SISTEMAS BASADOS EN MICROPROCESADOR **Segundo control 2022-2023**

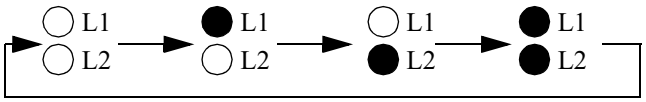
Apellidos:.....SOLUCIÓN.....

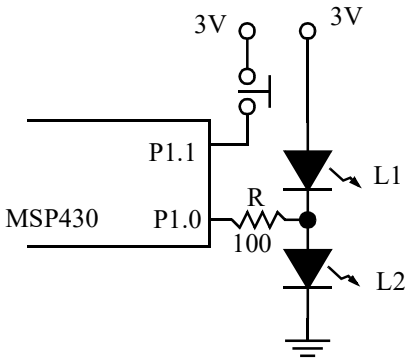
P1		P2	

Nombre:..... Puesto:.....

Duración 2:00 horas

1.- (5 PUNTOS) Considere el circuito de la figura. Haga un programa en ensamblador del MSP430 ***basado en multitarea cooperativa*** que inicialmente tiene los dos leds apagados. Cada vez que se acciona el pulsador, se cambian los leds encendidos según el ciclo:





Tenga en cuenta que no debe notarse cambio en el brillo aparente de los leds encendidos cuando se cambia de configuración.

Notas:

- Nótese que se pide un programa que use multitarea cooperativa, no que lea las teclas por interrupciones ni use el TimerA.
- Minimice el consumo del sistema manteniéndolo todo el tiempo posible en modo de bajo consumo.
- Considere el pulsador libre de rebotes, el perro guardián desactivado, los puertos desbloqueados, pila inicializada y LFXT en marcha con cristal de 32768Hz.
- Dispone del módulo `st.asm` y la subrutina `cmp32`.

SOLUCIÓN

Dado que el pulsador no tiene resistencia externa, hay que usar la interna. En este caso debe ser de pull-down.

La tensión de polarización de los leds es de 3V y puesto que la tensión de conducción de un led típico está en el entorno de 2V, no es posible encender los dos leds simultáneamente, pero sí apagarlos a la vez (poniendo el puerto como entrada). Si P1.0=0, se enciende L1 (y se apaga L2). Si P1.0=1, se enciende L2 (y se apaga L1). Para poder hacer la cuarta fase del ciclo será necesario encender los dos leds alternativamente en rápida secuencia. Para frecuencias superiores a 50Hz no se distingue el parpadeo, pero sí una reducción del brillo de los leds a la mitad. Es decir, habría que cambiar la salida de P1.0 entre 0 y 1.

Para evitar cambios de brillo en las fases 2 y 3 en las que sólo hay un led encendido, habrá que hacer que parpadeen con la misma frecuencia que antes, pero con un ciclo de trabajo del 50%. Esto es, habría que conmutar entre el puerto como entrada (los dos leds apagados) y el puerto como salida (uno de los leds encendido; si el valor de salida es 1, L2 ON; si el valor es 0, L1 ON).

Se fija la frecuencia de parpadeo de 50 Hz. Para el teclado se fija una frecuencia de 100Hz para tener una respuesta adecuada a las pulsaciones. La tarea de las teclas tendrá una estructura de máquina de estados con estados de tecla pulsada y no pulsada, para poder detectar los flancos. Habrá, además, los típicos estados internos de inicialización y error.

