Universidad De Málaga

E.T.S Ingeniería Infórmatica

Ingeniería de Computadores, 3A

PRÁCTICA 1. LEDs y Música

Diseño con Microcontroladores

Francisco Javier Cano Moreno

16 de Febrero 2023

Contents

1 Control del parpadeo de un diodo.

1.1 Mediante "Polling".

1.1.1 Código.

```
; Enciende/Apaga el diodo cada segundo
;
    .list off
    .Include sfr.inc
    .list on
;
;----- DEFINICIÓN DE SÍMBOLOS 3DKUM16C/62PU ----------------------------------------
;
    VramTOP .equ 000400h  ; inicio de la RAM interna
    VramEND .equ 007CFFh  ; final de la RAM interna
    VIstack .equ 007CFFh  ; Direccion de la cima de la pila stack pointer
    VprogTOP .equ 0A0000h ; inicio del area de programa
    Vintbase .equ 0FA000h ; inicio tabla de interrupciones variable
    Vvector .equ 0FFFDCh  ; inicio tabla de interrupciones fija
    SB_base .equ 000380h  ; Base del direccionamiento SB relativo
;
;----- AREA DE DATOS RAM ------------------------------------------------------------
;
    .section memory,data
    .org VramTOP
;
;----- AREA DE PROGRAMA ROM  -------------------------------------------------------
    .section prog,code
    .org VprogTOP
;
;----- inicializacin de temporizador y puertos 3DKUM16C/62PU ----------------------
;
reset:
    bset PD2_0          ; ? (elije el que quieras del P2_?
    mov.b #80h,TA0MR    ; TA0 en modo timer con f32
    mov.w #2250,TA0     ; 2250 periodos de 27,77 ns x 32= 2 ms. (reloj de 36MHz)
    bset TA0S           ; arranca el contador
    mov.w #0,r2         ; repetir 10 veces
;
;----- PROGRAMA PRINCIPAL ----------------------------------------------------------
;
main:
    mov.w #0,r0     ; r0 cuenta de periodos de 2 ms.
apagado:
    btstc 3,TA0IC   ; Desbordamiento del TA0 (2msec)?
    jnc apagado
    add.w #1,r0     ; r0 = r0+1
```

```
    cmp.w #1500,r0  ; r0=1000?
    jnz apagado
    bnot P2_0       ; cambia la polaridad del diodo
    mov.w #0,r0
encendido:
    btstc 3,TA0IC   ; Desbordamiento del TA0 (2msec)?
    jnc encendido
    add.w #1,r0     ; r0 = r0+1
    cmp.w #1000,r0  ; r0=1000?
    jnz encendido
    bnot P2_0       ; cambia la polaridad del diodo
    add.w #1,r2
    cmp.w #10,r2
    jnz main
;
;----- INTERRUPCIÓN DE RESET (INICIALIZA PC) ----------------------------------------
;
    .section int_reset,romdata
    .org  Vvector+(8*4)
    .lword  reset
;
    .end
```

1.1.2 Análisis.

Para estudiar lo que hace el código nos tenemos que centrar en las funciones main y reset.

En primer lugar, la función reset se encarga de inicializar el timer (TA0), el cual salta cuando se han producido 2250 periodos donde cada periodo es de 22,77 x 32 ns, es decir, se puede decir que el timer dura 2 ms.
Por otro lado, se inicializa el pin P2_0, haciendo referencia al LED 0 de la placa.
Por último, se inicializa un registro que cuenta las veces que se enciende el led, ya que se tiene que encender 10 veces.

La función que se encarga de ejecutar es la función main y en ella tenemos dos partes: apagado y encendido, es decir, cuando el led está apagado y cuando el led está encendido.
Ambas funciones ejecutan el mismo código pero con una variación, cuando termina encendido se suma 1 al contador y se salta a main. Estas dos funciones se encargan de contar hasta 1500 en el caso del led apagado y hasta 1000 en encendido ya que el led debe estar apagado 3 s y encendido 2 s. Aquí es donde entra el "polling" o sondeo, es decir, se está comprobando todo el rato si el valor ha llegado a lo que queremos para que continúe y cambie el estado del led.

