

**SISTEMAS BASADOS EN MICROPROCESADOR  
2º Grado Ingeniería Informática (EPS – UAM)**

**EXAMEN FINAL EXTRAORDINARIO – JULIO 2013**

**ENUNCIADO DEL PROBLEMA**

**IMPLEMENTACIÓN DE UN DRIVER DOS INTERFAZ CON UN SISTEMA DE  
RADIO BALIZA**

Un fabricante de sistemas de comunicación radio unidireccionales de larga distancia para aplicaciones especiales (tipo baliza de mensajes de texto) nos ha pedido el desarrollo de un driver DOS para PC que permita a los desarrolladores utilizar sus módulos de comunicación de la forma



más sencilla posible y transparente. Sus sistemas de radio baliza transmiten por defecto información GPS pero este fabricante quiere añadir la posibilidad de transmitir mensajes de texto (parecidos a un SMS) que permitan a los navegantes enviar información complementaria utilizando su radio baliza como teléfono de texto. En definitiva, quiere aumentar las prestaciones de sus equipos de emergencia y que puedan ser utilizados como teléfonos en situaciones de no emergencia, donde la información GPS no es transmitida ya que no hay peligro. Para ello se conectará la radio baliza a un ordenador personal con puerto LPT. Los módulos de comunicación radio tienen un puerto de 8 bits de datos de entrada y dos señales de protocolo

hardware, un pin de entrada activo a nivel bajo para conocer cuando hay un nuevo dato disponible en su puerto de datos y un pin de salida para confirmar la transmisión del dato y la posibilidad de un nuevo envío, también activo a nivel bajo. Los datos que se le tienen que enviar son los caracteres ASCII a transmitir. Ambos equipos (PC y módulo de comunicaciones) se conectarán mediante un cable de 11 hilos (8 de datos, 2 señales de protocolo y GND) terminado en sendos conectores DB-25. Uno de los conectores será compatible con el puerto LPT del PC y el otro estará adaptado al puerto de recepción de datos del módulo de comunicaciones.

El fabricante quiere ofrecer a sus clientes un software asociado a su módulo de comunicaciones formado por:

1. Una **librería de funciones (lib)** que deberá ser *linkada* con el programa en C que desarrollen aquellos que quieran utilizar el módulo.
2. Un **driver** que deberá ser instalado (ejecutado) de forma previa a la ejecución del programa de aplicación para que todo funcione correctamente.

El hardware, PC y módulo de comunicaciones, deberá estar conectado previamente a la ejecución del software y encendidos para evitar la pérdida de datos. La **librería de funciones** incluirá las siguientes funciones:

```
//Detecta la presencia o no del driver en memoria RAM, devolviendo un "1"  
//si el driver no está instalado y un "0" en caso contrario  
int DetectarDriver (void);
```

```
//Desinstala el driver instalado. Conviene llamar a esta función antes de  
//salir del programa de aplicación  
void DesinstalarDriver (void);
```

```
//Función que envía un dato (carácter ASCII) desde el PC al módulo de  
//comunicaciones. El dato (carácter) es el parámetro de entrada de la
```

```
//función. La función devuelve un código de error cada vez que se
//ejecuta, "0" para indicar transmisión OK y "1" para indicar ERROR
int TransmitirDato (char);
```

El **driver** ha sido implementado en ensamblador del 8086 para reducir al máximo su tamaño y conseguir el menor tiempo de ejecución posible. A continuación se incluye su código.

### **Driver de comunicación con la el sistema de radio baliza**

```
code segment
    assume cs:code

    ;Reservamos 100h bytes para el PSP
    org 100h

driver_start:
    jmp instalar

;Variables del driver

    old_70h        dw 0,0
    old_60h        dw 0,0
    old_A          db 0
    old_B          db 0
    flag_error     db 0
    flag_tx        db 0
    dato           db ? ;Carácter a transmitir
    contador       dw 0
    refresco       dw 2000

;Rutinas de Servicio

;Interrupciones Hardware

;Rutina de servicio del RTC
rutina_rtc proc far
    pushf
    push es
    push bx
    push ax
    push di
    push cx

    sti

    ;Leer el registro C del RTC
    mov al,0Ch
    out 70h,al
    in al,71h

    cmp flag_tx,1
    jne rutina_rtc_fin

    ;Decrementar el contador
    dec contador
    cmp contador,0
```

```

    jne rutina_rtc_fin

    ;Poner el flag de error a 1
    mov flag_error, 1

rutina_rtc_fin:

    ;Enviar el EOI al PIC esclavo
    mov al,20h
    out 0A0h,al
    ;Enviar el EOI al PIC maestro
    out 20h,al

    pop cx
    pop di
    pop ax
    pop bx
    pop es
    popf
    iret
rutina_rtc endp

;Interrupciones Software

;Interrupción software 60h
rutinas_driver proc near
    pushf
    push ds
    push di

    sti

    ;Desinstalar el driver
    cmp ah,01h
    jne driver_tx

    ;Desinstalar el driver
    call desinstalar

    jmp driver_fin

driver_tx:
    cmp ah,02h
    jne driver_presencia

    ;Inicializar variables relacionadas con la transmision
    mov bx,refresco
    mov contador,bx
    mov flag_tx,1
    mov flag_error,0

    ;Intentar transmitir el dato almacenado en AL
    call transmitir

    ;Devolver resultado de la transmisión en AL
    mov al,flag_error