La tarea de parpadeo de los leds tendrá 4 estados, uno por cada posible estados de los leds. La variable `EstLeds` tendrá un número entre 0 y 3. En cada semiciclo se conmutará una variable de estado del puerto de los leds. Dependiendo del estado, será el valor de salida o la

propia condición de salida del mismo. Para simplificar el cambio de estado, también se ajustará la parte constante del estado cada vez, aunque sea redundante, como poner el puerto como salida en el estado 3 (dos leds encendidos). La variable `EstLeds` se escribirá en el proceso de lectura de las teclas (`Tecla`) y se leerá en el proceso de parpadeo de los leds (`Leds`).

```
;-------------------------------
; Datos de configuración
;-------------------------------
; *** Puertos de E/S ***
LEDS        .equ   BIT0
PUL         .equ   BIT1

;Frecuencia de los distintos procesos. En Hz
FTECLA      .equ   100          ;Frecuencia de escaneo de la tecla
FLEDS       .equ    50          ;Frecuencia de parpadeo de leds

; *** Frecuencia del System Timer ***
FTA         .equ   32768        ;Frecuencia del reloj del TA. Hz
FST         .equ   100          ;Frecuencia del SystemTimer. Hz

;-------------------------------
; Constantes calculadas
;-------------------------------
CCR0        .equ   FTA/FST-1    ;Valor a programar en CCR0 para stIni
PERTECLA    .equ   FST/FTECLA   ;Tiempo entre ejecuciones (TICs)
PERLEDS     .equ   FST/FLEDS    ;Tiempo entre ejecuciones (TICs)

;-------------------------------
; Variables
;-------------------------------
            .bss   teclaPE, 4   ;Próxima Ejecución del prceso de Parpadeo
            .bss   EstTecla, 1  ;Estado del proceso
EINI        .equ   0*2          ;Posibles valores de modo
ENOPUL      .equ   1*2          ;Posibles valores de modo
EPUL        .equ   2*2          ;Posibles valores de modo
EULTIMO     .equ   EPUL         ;Posibles valores de modo
            .bss   ledsPE, 4    ;Próxima Ejecución del prceso de Parpadeo
            .bss   EstLeds, 1   ;Estado del proceso
;-------------------------------------------------------------------------------
; main                                                                    v1.0
;-------------------------------------------------------------------------------
main        ;Inicializar SystemTimer
            mov.w  #CCR0, r12
            call   #stIni

            ;Inicializar procesos
            call   #Inicializa

superbucle  call   #Tecla       ;Tarea que lee la tecla
            call   #Leds        ;Tarea que hace paroadear los leds
            bis.w  #LPM3+GIE, sr ;Entrar en bajo consumo
            jmp    superbucle
            .intvec RESET_VECTOR, main
            .text

;-------------------------------------------------------------------------------
; Inicializa                                                            v1.0
;
; Inicializar proceso
;-------------------------------------------------------------------------------
Inicializa  clr.w  &teclaPE     ;Ejecutar desde el principio
            clr.w  &teclaPE+2
            mov.b  #EINI, &EstTecla

            clr.w  &ledsPE      ;Ejecutar desde el principio
            clr.w  &ledsPE+2
            clr.b  &EstLeds     ;Al inicio, dos leds apagados
            ret
```

```
;-------------------------------------------------------------------------------
; Proceso Tecla                                                          v1.0
;-------------------------------------------------------------------------------
Tecla       call   #stTime       ;R13:12 = Ahora
            mov.w  &teclaPE, r14 ;R15:R14=Instante de próxima ejecución
            mov.w  &teclaPE+2, r15
            call   #cmp32        ;Comparar. Toca?
            jlo    teclaFin      ;...no. Salir
            add.w  #PERTECLA, &teclaPE;...sí.Actualizar instante próxima ejecución
            adc.w  &teclaPE+2

            ;Proceso útil de la tarea
            mov.b  &EstTecla, r14
            cmp.b  #EULTIMO+1, r14
            jhs    MdError
            add.w  r14, pc
            jmp    MdIni         ;Inicialización
            jmp    MdNoPul       ;No pulsada
            jmp    MdPul         ;Pulsada

            ;Estado Error. Desactivar tarea
MdError     mov.w  #-1, &teclaPE ;Próxima ejecución... en el infinito
            mov.w  #-1, &teclaPE+1
            jmp    teclaFin
            ;Estado inicialización. Puerto como entrada con res de pulldown
MdIni       ;bic.b #PUL, &P1DIR  ;Puerto como entrada
            bis.b  #PUL, &P1REN  ;P1.0 resistencia ...
            bic.b  #PUL, &P1OUT  ;... de pulldown
            mov.b  #ENOPUL, &EstTecla;Pasar a esperar pulsación de tecla
            ;jmp   teclaFin      ;Descomentar para esperar un ciclo
            ;Estado tecla no pulsada. Esperar a que se pulse
MdNoPul     bit.b  #PUL, &P1IN   ;Tecla pulsada?
            jz     teclaFin      ;...no. Salir
            mov.b  #EPUL, &EstTecla;...sí, cambiar de estado a esperar liberación
            inc.b  &EstLeds      ;Incrementar estado de leds...
            and.b  #BIT1|BIT0, &EstLeds;...circularmente
            jmp    teclaFin
            ;Estado tecla pulsada. Esperar a que se suelte
MdPul       bit.b  #PUL, &P1IN   ;Tecla pulsada?
            jnz    teclaFin      ;...sí. Salir
            mov.b  #ENOPUL, &EstTecla;Cambiar de estado
            ;jmp   teclaFin

teclaFin    ret

;-------------------------------------------------------------------------------
; Proceso Leds                                                           v1.0
;-------------------------------------------------------------------------------
Leds        call   #stTime       ;R13:12 = Ahora
            mov.w  &ledsPE, r14  ;R15:R14=Instante de próxima ejecución
            mov.w  &ledsPE+2, r15
            call   #cmp32        ;Comparar. Toca?
            jlo    ledsFin       ;...no. Salir
            add.w  #PERLEDS, &ledsPE;...sí.Actualizar instante próxima ejecución
            adc.w  &ledsPE+2

            ;Proceso útil de la tarea
            mov.b  &EstLeds, r14
            rla.w  r14
            add.w  r14, pc
            jmp    LedsOff
            jmp    L1On
            jmp    L2On
            jmp    LedsOn

LedsOff     bic.b  #LEDS, &P1DIR ;Configuración puerto de leds de entrada
            jmp    ledsFin
```

```
L1On        bic.b  #LEDS, &P1OUT ;Configuración L1 encendido
            xor.b  #LEDS, &P1DIR ;Cambiar entre entrada (leds off) y salida (L1 on)
            jmp    ledsFin
L2On        bis.b  #LEDS, &P1OUT ;Configuración L2 encendido
            xor.b  #LEDS, &P1DIR ;Cambiar entre entrada (leds off) y salida (L2 on)
            jmp    ledsFin
LedsOn      bis.b  #LEDS, &P1DIR ;Configuración puerto de leds de salida
            xor.b  #LEDS, &P1OUT ;Cambiar de led
            ;jmp   ledsFin
ledsFin     ret
```

