SISTEMAS BASADOS EN MICROPROCESADORES
(2º - GRADO ING. INFORMÁTICA)
EJERCICIO CLASE

El fabricante de una familia de microcontroladores de pequeñas prestaciones orientados al sector automovilístico nos ha pedido que diseñemos un sencillo sistema de programación para los mismos. Este programador estará formado por:

1. Un circuito hardware que se encargará de generar la tensión de grabación (V_{pp}) en uno de los terminales del microcontrolador (según especificaciones del fabricante), así como la lógica necesaria (adaptador de tensiones de línea) para implementar un puerto serie asíncrono compatible RS-232C. Los microcontroladores disponen de una sencilla UART con los terminales Tx y Rx (transmisión y recepción de datos) pero sin señales de protocolo (RTS, CTS, DTR, DSR, etc.) y en modo programación (con el terminal de grabación a V_{pp}) sólo funciona a 9600 baudios, 8 bits de datos, sin paridad y 1 bit de parada. Este circuito dispone de un zócalo (de presión nula) donde se inserta el microcontrolador para su grabación.

2. Un software para PC que consta de un programa principal desarrollado en C con algunas rutinas escritas en ensamblador del 8086 y un *driver* escrito en ensamblador para comunicarnos con la tarjeta (hardware) de programación.

3. Un cable de conexión serie con conectores DB-9 en sus extremos (para distinguirlos sin problemas, el del PC será hembra y el de la tarjeta macho). Este cable se conectará a uno de los puertos serie del PC y al puerto serie de la tarjeta de programación. El conector DB-9 del PC tendrá las conexiones necesarias para implementar un MODEM nulo y así “engañar” al PC, que espera que haya señales de protocolo. Sin embargo, el conector DB-9 de la tarjeta de programación sólo tendrá activos los terminales Tx, Rx y GND ya que la UART de los microcontroladores sólo dispone de estos terminales.

Los microcontroladores de esta familia disponen de un programa cargador (bootloader) grabado en su memoria ROM interna, capaz de comunicarse a través de su puerto serie asíncrono, compatible RS-232C, con el software de grabación que funciona en el PC. Para grabar el código de un programa en la memoria EEPROM del microcontrolador hay que enviarle byte a byte los caracteres ASCII del fichero generado por el compilador y que contiene el código máquina del programa de aplicación en un formato hexadecimal. (AFBB2F347ADE98F8H ....). Al enviar cada carácter estamos enviando el código ASCII del nibble de cada byte de código, formado por 2 nibbles consecutivos. El programa cargador del microcontrolador se encarga de adaptar esta información para su grabación en binario en las distintas direcciones de la memoria de programa.

Programa Principal

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

/* Prototipos de las funciones escritas en ensamblador */
extern int far DetectarDriver ();
extern void far DesinstalarDriver ();
extern int far LeerArgumentos (char *, char *);
extern void far InicializarPuertoSerie (int);
extern int far GrabarCodigo (char);

void main(void)
 {
  char PuertoSeriePar[10];
  char NomFichCod[100];
  char cod;
  int PuertoSerie;
  FILE *FPCod;

  /* Lectura de argumentos */
  if (LeerArgumentos(PuertoSeriePar,NomFichCod)==1)
   {
    printf("Parámetros de entrada incorrectos.\n");
    printf("Recuerde: mc-prog –n fichero\n");
    printf("n : indica el puerto serie del PC (1 o 2)\n")
    printf("fichero : fichero que contiene el código a grabar\n");
    exit(0);
   }

  PuertoSeriePar[0] = PuertoSeriePar[1];
  PuertoSeriePar[1] = "\0";
  PuertoSerie = atoi (PuertoSeriePar);

  if (DetectarDriver()==1)
   {
    printf("Driver no instalado.\n");
    exit(0);
   }

  InicializarPuertoSerie(PuertoSerie); /* Inicializa puerto serie PC */

  /* Abrir fichero de código a programar */
  if ((FPCod=fopen(NomFichCod,"rt"))==NULL)
   {
    printf("Error al abrir el fichero %s\n",NomFichCod);
    DesinstalarDriver();
    exit(0);
   }

  cod = fgetc(FPCod);  /* Lectura del nibble de código del fichero */

  while (!feof(FPCod))
   {
    /* Saltar RC (13), LF (10) y caracteres vacíos */
```