1.2 Mediante Interrupciones.

1.2.1 Código.

```
; Enciende/Apaga el diodo cada segundo
;
    .Include sfr.inc

;
;
;----- Definicin de Smbolos para 3DKUM16C/62PU -------------------------------------
;
    VramTOP .equ 000400h     ; inicio de la RAM interna
    VramEND .equ 007CFFh     ; final de la RAM interna
    VprogTOP .equ 0A0000h    ; inicio del area de programa
    Vvector .equ 0FFFDCh     ; inicio tabla de interrupciones fija
    Vintbase .equ 0FA000h    ; inicio tabla de interrupciones variable
    VIstack .equ 007CFFh     ; stack pointer
    SB_base .equ 000380h     ; base address of SB recative
;
;
;----- keep of RAM area ------------------------------------------------------------
;
    .section memory,data
    .org VramTOP
;
num:
    .blkw 2
data:
    .blkw 500
;
    .section prog,code
    .org VprogTOP
    .sb SB_base             ; assings aprovisional SB register value
    .sbsym mnum             ; place data in SB addressing mode
    .sbsym num
;
;
;----- clear of RAM ----------------------------------------------------------------
;
reset:
    ldc #VIstack,ISP                ; set Interrupt Stack Pointer
    ldc #SB_base,SB                 ; set SB register
    ldintb #Tabla_Vector_Usuario    ; set INterrupt TaBle register
;
    mov.w #0,R0                     ; 0 clear
    mov.w #(VramEND+1-VramTOP)/2,R3 ; number of times
    mov.w #VramTOP,A1               ; start address
```

```
    sstr.w

;---------------------- Inicializacin ---- (3DKM16C/62PU) --------------------------
    ldintb #Tabla_Vector_Usuari         ; Necesito dummy
    bset PD2_7                          ; P2_7 salida
    fclr I
    mov.b #3h,INT0IC
    fset I
;
;----- programa principal ------------------------------------------------------
;
main:
    jmp main

;
;----- manejador de la rutina de interrupcin del TA0 --------------------------------
;
sw_ta0:
    add.w #1,r0         ; desbordamiento (cada 2 ms)
    reit
;
;----- manejador de la rutina dummy -------------------------------------------------
;
dummy:
    reit

;
;-----manejador interrupcion boton-----------------------------------------------------
;
sw_int0:
    bnot P2_7           ; cambia la polaridad del diodo
    reit



;
;----- interrupcin del temporizador TA0 (tabla de vectores variable) ------------------
;
    .section int_ta0,romdata
    .org Vintbase
Tabla_Vector_Usuario:
    .lword dummy ; No0  Break Interrupt
    .lword dummy ; No1  Break Interrupt
    .lword dummy ; No2  Break Interrupt
    .lword dummy ; No3  Break Interrupt
