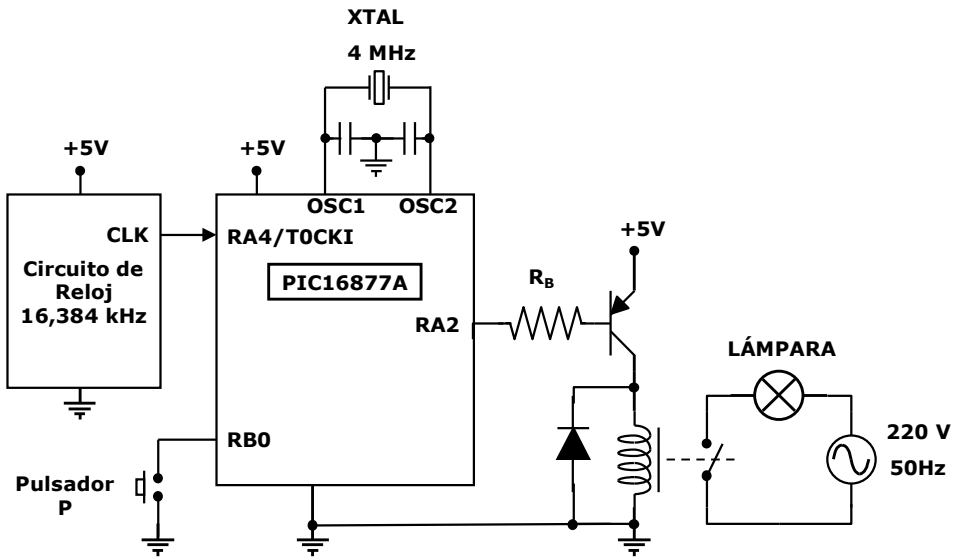
Universidad de Oviedo. Escuela Politécnica de Ingeniería de Gijón
Tecnología Electrónica de Computadores. 2º Curso
Grado en Ingeniería Informática en Tecnologías de la Información

APELLIDOS Y NOMBRE .....

DNI .....

GRUPO PA2 **MODELO A**

Se quiere realizar un circuito para controlar el encendido temporizado de la lámpara de un borrador de EPROM utilizando un microcontrolador PIC16F877A. Se ha decidido utilizar el siguiente circuito:



El cristal del oscilador para el ciclo de instrucción interno tiene una frecuencia $f_{CLK-INT}$=4 MHz y se dispone, adicionalmente, de un reloj externo de frecuencia $f_{CLK-EXT}$=16384 Hz = 16,384 kHz)

El funcionamiento deseado es el siguiente:

- Al arrancar el circuito (tras el encendido o un RESET) la luz estará apagada.
- Cuando se accione el pulsador (conectado a RB0 como se indica) la lámpara (accionada mediante la conexión que se indica a RA2) deberá encenderse durante 85 segundos y luego apagarse. Suponga que la resistencia R_B está correctamente calculada para saturar el transistor cuando entra en conducción

Para conseguir el funcionamiento deseado, nos han proporcionado el programa que se adjunta en la siguiente página. Sin embargo, al probar el programa sobre el circuito, el programa no funciona correctamente.

Se pide:

a) Calcular, con la actual configuración del temporizador, y la actual subrutina de temporización, cuál sería el tiempo que se temporiza realmente en la subrutina "Temporiza", justificando la respuesta

b) Proponer los cambios a realizar en el programa (tanto en la configuración como en el uso del TIMER TMR0) para conseguir temporizar el tiempo solicitado, justificando la respuesta.

c) Con la conexión del PIC que se indica, analizar si la configuración y el uso posterior de los puertos utilizados (PORTA y PORTB) es correcta para conseguir el propósito del programa. Justificar si el programa es correcto y, de no ser así, indicar los errores cometidos y los cambios a realizar en el programa para conseguir un correcto funcionamiento, coherente con el esquema eléctrico del circuito, justificando la respuesta.

