SISTEMAS BASADOS EN MICROPROCESADORES (SBM)

Grado en Ingeniería Informática
Escuela Politécnica Superior – UAM

FINAL MAYO
Curso 16-17

ENUNCIADO

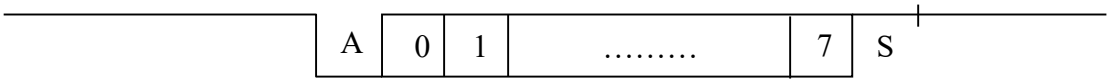
SIMULACIÓN DE UN PUERTO SERIE ASÍNCRONO (UART) BASADA EN EL PUERTO LPT1

Se necesita enviar datos a un equipo de comunicaciones (router/firewall) a través de un puerto serie asíncrono compatible RS-232C (UART) disponible en dicho equipo. Para ello se ha decidido utilizar un ordenador compatible PC que no dispone de una UART pero sí de un puerto paralelo (LPT1). Se ha decidido simular el funcionamiento del mismo sólo para transmisión de datos (no recepción) utilizando el pin STROBE del LPT1.

La UART del equipo de comunicaciones está configurada como 8,N,1 (caracteres de 8 bits, No Paridad, 1 bit de parada, "S") y una velocidad de transferencia de 4800 baudios (1 baudio = 1 bit / s). Los caracteres se transmiten en serie, empezando por el bit menos significativo (bit 0). Para indicar el comienzo de una transmisión, se envía un bit de arranque cuyo valor es "0" (bit "A"). La línea de transmisión se mantiene en estado alto (a "1") mientras no se realiza una transmisión. Al final de la misma, vuelve a dicho estado.

Para simular la UART en el PC usando el puerto LPT1, utilizaremos las interrupciones del RTC debidamente configurado para generar unas 4800 interrupciones por segundo (lo más aproximado posible). Con cada interrupción se enviará un bit a través del bit STROBE del LPT1. La transmisión comenzará cuando se llame a la función ***void TransmitirDato()***, que pondrá a "1" una variable global ***transmitiendo***, así como a "0" otra variable global ***posicion***, que servirá para saber cuál es el bit (b0 a b7) que toca transmitir en cada interrupción del RTC. Cuando ***posicion*** sea 9, tras transmitir el bit de parada (S) que siempre es "1", las variables ***transmitiendo*** y ***posicion*** deben ser puestas a "0" (se han transmitido todos los bits del carácter más el bit de parada).

El formato de un carácter en las transmisiones serie a través de una UART es el siguiente:



Considere que el pin del conector vinculado al bit STROBE está conectado a un driver de línea que transforma las tensiones digitales de 0 v. y 5 v. en -10 v. y +10 v. respetando el estándar RS-232C. A efectos de este ejercicio, esto no afecta al desarrollo del software.

Programa principal en C del programa .EXE

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

extern void far ConfigurarRTC();
extern void far InstalarInterrupcionRTC();
extern void far DesinstalarInterrupcionRTC();
extern void far TransmitirDato();

char dato;
char transmitiendo;
char posicion;

void main(void)
{
  ConfigurarRTC();
  InstalarInterrupcionRTC();
  transmitiendo = 0;

  while(true)
  {
   if (transmitiendo == 0)
    {
       dato = getc();
       if (dato == 27) //ESC
              break;
      else
              TransmitirDato();
    }
  }
  DesinstalarInterrupcionRTC(); exit(0);
 } /* Fin del Programa Principal */
```

Rutinas en ensamblador del 8086 que forman parte del programa .EXE

```
_codigo_rutinas segment byte public
      assume cs:_codigo_rutinas, ds: _BSS
extrn _dato : BYTE
extrn _posicion : BYTE
extrn _transmitiendo : BYTE

_ConfigurarRTC proc far
       public _ConfigurarRTC
_ConfigurarRTC endp

_InstalarInterrupcionRTC proc far
       public _InstalarInterrupcionRTC
_InstalarInterrupcionRTC endp

_DesinstalarInterrupcionRTC proc far
       public _DesinstalarInterrupcionRTC
_DesinstalarInterrupcionRTC endp

_DesinstalarDriver proc far
       public _DesinstalarDriver
_DesinstalarDriver endp

_TransmitirDato proc far
       public _TransmitirDato
_TransmitirDato end

_RutServRTC proc far
_RutServRTC endp

_codigo_rutinas ends
end
```

SISTEMAS BASADOS EN MICROPROCESADOR (SBM)

Grado en Ingeniería Informática Escuela Politécnica Superior – UAM

FINAL MAYO
Curso 16-17

NOMBRE :_____ **DNI :** _____
APELLIDOS : _____

P1. Escriba el código de la rutina ***_ConfigurarRTC*** en ensamblador para que programe las interrupciones del RTC con una frecuencia lo más cercana posible a 4800 interrupciones por segundo. Debe también configurar las máscaras de interrupciones de los PIC por donde entran las interrupciones del RTC. No debe habilitar la máscara de interrupciones periódicas del RTC. Tenga en cuenta la información del enunciado. **(2 p.)**