    .lword dummy ; No4  Break Interrupt
    .lword dummy ; No5  Break Interrupt
    .lword dummy ; No6  Break Interrupt
```

```
.lword dummy ; No7  Break Interrupt
.lword dummy ; No8  Break Interrupt
.lword dummy ; No9  Break Interrupt
.lword dummy ; No10 Bus Clash Detect
.lword dummy ; No11 DMA0
.lword dummy ; No12 DMA1
.lword dummy ; No13 KEY IN Interrupt
.lword dummy     ; No14 A-D Interrupt
.lword dummy ; No15 UART2 Transmission Interrupt
.lword dummy ; No16 UART2 receive Interrupt
.lword dummy ; No17 UART0 Transmission Interrupt
.lword dummy ; No18 UART0 receive Interrupt
.lword 0FF900H ; No19 UART1 Transmission Interrupt
.lword 0ff900H ; No20 UART1 receive Interrupt
.lword dummy  ; No21 TimerA0 Interrupt
.lword dummy    ; No22 TimerA1 Interrupt
.lword dummy ; No23 TimerA2 Interrupt
.lword dummy ; No24 TimerA3 Interrupt
.lword dummy ; No25 TimerA4 Interrupt
.lword dummy ; No26 TimerB0 Interrupt
.lword dummy ; No27 TimerB1 Interrupt
.lword dummy ; No28 TimerB2 Interrupt
.lword sw_int0 ; No29 INIT0(Active Low) Interrupt
.lword dummy ; No30 INIT1(Active Low) Interrupt
.lword dummy ; No31 INIT2(Active Low) Interrupt
.lword dummy ; No32 S/W Interrupt
.lword dummy ; No33 S/W Interrupt
.lword dummy ; No34 S/W Interrupt
.lword dummy ; No35 S/W Interrupt
.lword dummy ; No36 S/W Interrupt
.lword dummy ; No37 S/W Interrupt
.lword dummy ; No38 S/W Interrupt
.lword dummy ; No39 S/W Interrupt
.lword dummy ; No40 S/W Interrupt
.lword dummy ; No41 S/W Interrupt
.lword dummy ; No42 S/W Interrupt
.lword dummy ; No43 S/W Interrupt
.lword dummy ; No44 S/W Interrupt
.lword dummy ; No45 S/W Interrupt
.lword dummy ; No46 S/W Interrupt
.lword dummy ; No47 S/W Interrupt
.lword dummy ; No48 S/W Interrupt
.lword dummy ; No49 S/W Interrupt
.lword dummy ; No50 S/W Interrupt
.lword dummy ; No51 S/W Interrupt
.lword dummy ; No52 S/W Interrupt
.lword dummy ; No53 S/W Interrupt
```

```
    .lword dummy ; No54 S/W Interrupt
    .lword dummy ; No55 S/W Interrupt
    .lword dummy ; No56 S/W Interrupt
    .lword dummy ; No57 S/W Interrupt
    .lword dummy ; No58 S/W Interrupt
    .lword dummy ; No59 S/W Interrupt
    .lword dummy ; No60 S/W Interrupt
    .lword dummy ; No61 S/W Interrupt
    .lword dummy ; No62 S/W Interrupt
    .lword dummy ; No63 S/W Interrupt

;
    .section inter,romdata
    .org Vvector+(8*4)
    .lword  reset


;Para la placa 3DKM16C/62PU la direccin asignada a las interrupciones 19 y 20
;de transmisin y recepcin de la UART1 (del programa monitor) en la tabla de
;vectores variables es 0FF900H
    .end
```