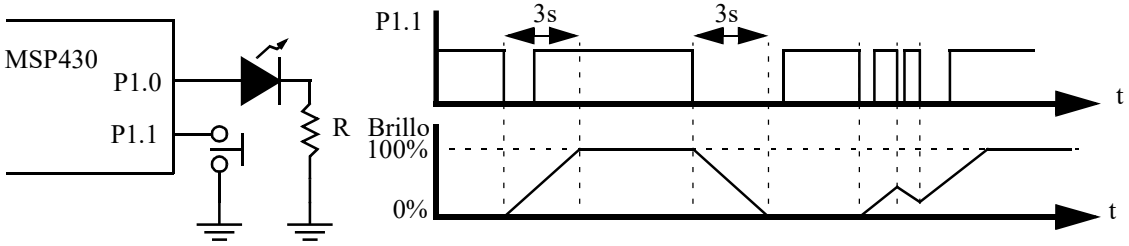
CRITERIO DE CORRECCIÓN

- Inicialización y prinicipal:10%
- Tecla: 40% (por nivel, -20%)
- Leds: 50% (brillo mal, -20%)

2.- (5 PUNTOS) Considere el circuito de la figura. Se trata de un sistema de luces con encendido y apagado retardado. Cada vez que se pulsa el switch, se conmuta el estado del led. Pero el encendido/apagado no es instantáneo, sino que se hace variando suavemente el brillo. Haga un programa de mínimo consumo en ensamblador del MSP430 que gestione con el TA0 y por interrupciones la conmutación del estado del led cada vez que se pulse el switch.