d) En el caso de que detecte algún error adicional en el desarrollo del programa, explicar en qué consiste y corregirlo. Tras revisar todos los errores indicar CLARAMENTE los cambios a realizar o, si se prefiere, escribir de nuevo el programa corregido con los cambios necesarios.

```
;********** Grabador_EPROM.asm  ************************
            LIST p=16F877A
            INCLUDE <p16f877A.inc>
; Palabra de configuración del microcontrolador
            __CONFIG   _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_ON & _LVP_OFF

CONT        EQU 0x20

; Vector de RESET

            ORG 0x00
; Prepara contenido de PORTB inicial
; Configuración de puertos y temporizador
Inicio      clrf PORTB
            bsf STATUS,RP0
            movlw b'00000000'
            movwf TRISB
            movlw b'00000000'
            movwf TRISA
            movlw b'10000110'
            movwf OPTION_REG
            bcf STATUS, RP0

; Programa principal
Principal   btfsc PORTB,1
            call Temporiza
            goto Principal


; Subrutina de temporización sobre el pin RA2
Temporiza   bsf PORTB, 1
            movlw d'35'
            movwf CONT
Tparcial    bcf INTCON, T0IF
            movlw d'120'
            movwf TMR0
Espera      btfsc INTCON, T0IF
            goto Espera
            decfsz CONT
            goto Tparcial
            bsf PORTB,2
            return
; Fin de la subrutina de temporización

            END
```

SOLUCIÓN

Se resuelve únicamente el primero de los modelos (PA2 Modelo A). El resto se razonan de forma similar

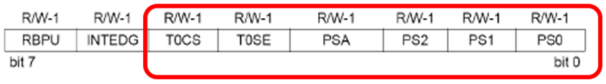
a) Calcular, con la actual configuración del temporizador, y la actual subrutina de temporización, cuál sería el tiempo que se temporiza realmente en la subrutina "Temporiza", justificando la respuesta

En primer lugar, observamos la configuración del Timer0, definida por el registro OPTION_REG:

```
movlw b'10000110'
movwf OPTION_REG
```

El significado del contenido cargado en OPTION_REG relacionado con la configuración es:

Registro OPTION_REG



Bits de configuración del Timer TMR0

T0CS=0 Utiliza el ciclo interno de instrucción, luego $T_{INSTRUCCIÓN}=4*T_{CLK}=4*(1/f_{CLK})=4/(4MHz)=1\ \mu s$

T0SE=0 Utiliza el flanco ascendente si cuenta pulsos del pin T0CKI. Irrelevante, no afecta

PSA=0 Asigna el prescaler al Timer0

PS2:PS0 = 110. Usa un ratio 1:128 en el prescaler. Prescaler divide por 128 la frecuencia.

El tiempo temporizado en una temporización viene dado por el tiempo de desborde de TMR0, es decir:

Tiempo = [(256 - precarga) · Prescaler + 2] · T, siendo precarga el valor cargado en el TMR0 para iniciar la temporización. Si la rutina de temporización estuviera bien hecha (que no lo está) se realizarían 35 temporizaciones del Timer0, precargando un valor TMR0=120.

Por tanto: Tiempo=[(256 - 120) ·128 + 2] · 1 μs=17,410 ms.

Y entonces, cada vez que se ejecutaría Temporiza se realizaría una temporización de:

35*17,410 ms=609,35 ms

Esta solución sería la temporización si se hiciera bien el proceso de la temporización y el resto del programa fuera correcto.

NOTA Adicional: Siendo puristas, y puesto que el flag también se comprueba mal en el programa propuesto (se decrementa justo después de cargar el TMR0, cuando T0IF es cero) realmente el tiempo temporizado es únicamente el correspondiente a la ejecución de las instrucciones, que normalmente despreciamos, aunque se podría calcular en ciclos: esto sería lo que realmente sucede, pero no haría falta hacerlo):

```
Temporiza   bsf PORTB, 1 (un ciclo de instrucción completo, cuatro ciclos de reloj, al venir de un salto)
            movlw d'35' (dos ciclos)
            movwf CONT (dos ciclos)
```

A continuación se ejectuta Tparcial 35 veces, hasta que decsfz CONT da cero como resultado:

```
Tparcial    bcf INTCON, T0IF (dos ciclos la primera vez, cuatro las siguientes, al venir de un salto)
            movlw d'120`(dos ciclos)
            movwf TMR0 (dos ciclos)
Espera      btfsc INTCON, T0IF (dos ciclos, esta es la comprobación que se hace mal, siempre salta)
            decfsz CONT (cuatro ciclos, viene de un salto)
            goto Tparcial (dos ciclos)
```