1.2.2 Análisis.

Para comentar como funciona el código tenemos que tener en cuenta varias partes: inicialización de variables e interrupciones, la función main y los manejadores de interrupción.

En primer lugar tenemos la parte de inicialización, donde cargamos la tabla con los vectores de interrupción, la cual se encarga de asignar un manejador a cada interrupción; configuramos el led número 7 como salida y activamos la interrupción INT0IC por botón.

En segundo lugar, aparece la función main, en la cual creamos un bucle infinito que hace que el programa no termine y pueda recibir interrupciones cada vez que se pulsa un botón.

Por último, tenemos los manejadores de interrupción dummy y sw_ta0. Dummy se encarga de retornar a la función principal y se asigna a todas las interrupciones por defecto para que no se ejecute nada en caso de saltar una interrupción inesperada. En el caso de sw_ta0, se encarga de cambiar la polaridad del led, por lo que cada vez que se pulsa el botón se enciende o se apaga, y se asigna a la interrupción nº 29, es decir, la de INIT0.

2 Reproducción de una canción.

2.1 Código

```
;*****************************************************************************
;* *
;*    Prctica 2.3.1: Elaboracin de cdigo de un fragmento de una cancin *
;* *
;*   Cancin escogida: Imperial March *
;* (Tema de Darth Vader de La Guerra de las Galaxias) *
;* *
;*****************************************************************************
;
;----- include of sfr file --------------------------------------------------
;
    .list off
    .include sfr.inc
    .list on
;
;
;----- Definicin de Smbolos para 3DKUM16C/62PU ------------------------------
;
    VramTOP .equ 000400h     ; inicio de la RAM interna
    VramEND .equ 007CFFh     ; final de la RAM interna
    VprogTOP .equ 0A0000h    ; inicio del area de programa
    Vvector .equ 0FFFDCh     ; inicio tabla de interrupciones fija
    Vintbase .equ 0FA000h    ; inicio tabla de interrupciones variable
    VIstack .equ 007CFFh     ; stack pointer
    SB_base .equ 000380h     ; base address of SB recative
;
;
;----- keep of RAM area -----------------------------------------------------
;
    .section memory,data
    .org VramTOP
;
num:
    .blkw 2
data:
    .blkw 500
;
    .section prog,code
    .org VprogTOP
    .sb SB_base             ; assings aprovisional SB register value
    .sbsym mnum             ; place data in SB addressing mode
    .sbsym num
;
```

```
;
;----- clear of RAM ------------------------------------------------------------
;
reset:
    ldc #VIstack,ISP                ; set Interrupt Stack Pointer
    ldc #SB_base,SB                 ; set SB register
    ldintb #Tabla_Vector_Usuario    ; set INterrupt TaBle register
;
    mov.w #0,R0                     ; 0 clear
    mov.w #(VramEND+1-VramTOP)/2,R3 ; number of times
    mov.w #VramTOP,A1               ; start address
    sstr.w
;
;
;----- initiallize -------------------------------------------------------------
;
    mov.w #0,PD8 ; P8,P9 input direction
;
    mov.b #80h,TA0MR ; set TA0MR (timer mode)10->f/32
    mov.w #0fFF0h,TA0 ; set TA0Valor que ponemos al contador
    mov.b #7,TA0IC        ;prioridad de captura de finalizacion del contador
    mov.b #04h,TA4MR ; set TA4MR (timer mode)Se arranca la cuenta cuando esta a 0 ta4in
    mov.w #0,TA4    ; set TA4
    mov.b #11h,TABSR ; timer start
;
    fset I
;
;
;----- main program ------------------------------------------------------------
;
main:
    jsr data_set
?:
    jmp ?-
;
;
;----- music data set ----------------------------------------------------------
; carga el cdigo de la cancin en memoria
;
data_set:
    mov.w #0,r0 ; inicializa ro a 0
