

## SISTEMAS BASADOS EN MICROPROCESADORES (SBM)

Grado de Ingeniería Informática  
Doble Grado Informática-Matemáticas  
Escuela Politécnica Superior – UAM

PARCIAL 3 / FINAL MAYO - Curso 18-19

### ÓRGANO MUSICAL

La siguiente aplicación es la base para implementar un órgano musical basado en un PC. Para ello, necesitamos poder generar ondas de diferentes frecuencias y duraciones. Cada onda se corresponde con una determinada nota musical. Las notas musicales se oirán a través del altavoz interno del PC, que está conectado al TIMER (8254) convenientemente configurado para generar **una onda cuadrada periódica** de frecuencia igual a la de la nota musical (lo más aproximadamente posible).

Cuando se pulse una tecla del teclado (en el código empezamos utilizando la tecla D en minúsculas), se producirá o generará la nota musical con una duración determinada (**void GenerarNota (int frec, int duracion)**). Cuando se termine de generar la nota, el programa principal volverá a esperar que pulsemos la tecla si queremos que vuelva a sonar. La idea es utilizar un conjunto de teclas que emulen un teclado de órgano y que cada una de las teclas tenga asignada una determinada nota, codificadas en: frecuencia en Hz y duración en número de intervalos de 50 ms

Para poder controlar la duración de la nota (el tiempo que está sonando el altavoz tras pulsar una tecla) utilizaremos la capacidad del RTC (**MC146818**) para generar interrupciones periódicas con una frecuencia (o período) determinada. En nuestra aplicación necesitamos generar una interrupción cada 50 ms, lo que nos permitirá establecer duraciones para las notas con una buena precisión. La duración vendrá determinada por el número de interrupciones generadas desde el comienzo de la onda cuadrada producida por el TIMER hasta alcanzar el valor que equivale a la duración deseada (duración en ms = parámetro "duracion" \* 50 ms). Recuerde que especificaremos la duración de la nota indicando el número de interrupciones del RTC a contabilizar antes de dejar de generar la onda cuadrada con el TIMER.

La aplicación será un programa (.EXE) escrito en lenguaje C (programa principal) y en ensamblador del x86 (rutinas llamadas desde el programa principal y otras que son necesarias pero no se llaman desde el mismo). Además, se usarán 3 variables globales definidas en el programa principal y que se utilizarán en las subrutinas escritas en ensamblador. Recuerde que debe considerar los convenios del compilador de C para escribir las rutinas en ensamblador y hacer uso de las variables globales.

### Programa Principal (en un fichero .c)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

extern void Configurar_RTC();
extern void Configurar_TIMER();
extern void GenerarNota(int, int);

int IniNota = 0;
int NumInt = 0;

void main(void) {
    char tecla = ' ';

    Configurar_RTC();
    Configurar_TIMER();

    while (tecla != 'q')
    {
        tecla = getc();
        if (tecla == 'd')
            GenerarNota(400,10);
    }
}
```

### Rutinas en Ensamblador (en un fichero .asm)

```
_codigo_rutinas segment byte public
    assume cs:_codigo_rutinas, ds:_DATA

_Configurar_RTC proc far
    .....
_Configurar_RTC endp

_Configurar_TIMER proc far
    .....
_Configurar_TIMER endp

_GenerarNota proc far
    .....
_GenerarNota endp

Temporizador proc far
    .....
Temporizador endp

public _Configurar_RTC
public _Configurar_TIMER
public _GenerarNota
extrn _IniNota : WORD
extrn _NumInt : WORD

_codigo_rutinas ends
end
```

## SISTEMAS BASADOS EN MICROPROCESADORES (SBM)

Grado en Ingeniería Informática Doble  
Grado Informática-Matemáticas Escuela  
Politécnica Superior – UAM

### PARCIAL 3 / FINAL MAYO - Curso 18-19 SOLUCIÓN PREGUNTAS

**P1. A la vista del código fuente (.c y .asm) y del enunciado del problema, escriba el código de la rutina en ensamblador del programa principal *\_Configurar\_RTC*. Añada comentarios para generar un código claro. Justifique su respuesta. (2.5 p.)**