```
_ConfigurarRTC      proc far
                    push ax

                    ;Programar frecuencia interrupciones RTC a 4800 Hz aprox. (4096 Hz)
                    mov al, 0Ah
                    out 70h, al
                    mov al, 24h     ; programa bits DV y RS (0010 0100)
                    out 71h, al

                    ;Habilitar interrupciones por IR2 en PIC-0
                    in al, 21h
                    and al, 11111011b
                    out 21h, al

                    ;Habilitar interrupciones por IR0 (RTC) en PIC-1
                    in al, 0A1h
                    and al, 11111110b
                    out 0A1h, al

                    pop ax
                    ret
_ConfigurarRTC      endp
```

P2. Escriba el código de la rutina ***_InstalarInterrupcionRTC*** en ensamblador para que inicialice el vector de interrupción vinculada en el PC a las interrupciones del RTC. Esta rutina debe también habilitar las interrupciones periódicas del RTC (máscara local). Tenga en cuenta la información del enunciado. **(2 p.)**

```
_InstalarInterrupcionRTC    proc far
                            push ax es

                            ;Inicializar vector 70h con dirección de _RutServRTC
                            xor ax, ax
                            mov es, ax
                            cli
                            mov es:[70h*4], offset _RutServRTC
                            mov es:[70h*4+2], seg _RutServRTC
                            sti

                            ;Habilitar interrupciones periódicas (PIE a 1)
                            mov al, 0Bh
                            out 70h, al
                            in al, 71h
                            or al, 01000000b
```

```
                        mov  ah, al
                        mov al, 0Bh
                        out 70h, al
                        mov al, ah
                        out 71h, al
                        pop es ax
                        ret
_InstalarInterrupcionRTC endp
```

P3. Escriba el código de la rutina ***_DesinstalarInterrupcionRTC*** en ensamblador para que desinstale el vector de interrupción vinculada en el PC a las interrupciones del RTC (valor por defecto de un vector FFFFh:FFFFh). Esta rutina debe también deshabilitar las interrupciones periódicas del RTC (máscara local). Tenga en cuenta la información del enunciado. **(2 p.)**

```
_DesinstalarInterrupcionRTC proc far
                        push ax es

                        ;Inicializar vector 70h con dirección FFFFh:FFFFh
                        xor ax, ax
                        mov es, ax
                        cli
                        mov word ptr es:[70h*4], 0FFFFh
                        mov word ptr es:[70h*4+2], 0FFFFh
                        sti

                        ;Deshabilitar interrupciones periódicas (PIE a 0)
                        mov al, 0Bh
                        out 70h, al
                        in al, 71h
                        and al, 10111111b
                        mov ah, al
                        mov al, 0Bh
                        out 70h, al
                        mov al, ah
                        out 71h, al

                        pop es ax
                        ret
_DesinstalarInterrupcionRTC endp
```

P4. Escriba el código de la rutina de servicio ***_RutServRTC*** en ensamblador para que transmita cada uno de los bits del dato (empezando por el bit 0) a través del pin STROBE del LPT1 con cada interrupción del RTC. La transmisión de un bit dependerá de que la variable ***_transmitiendo*** valga 1 (mientras valga 0 no hará nada). Cada vez que un bit sea transmitido, la variable ***_posicion*** debe ser incrementada en una unidad. Cuando ***_posicion*** sea 9, deberá transmitirse todavía el bit de parada (S), que será un 1. Tras transmitir el bit de parada, las variables ***_transmitiendo*** y ***_posicion*** deberán ser inicializadas a 0 para preparar la siguiente transmisión de un nuevo dato. Recuerde que lo primero que debe transmitirse es el bit de arranque, que es siempre 0. **(4 p.)**

```
_RutServRTC proc far
                sti
                push ax cx dx es

                ;Lectura del registro C del RTC para borrar el flag de interr. (PIF)
                mov al, 0Ch
                out 70h, al
                in al, 71h
```

```
                cmp _transmitiendo, 1
                jne salir

                xor ax, ax              ;Obtener dirección de reg. control LPT1
                mov es, ax
                mov dx, es:[0408h]
                inc dx
                inc dx

                mov ah, _dato
                mov cl, _posicion

                cmp cl, 0
                je env_A                ;bit de Arranque
                cmp cl, 9
                je env_P                ;bit de Parada

                dec cl
                ror ah, cl
                test ah, 01h
                jz env_0
                jmp env_1

env_A:
                in al, dx
                or al, 01h
                out dx, al
                jmp actualizar

env_P:
                in al, dx
                and al, 0FEh
                out dx, al
                mov _posicion, 0
                mov _transmitiendo, 0
                jmp salir

env_0:          in al, dx
                or al, 01h
                out dx, al
                inc cl
                jmp actualizar
env_1:
                in al, dx
                and al, 0FEh
                out dx, al
                inc cl

actualizar:
                inc cl
                mov _posicion, cl
salir:
                ;Envío de los EOI's a los PICs
                mov al, 20h
                out 0A0h, al
                out 20h, al

                pop es  dx cx ax
                iret
_RutServRTC endp
```