    mov.w #0,r3 ; inicializa r3 a 0
    mov.w #0,a1 ; inicializa a1 a 0
    lde.w music1_count,r2 ; carga en r2 el valor de music1_count (256)
    mov.w r2,num ; mueve este valor a num
    shl.w #1,num
loop: ; inicio del bucle
```

```
; en cada iteracin cargaremos una palabra de la secuencia musical
    mov.w a1,a0 ; mueve a1 a a0
; con a0 como desplazamiento, carga una palabra del cdigo musical en r1
    lde.w music1_data[a0],r1
; carga los 4 bits ms significativos de r1h en los cuatro menos de r0l
    movhl r1h,r0l
    jsr ram_store ; salta a la subrutina ram store
; carga los 4 bits menos significativos de r1h en los cuatro menos de r0l
    movll r1h,r0l
    jsr ram_store ; salta a la subrutina ram store
; carga los 4 bits ms significativos de r1l en los cuatro menos de r0l
    movhl r1l,r0l
    jsr ram_store ; salta a la subrutina ram store
; carga los 4 bits menos significativos de r1l en los cuatro menos de r0l
    movll r1l,r0l
    jsr ram_store ; salta a la subrutina ram store
    add.w #2,a1 ; incrementa 2 unidades el registro a1
; decrementa una unidad el registro r2 y si no es cero, vuelve al inicio del bucle
    sbjnz.w #1,r2,loop
    rts ; volvemos a la llamada del main
;
ram_store:
; cada vez que se ejecuta, en r0l estar guardada una de las 16 notas que
;usamos en la cancin

; salvamos el valor de a1 en la pila
    push.w a1
    mov.b r0l,a0 ; movemos r0l a a0
    shl.w #1,a0 ; multiplicamos por dos el valor del registro a0
; cargamos la frecuencia de la nota dada por el desplazamiento a0 en a1
    lde.w sound_data[a0],a1
    mov.w r3,a0 ; movemos r3 a a0
    shl.w #1,a0 ; multiplicamos a0 por dos
    mov.w a1,data[a0] ; movemos a1 a la zona de memoria con desplazamiento a0
    add.w #1,r3 ; incrementamos el valor de r3
    pop.w a1 ; recuperamos el valor original de a1
    rts ; volvemos a la llamada del bucle
;
;
;----- sound -----------------------------------------------------------------
;
sound:
    push.w a0
    mov.w num+2,a0
    mov.w data[a0],TA4
    add.w #2,num+2
    cmp.w num,num+2
```

```
    jne ?+
    mov.w #0,num+2
?:
    pop.w a0
    reit
;



;
;------- NOTAS Y SU DURACIN DE LA CANCIN
;-------------------------------------------------------------------------------
music1_count:
    .word 44
music1_data:
    .word 05555h, 05555h, 05555h, 05555h
    .word 01111h, 01111h, 01111h, 01111h
    .word 00333h, 04445h, 05550h, 00000h
    .word 03333h, 05554h, 04433h, 03300h
; 64
    .word 00AAAh, 0A000h, 0AAAAh, 0000Ah
    .word 0AAA0h, 000BBh, 0BB00h, 06033h
    .word 03333h, 00011h, 01100h, 06044h
    .word 04444h, 04400h, 00000h, 00000h
; 128
    .word 0FFFFh, 0FF00h, 04444h, 00040h
    .word 0FFFFh, 0FF00h, 00EEEh, 0E0DDh
    .word 00CCBh, 0B0CCh, 00000h, 00040h
    .word 00999h, 09000h, 08888h, 00770h
; 192
    .word 06655h, 00770h, 00000h, 01000h
    .word 03333h, 00002h, 02220h, 07044h
    .word 04444h, 00001h, 01110h, 06044h
    .word 04444h, 04400h, 00000h, 00000h
; 256
;
;------- NOTAS QUE SE UTILIZAN EN LA CANCIN
;-------------------------------------------------------------------------------

sound_data:
    .word 0000h, 7D9AH, 765FH, 6993H ; Silencio, Mib, Mi, Fa#
    .word 63A6H, 58C7H, 53D3H, 4F34H ; Sol, La, Sib, Si
    .word 4AB0H, 4682H, 4280H, 3ECDH ; Do, Do#, Re, Mib
    .word 3B5EH, 37F7H, 34CAH, 31D3H ; Mi, Fa, Fa#, Sol
;
;
;----- manejador de la rutina dummy -------------------------------------------
```

```
;
dummy:
    reit
;
;----- vector table ------------------------------------------------------------
;
    .section uniter,romdata
    .org Vintbase

Tabla_Vector_Usu:

    .lword dummy ; No0  Break Interrupt
    .lword dummy ; No1  Break Interrupt
    .lword dummy ; No2  Break Interrupt