Y, finalmente, se ejecuta:

```
            bsf PORTB,2 (cuatro ciclos)
            return (dos ciclos)
```

Por tanto, la rutina tardaría realmente en ejecutarse:
8+14+34*16+6 = 572 ciclos de reloj =143 μs = 0,143 ms
(Puede verse como, en condiciones normales, este tiempo siempre es mucho menor que la temporización, en este caso 609 ms, por eso lo despreciamos usualmente. En todo caso, este cálculo no habría que hacerlo en el examen)

b) Proponer los cambios a realizar en el programa (tanto en la configuración como en el uso del TIMER TMR0) para conseguir temporizar el tiempo solicitado, justificando la respuesta.
Vamos a suponer que hacemos bien el programa al comprobar el FLAG, ya que lo vamos a arreglar después.
Se quieren temporizar (según los modelos) entre 70 y 100 segundos, según los modelos.
Calculamos en primer lugar, el tiempo máximo que se puede temporizar en una sola temporización, con cada reloj.
Fijaremos para ello: Precarga=0, y Prescaler=256 (máximo posible)
- Si usamos el reloj interno:
Tiempo=[(256-Precarga)*Prescaler)+2]*4*T_{CLK}=[256*256+2]*4/4MHz= 65,538 ms
Por tanto, para temporizar 70 segundos (el menor de los valores) habría que realizar 267 temporizaciones. Como el registro usado solo tiene 8 bits (máximo 256 temporizaciones) habría que anidar dos bucles, complicando el programa.

Si usamos el reloj externo, entonces:
Tiempo=[(256-Precarga)*Prescaler)+2]*$T_{CLK\text{-}EXTERNO}$=[256*256+2]*1/16,384kHz= 4 s. Por lo que podríamos, con un solo contador de temporizaciones de 8 bits, temporizar hasta 256*4= 1024 segundos, lo cual es válido para todos los modelos pedidos. Además, hay bastante margen. Una solución cómoda es temporizar un segundo, y luego usar la variable contador con los segundos a temporizar: CONT= segundos a temporizar.
Por tanto, usaremos el reloj externo para temporizar un segundo:

1 s=[(256-precarga)*Prescaler)+2]*$T_{CLK\text{-}EXTERNO}$=[(256-precarga)*Prescaler+2]*1/16,384kHz ⇒
(256-precarga)*Prescaler)+2=16384 ⇒ (256-Precarga)*Prescaler)=16384-2=16382 ⇒
⇒(256-Precarga)=16382/Prescaler ⇒Precarga=256-16382/Prescaler
Van a existir diferentes soluciones, vale cualquiera:

Prescaler	Precarga
256	192
128	128
64	0
32	-256 (IMPOSIBLE)

Veamos, por ejemplo, la configuración para un prescaler de 128, con lo que se precargaría precarga=128 en el TMR0 durante la ejecución del programa:

T0CS=1 Utiliza la entrada externa T0CKI, luego T= $T_{CLK\text{-}EXTERNO}=1/f_{CLK}=1/(16{,}384kHz)=15{,}384$ ms

T0SE=0 Utiliza el flanco ascendente si cuenta pulsos del pin T0CKI. Irrelevante, podemos contar flancos ascendentes o descendentes

PSA=0 Asigna el prescaler al Timer0

PS2:PS0 = 110. Usa un ratio 1:128 en el prescaler. Prescaler divide por 128 la frecuencia.

Por tanto, hacemos:

Registro OPTION_REG

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Bits de configuración del Timer TMR0

OPTION_REG = b'- - 1X0110'

Los bits 7 (RBPU) y 6 (INTEDG), en rojo, serán configurados con el puerto B.

Es decir: OPTION_REG = b'- - 1X0110'.

Con esto la subrutina Tparcial temporizará 1 segundo. Según los modelos ajustaremos la variable que cuenta el numero de temporizaciones (CONT) al número de segundos a temporizar.