Notas:

- La pendiente es de 100%/3s. El apagado/encendido siempre se empezará desde el nivel de brillo actual del led y con la pendiente inversa (vea figura). Use una frecuencia de 128 Hz y 32 niveles de brillo.
- Considere el pulsador libre de rebotes, el perro guardián desactivado, los puertos desbloqueados, pila inicializada y LFXT en marcha con cristal de 32768Hz.
- La función primaria de P1.0 es TA0.1 (CCI1A para capturas y Out1 para salidas).
- La función primaria de P1.1 es TA0.2 (CCI2A para capturas y Out2 para salidas).



SOLUCIÓN

Inicialmente se configurará P1.1 como entrada con resistencia de *pull-up*. La tecla pulsada se detectará con un valor bajo (lógica negativa). El control del led es con lógica positiva (1 enciende, 0 apaga).

Ambos puertos se configurarán en su función primaria para que sean controlados por TA0 directamente. El CCR2 se configurará en modo de captura sensible a flanco de bajada en el canal CCIA con las interrupciones habilitadas. También sería válido usar la capacidad de interrupción del P1.1, ya que sólo se necesita la capacidad de detección del flanco de bajada, y no la medición del tiempo de pulsación.

Dado que $f_{TA} = 2^N \cdot f_{PWM}$, sustituyendo N=5 (32 niveles de brillo) y f_{PWM}=128, obtenemos f_{TA}=4096 Hz. Tomando ACLK como entrada de TA0, tenemos que usar un divisor de 32768/4096=8.

El CCR0 será el encargado de fijar la frecuencia de la señal PWM que gobernará el brillo del led (CCR0=32-1). El propio brillo será controlado por el CCR1 en modo RESET/SET (lógica positiva). Para poder variar dinámicamente el brillo del led se usará la interrupción del CCR0.