    .lword dummy ; No3  Break Interrupt
    .lword dummy ; No4  Break Interrupt
    .lword dummy ; No5  Break Interrupt
    .lword dummy ; No6  Break Interrupt
    .lword dummy ; No7  Break Interrupt
    .lword dummy ; No8  Break Interrupt
    .lword dummy ; No9  Break Interrupt
    .lword dummy ; No10 Bus Clash Detect
    .lword dummy ; No11 DMA0
    .lword dummy ; No12 DMA1
    .lword dummy ; No13 KEY IN Interrupt
    .lword dummy     ; No14 A-D Interrupt
.lword dummy ; No15 UART2 Transmission Interrupt
    .lword dummy ; No16 UART2 receive Interrupt
    .lword dummy ; No17 UART0 Transmission Interrupt
    .lword dummy ; No18 UART0 receive Interrupt
    .lword 0FF900H ; No19 UART1 Transmission Interrupt
    .lword 0ff900H ; No20 UART1 receive Interrupt
    .lword sound  ; No21 TimerA0 Interrupt
    .lword dummy    ; No22 TimerA1 Interrupt
    .lword dummy ; No23 TimerA2 Interrupt
    .lword dummy ; No24 TimerA3 Interrupt
    .lword dummy ; No25 TimerA4 Interrupt
    .lword dummy ; No26 TimerB0 Interrupt
    .lword dummy ; No27 TimerB1 Interrupt
    .lword dummy ; No28 TimerB2 Interrupt
    .lword dummy ; No29 INIT0(Active Low) Interrupt
    .lword dummy ; No30 INIT1(Active Low) Interrupt
    .lword dummy ; No31 INIT2(Active Low) Interrupt
    .lword dummy ; No32 S/W Interrupt
    .lword dummy ; No33 S/W Interrupt
    .lword dummy ; No34 S/W Interrupt
    .lword dummy ; No35 S/W Interrupt
```

```
    .lword dummy ; No36 S/W Interrupt
    .lword dummy ; No37 S/W Interrupt
    .lword dummy ; No38 S/W Interrupt
    .lword dummy ; No39 S/W Interrupt
    .lword dummy ; No40 S/W Interrupt
    .lword dummy ; No41 S/W Interrupt
    .lword dummy ; No42 S/W Interrupt
    .lword dummy ; No43 S/W Interrupt
    .lword dummy ; No44 S/W Interrupt
    .lword dummy ; No45 S/W Interrupt
    .lword dummy ; No46 S/W Interrupt
    .lword dummy ; No47 S/W Interrupt
    .lword dummy ; No48 S/W Interrupt
    .lword dummy ; No49 S/W Interrupt
    .lword dummy ; No50 S/W Interrupt
    .lword dummy ; No51 S/W Interrupt
    .lword dummy ; No52 S/W Interrupt
    .lword dummy ; No53 S/W Interrupt
    .lword dummy ; No54 S/W Interrupt
    .lword dummy ; No55 S/W Interrupt
    .lword dummy ; No56 S/W Interrupt
    .lword dummy ; No57 S/W Interrupt
    .lword dummy ; No58 S/W Interrupt
    .lword dummy ; No59 S/W Interrupt
    .lword dummy ; No60 S/W Interrupt
    .lword dummy ; No61 S/W Interrupt
    .lword dummy ; No62 S/W Interrupt
    .lword dummy ; No63 S/W Interrupt

;
    .section inter,romdata
    .org Vvector+(8*4)
    .lword reset
;
;
;----- program end ------------------------------------------------------------
;
    .end
;
```

2.2 Explicación

Para entender cómo funciona este código debemos entender como se van cargando las notas y su respectiva duración y, por otro lado, las interrupciones que se usan.

En primer lugar, tenemos una lista de 16 notas guardada en *sound_data* donde 0 sería el silencio y F sería un Sol más agudo. Después, tenemos la lista *music1_data*, de la cual se van cargando palabras, de la cual se van utilizando los 4 bits más significativos con un desplazamiento, por lo que se puede recorrer cada palabra. Cada uno de estos valores hace referncia a una nota de la anterior lista, consiguiendo así que se reproduzca la secuencia de notas que queramos. En este caso, en las primeras 44 notas hay un intento de ejecución de la canción de Juego de Tronos. Además tenemos una variable *music1_count* que indica el número de notas que se tocan.

Por otro lado, tenemos la interrupción TA0, que hace que cada vez que salte se ejecuten las notas que haya en memoria, ya que las notas se van cargando a memoria poco a poco.