```

```

        jmp driver_fin

driver_presencia:
    cmp ah,00h
    jne driver_fin

    ;Codigo de presencia a devolver
    mov ax,F0F0h

driver_fin:
    pop di
    pop ds
    popf
    iret
rutinas_driver endp

;Rutinas auxiliares del driver

;Rutina para transmitir el dato recibido utilizando el puerto LPT1 para
;enviarlo al módulo de comunicaciones

transmitir proc near
    push es
    push bx
    push dx
    push cx

    ;Envío del dato desde el PC al módulo de comunicaciones

    mov cx,40h
    mov es,cx
    mov dx,es:[8h]
    mov bl,dato
    out dx,bl

    ;Protocolo HW

    ;Activar la señal de protocolo que indica que un nuevo dato
    ;se encuentra en el puerto de datos para ser leído

    mov cx,40h
    mov es,cx
    mov dx,es:[8h]
    inc dx
    inc dx
    in bl,dx
    and bl,fbh
    out dx,bl

    ;;Esperar a que el módulo de comunicaciones confirme la
    ;;recepción del dato y su transmisión

    mov cx,40h
    mov es,cx
    mov dx,es:[8h]
    inc dx

```

```

_esperar_conf:
    cmp flag_error,0
    jne _dato_nook
    in bl,dx
    test bl,01000000b
    jnz _esperar_conf
    je _dato_ok
_dato_nook:
    mov al,01h ;Transmisión errónea
    jmp _fin_tx
_dato_ok:
    mov al,00h ;Transmisión realizada con éxito
    mov flag_tx,0
_fin_tx:
    pop cx
    pop dx
    pop bx
    pop es
    ret
transmitir endp

```

#### **;Rutinas de instalación / desinstalación del driver**

*;Función que recupera los vectores de interrupción y desactiva el RTC*  
desinstalar proc near

```

    push ax
    push es
    xor ax,ax
    mov es,ax

```

```

    cli

```

*;Recuperar los registros A y B del RTC*

```

    mov al,0bh
    out 70h,al
    mov al,old_B
    out 71h,al
    mov al,0ah
    out 70h,al
    mov al,old_A
    out 71h,al

```

*;Recuperar los antiguos vectores de interrupción*

```

;Vector 70h
    mov ax,old_70h
    mov es:[70h*4],ax
    mov ax,old_70h+2
    mov es:[70h*4+2],ax
;Vector 60h
    mov ax,old_60h
    mov es:[60h*4],ax
    mov ax,old_60h+2
    mov es:[60h*4+2],ax

```

```

    sti

```

```

    mov es,cs:[2ch]
    mov ah,49h
    int 21h

    mov ax,cs
    mov es,ax
    mov ah,49h
    int 21h

    pop es
    pop ax
    ret
desinstalar endp

instalar proc near
    xor ax,ax
    mov es,ax

    cli

    ;Guardar vectores de interrupción iniciales
    mov ax,es:[70h*4]
    mov old_70h,ax
    mov ax,es:[70h*4+2]
    mov old_70h+2,ax

    mov ax,es:[60h*4]
    mov old_60h,ax
    mov ax,es:[60h*4+2]
    mov old_60h+2,ax

    ;Guardar los registros A y B del RTC
    mov al,0ah
    out 70h,al
    in al,71h
    mov old_A,al

    mov al,0bh
    out 70h,al
    in al,71h
    mov old_B,al

    ;Instalar los nuevos vectores de interrupción
    mov es:[70h*4],offset rutina_rtc
    mov es:[70h*4+2],cs

    mov es:[60h*4],offset rutinas_driver
    mov es:[60h*4+2],cs

    ;Programar PIC esclavo habilitando interrupciones del RTC
    in al,0a1h
    and al,11111110b
    out 0a1h,al

    ;Programar frecuencia del RTC
    mov al,0ah

```

```
    out 70h,al
    mov al,26h
    out 71h,al

    ;Activar el PIE del RTC
    mov al,0bh
    out 70h,al
    in al,71h
    or al,01000000b
    out 71h,al

    sti

    mov dx,offset instalar
    int 27h
instalar endp

code ends
end driver_start
```