Para este modelo PA2, Modelo A, sería de 85 segundos, luego CONT=85

c) Con la conexión del PIC que se indica, analizar si la configuración y el uso posterior de los puertos utilizados (PORTA y PORTB) es correcta para conseguir el propósito del programa. Justificar si el programa es correcto y, de no ser así, indicar los errores cometidos y los cambios a realizar en el programa para conseguir un correcto funcionamiento, coherente con el esquema eléctrico del circuito, justificando la respuesta.

La configuración correcta de los puertos sería:
Puerto A: RA2 salida. Resto de líneas serán entradas por seguridad, luego TRISA=b'xx111011'. Por ejemplo: TRISA='11111011'
Puerto B: RB0 entrada. Resto de líneas serán entradas por seguridad, luego TRISB=b'11111111'.

Además, activamos las resistencias de PULL UP. Particularmente, para usar las resistencias de PULL UP para leer correctamente el pulsador en RB0, haremos RBPU=0, con lo que:

OPTION_REG = b'0X - - - - - -'

Combinando con los valores necesarios de b) para OPTION_REG se tiene, por ejemplo:
OPTION_REG = b'00100110'
El uso de los puertos y los errores se muestran sobre el programa, en forma de comentarios

d) En el caso de que detecte algún error adicional en el desarrollo del programa, explicar en qué consiste y corregirlo. Tras revisar todos los errores indicar CLARAMENTE los cambios a realizar o, si se prefiere, escribir de nuevo el programa corregido con los cambios necesarios

Idem: ver los comentarios sobre el programa

```
;********** Grabador_EPROM.asm  ************************
            LIST p=16F877A
            INCLUDE <p16f877A.inc>
; Palabra de configuración del microcontrolador
            __CONFIG   _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_ON & _LVP_OFF

CONT        EQU 0x20

; Vector de RESET

            ORG 0x00
; Prepara contenido de PORTB inicial
; Configuración de puertos y temporizador
Inicio      clrf PORTB          bsf PORTA, 2 ;Para desactivar el relé, inicialmente se ha de poner RA2=1
            bsf STATUS,RP0      ; Bien: pasa al banco 1
            movlw b'00000000'   movlw b'11111011'; Según Apartado c: TRISA='11111011'
            movwf TRISB
            movlw b'00000000'   movlw b'11111111'; Según Apartado c: TRISB=b'11111111'
            movwf TRISA
            movlw b'10000110'   movlw b'00100110'; Según Apartados b y c: OPTION_REG = b'00100110'
            movwf OPTION_REG
            bcf STATUS, RP0     ; Bien: pasa al banco 0

; Programa principal
Principal   btfsc PORTB,1       btfss PORTB, 0 ;Se ha de saltar si RB0 no pulsado, es decir si RB0=1
            call Temporiza      ; Si no se cumple la condición (es decir RB0 es 1) llama a la temporización
            goto Principal      ; Si se cumple (es decir el bit 0 (RB0) está a uno, sigue esperando


; Subrutina de temporización sobre el pin RA2
Temporiza   bsf PORTB, 1        bcf PORTA, 2 ;Para activar el relé, se ha de poner RA2=0, saturando el PNP
            movlw d'35'         movlw d'85' ; Según apartado b) se carga CONT con los segundos a temporizar
            movwf CONT                      ; Los segundos a temporizar, varían según el modelo
Tparcial    bcf INTCON, T0IF    ; Bien: borra el flag de desborde del Timer0
            movlw d'120'        movlw d'128' ; Según apartado b) la precarga en el TMR0 es de 128
            movwf TMR0
Espera      btfsc INTCON, T0IF  btfss INTCON, T0IF ;Si el flag se activa (T0IF=1) salta a decfsz CONT
            goto Espera         ;si no, sigue esperando el desborde del TMR0 que marca fin de temporización
            decfsz CONT         ; Bien. Cuando la temporización parcial (de 1 s) acaba, decrementa la
            goto Tparcial       cuenta de temporizaciones pendientes (CONT) y salta si llega a cero.
            bsf PORTB,2         ; Si no salta (CONT no es cero) realiza otra temporización parcial Tparcial
            return              bsf PORTA, 2 ; Al acabar la temporización total pone RA2=1,  acaba la
; Fin de la subrutina de temporización  temporización de 85 segundos, el PNP pasa a corte y desactiva el relé
                                ; Bien: retorna de la subrutina Temporiza
            END
```