El control del brillo se hará con la ayuda de una variable de estado `Brillo`, que podrá valer `BR_EST` (0, brillo estático al 0% o al 100%), `BR_SUB` (1, brillo subiendo) o `BR_BAJ` (-1, brillo bajando). La pendiente de subida o bajada es de 100%/3s, lo que significa variar 32 tonos de brillo en 3 segundos. Para una señal PWM de 128Hz (128 ciclos en un segundo), significa variar 32 veces en 3*128 ciclos, o 1 vez cada 3*4 ciclos.

```
LED         .equ   BIT0
PUL         .equ   BIT1

FACLK       .equ   32768         ;Frecuencia de ACLK. En Hz
FPWM        .equ   128           ;Frecuencia del PWM. En Hz
RESPWM      .equ   32            ;Resolución del PWM. En unidades
FTA         .equ   FPWM * RESPWM ;Frecuencia del TA0. En Hz
DIVTA       .equ   FACLK/FTA     ;Divisor del TA0
VBRILLO     .equ   3*FPWM/RESPWM ;Velocidad de la pendiente de brillo
BR_EST      .equ   0             ;Brillo estático
BR_SUB      .equ   1             ;Brillo subiendo
BR_BAJ      .equ   -1            ;Brillo bajando
            .bss   Brillo, 1     ;Modo del brillo
            .bss   ContBr, 1     ;Contador de estados con mismo brillo

;-------------------------------------------------------------------------------
;main                                                                      v1.0
;-------------------------------------------------------------------------------
main        ;Inicializar variables
            mov.b  #BR_EST, &Brillo;Brillo estático (a 0%)
            mov.b  #VBRILLO, &ContBr;Contador de ciclos de brillo a máximo

            ;P1.1 entrada pulldown función 1 (TA0.2)
            ;bic.b #PUL, &P1DIR  ;Entrada
            bis.b  #PUL, &P1REN  ;Resistencia...
            bic.b  #PUL, &P1OUT  ;...de pulldown

            ;P1.0 salida función 1 (TA0.1)
            bis.b  #LED, &P1DIR  ;Salida
            bis.b  #PUL|LED, &P1SEL0;Función 1

            ;TA0.2 en modo captura con entrada CCIA en flanco de bajada. IRQ
            mov.w  #CAP|CCIS_0|CM_2|SCS|CCIE, &TA0CCTL2

            ;TA0.1 en modo comparación con salida PWM RESET-SET. Brillo a 0%
            mov.w  #OUTMOD_7, &TA0CCTL1
            clr.w  &TA0CCR1

            ;TA0.0 fijando excursión del TA0, modo UP. IRQ
            mov.w  #CCIE, &TA0CCTL0
            mov.w  #RESPWM-1, &TA0CCR0

            ;TA0 con ACLK/DIVTA modo UP
            mov.w  #DIVTA-1, &TA0EX0;Divisor extra
            mov.w  #TASSEL__ACLK|ID__1|MC__UP|TACLR, &TA0CTL
            bis.w  #LPM3|GIE, sr ;Ea, a dormir

;-------------------------------------------------------------------------------
; TA00ISR                                                                  v1.0
;
;ISR del CCR0. Cambiar el brillo si procede
;Si el brillo es estático, no hacer nada.
;Si es de subida/bajada, decrementar número de ciclos en este brillo.
;Cuando se llegue a 0, cambiar (subir o bajar) una unidad hasta llegar al máximo
;o al mínimo. En ese caso, pasar a modo estático
;-------------------------------------------------------------------------------
TA00ISR     tst.b  &Brillo       ;Qué modo está programado?
            jz     TA00ISRFin    ;...estático, salir
            dec.b  &ContBr       ;Decrementar contador de ciclos. Fin?
            jnz    TA00ISRFin    ;...no, salir
            mov.b  #VBRILLO, &ContBr;...sí, reponer contador de ciclos
            tst.b  &Brillo       ;Ver pendiente
            jn     PendNeg       ;...negativa
PendPos     inc.w  &TA0CCR1      ;...positiva. Incrementar brillo
            cmp.w  &TA0CCR0, &TA0CCR1;Ha llegado al 100%?
            jlo    TA00ISRFin    ;...no, salir
            jmp    FijarBr       ;...sí, pasar a modo estático
PendNeg     dec.w  &TA0CCR1      ;Decrementar brillo. Ha llegado al 0%?
```

```
            jnz    TA00ISRFin    ;...no, salir
FijarBr     mov.b  #BR_EST, &Brillo;...sí, pasar a modo estático
TA00ISRFin  reti

;-------------------------------------------------------------------------------
; TA01ISR                                                                  v1.0
;
;ISR del CCR2. Gestionar la pulsación de la tecla. En la libración no se actúa.
;Si el brillo está parado, se escoge la dirección (subir/bajar) en función del
; brillo (0%/100%).
;Si el brillo está cambiando, simplemente, cambiar la dirección.
;-------------------------------------------------------------------------------
TA01ISR     bic.w  #CCIFG, &TA0CCTL2;Borrar flag
            tst.b  &Brillo       ;Qué modo está programado?
            jnz    CambiaPend    ;...pendiente. Cambiar dirección
            tst.w  &TA0CCR1      ;Ver en qué extremo estamos
            jnz    TA01ISRNeg    ;...100%. Pendiente negativa
            mov.b  #BR_SUB, &Brillo;...0%. Pendiente positiva
            jmp    TA01ISRFin    ;Salir
TA01ISRNeg  mov.b  #BR_BAJ, &Brillo;Pendiente negativa
            jmp    TA01ISRFin    ;Salir
CambiaPend  inv.b  &Brillo       ;Cambiar signo. Primero Ca1
            inc.b  &Brillo       ;Después +1
TA01ISRFin  reti

            .intvecTIMER0_A0_VECTOR, TA00ISR
            .intvecTIMER0_A1_VECTOR, TA01ISR
            .intvecRESET_VECTOR, main
```

CRITERIO DE CORRECCIÓN

a) main: 10%
b) ISR tecla: 40%
c) ISR led: 50%