```
_Configurar_RTC proc far
    PUSH    AX ES
    XOR     AX, AX
    MOV     ES, AX
    CLI                                           ;Inhibir las interrupciones CPU
    ;Inicializar vector de interrupción 70H
    MOV     ES:[70H*4], OFFSET Temporizador
    MOV     ES:[70H*4 + 2], SEG Temporizador
    STI                                           ;Habilitar las interrupciones CPU
    MOV     AL, 0AH                             ;Programar RTC (20 Hz. aprox: DV=02H / RS=0Bh)
    OUT     70H, AL
    MOV     AL, 2BH
    OUT     71H, AL
    MOV     AL, 0BH                             ;Programar IRQ de RTC
    OUT     70H, AL
    IN      AL, 71H
    OR      AL, 40H                             ;PIE = 1 para habilitar interrupciones periódicas
    MOV     AH, AL
    MOV     AL, 0BH
    OUT     70H, AL
    MOV     AL, AH
    OUT     71H, AL
    IN      AL, 0A1H                           ;Habilita interr. PIC-1 (IR0) (bit 0 del IMR a 0)
    AND     AL, 0FEH
    OUT     0A1H, AL
    POP     ES AX
    RET
_Configurar_RTC endp
```

**P2. A la vista del código fuente (.c y .asm) y del enunciado del problema, escriba el código de la rutina en ensamblador del programa principal *\_Configurar\_TIMER*. Esta rutina debe configurar el hardware para que la onda cuadrada generada por el TIMER llegue hasta el altavoz del PC. Añada comentarios para generar un código claro. Justifique su respuesta. (1.5 p.)**

```

_Configurar_TIMER proc far
    PUSH AX
    MOV AL, 10110110b ;Control word: SC=2 | RW=3 | M=3 | BCD=0
    OUT 43h, AL ;Send control word
    IN AL, 61H
    AND AL, 0FEH ;Bit 0 a 0 para GATE=0 y bit 1 a 1 para AND del altavoz
    OR AL, 02H
    OUT 61H, AL
    POP AX
    RET
_Configurar_TIMER endp

```

**P3. De acuerdo con el enunciado y el código fuente del programa (.c y .asm), escriba el código en ensamblador de la rutina *Temporizador*. Esta rutina incrementará la variable global *NumInt* en una unidad con cada interrupción del RTC, siempre que la variable global *IniNota* valga 1. Añada comentarios para generar un código claro. (2.5p)**

```

Temporizador proc far
    STI ;Permitir anidamiento de interrupciones
    PUSH AX
    CMP _IniNota, 1
    JNE fin
    INC _NumInt
fin: MOV AL, 0CH ;Leer el registro C para borrar el flag de interrupción
    OUT 70H, AL
    IN AL, 71H ;
    MOV AL, 20H ;Enviar un EOI (20H) a cada PIC
    OUT 20H, AL
    OUT 0A0H, AL
    POP AX
    IRET
Temporizador endp

```

**P4. Teniendo en cuenta el enunciado y el código fuente del enunciado (.c y .asm), escribe el código en ensamblador de la rutina *GenerarNota (int freq, int duracion)*. Esta rutina será la encargada de iniciar la generación de la onda cuadrada, configurando su frecuencia (parámetro *freq*), inicializar la temporización de la nota musical (variable *IniNota*) y controlar cuando ha transcurrido la duración programada (parámetro *duracion*), finalizando la generación de la onda y reseteando las variables globales pertinentes. Añada comentarios para generar un código más claro. (3.5 p.)**

```

_GenerarNota proc far
    PUSH BP
    MOV BP, SP
    PUSH AX BX CX DX
    MOV CX, [BP+6] ;Frecuencia en Hz de la nota
    MOV BX, [BP+8] ;Duración como número de interrupciones a contar
    MOV DX, 0012H ;Escribimos en DX:AX el valor del dividendo (1193182 Hz)
    MOV AX, 34DEH
    DIV CX ;CX contiene el divisor pasado como parámetro (frecuencia nota)
    ;La división devuelve en AX el cociente y en DX el resto (no usar)
    OUT 42h, AL ;Send low byte of initial count value (AL).
    OUT 42h, AH ;Send high byte of initial count value (AH)

```

```

        MOV     _IniNota, 1      ;Activa conteo del Temporizador
        IN      AL, 61H
        OR      AL, 03H          ;Bit 0 a 1 para GATE=1 y bit 1 a 1 para AND que conecta al altavoz
        OUT     61H, AL
Wait:   CMP     _NumInt, BX
        JE      Salir
        JMP     Wait
Salir:   MOV     _IniNota, 0
        MOV     _NumInt, 0
        IN      AL, 61H
        AND     AL, 0FEH         ;Bit 0 a 0 para GATE=0
        OUT     61H, AL
        POP     DX CX BX AX BP
        RET
_GenerarNota endp

```