```
    if ((cod != '\n') && (cod != '\r') && (cod != ' '))
     {
       if (GrabarCodigo(cod)==1) /* Grabar cada nibble de código */
        {
          printf("Error en el envío de código al programador.\n");
          break;
        }
       }
    cod = fgetc (FPCod); /* Lectura del nibble de código del fichero */
   }

  if (FPCod != NULL) fclose(FPCod); /* Cerrar el fichero de código */
  DesinstalarDriver();

  printf("Fin del Proceso de Grabación.\n");
  exit(0);

 } /* Fin del Programa Principal */
```

Rutinas en Ensamblador

```
_text segment byte public 'code'
      assume cs:_text

_DetectarDriver proc far
      push es
      xor ax,ax
      mov es,ax
      cmp word ptr es:[60h*4],0
      jne detectar_int
      cmp word ptr es:[60h*4+2],0
      je detectar_nodriver
detectar_int:
      mov ah,00h
      int 60h
      cmp ax,0F0F0h
      jne detectar_nodriver
      xor ax,ax
      jmp detectar_fin
detectar_nodriver:
      mov ax,1
detectar_fin:
      pop es
      ret
_DetectarDriver endp

_DesinstalarDriver proc far
      mov ah,03h
      int 60h
      ret
_DesinstalarDriver endp

_LeerArgumentos proc far
.................
;Función que devuelve los dos parámetros de entrada
;Si todo va bien, la función retorna en AX un 0
```

```
;Si hay error, la función retorna en AX un 1
.................
_LeerArgumentos endp

_InicializarPuertoSerie proc far
      push bp
      mov bp,sp
      mov ax,bp[6] ; En AL se queda el número del puerto
      mov ah,01h
      int 60h
      pop bp
      ret
_InicializarPuertoSerie endp

_GrabarCodigo proc far
      push bp
      mov bp,sp
      mov ax,[bp+6]
      mov ah,02h
      int 60h
      xor ah,ah
      pop bp
      ret
_GrabarCodigo endp

public _DetectarDriver
public _DesinstalarDriver
public _LeerArgumentos
public _InicializarPuertoSerie
public _GrabarCodigo

_text ends
end
```

Driver de comunicación con la tarjeta de programación

```
code segment
      assume cs:code

      ;Reservamos 256 bytes para el PSP
      org 256

driver_start:
      jmp instalar

;Variables del driver

      flag_error    db 0
      flag_grabando db 0
      dato          db ?      ; Nibble a grabar
      frec_RTC      db 26h    ; Frecuencia de interrupciones del RTC
                              ; 1000 interrupciones por segundo
      dirbasep      dw ?      ; Dir. base del puerto RS-232C
      contador      dw 0
      refresco      dw 2000   ;Interrupciones en 2 segundos
```

```
;Rutinas de Servicio

;Interrupciones Hardware

;Rutina de servicio del RTC
rutina_rtc proc far
      sti
      push ax
      ;Leer el registro C del RTC
      mov al,0Ch
      out 70h,al
      in al,71h
      cmp flag_grabando,1
      jne rutina_rtc_fin
      ;Decrementar el contador
      dec contador
      jne rutina_rtc_fin
      ;Poner el flag de error a 1 (han transcurrido 2 segundos)
      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 ax
      iret
rutina_rtc endp


;Interrupciones Software

;Interrupción software 60h
rutinas_driver proc near
      sti
      push bx
      cmp ah,03h
      jne driver_grabar
      ;Desinstalar el driver
      call desinstalar
      jmp driver_fin

driver_grabar:
      cmp ah,02h
      jne driver_iniciarpuerto
      ;Inicializar variables relacionadas con la grabación
      mov bx,refresco
      mov contador,bx
      mov flag_grabando,1
      mov flag_error,0
      ;Intentar grabar. Nibble a grabar en AL
      call grabar
      ;Devolver resultado de la grabación en AL
      mov al,flag_error
      jmp driver_fin

driver_iniciarpuerto:
```

```
      cmp ah,01h
      jne driver_presencia
      ;Configurar RS-232C. El puerto (1 o 2) en AL
      call configurar
      jmp driver_fin

driver_presencia:
      cmp ah,00h
      jne driver_fin
      ;Codigo de presencia a devolver
      mov ax,0F0F0h
driver_fin:
      pop bx
      iret
rutinas_driver endp

;Rutinas auxiliares del driver

;Rutina para inicializar el puerto serie (9600 baudios, 8 bits, No
Paridad, 1 bit Parada). Puerto (1,2) en AL
configurar proc near

;Pregunta P4

.................

configurar endp

;Rutina para grabar el dato (nibble) recibido
;utilizando puerto serie para enviarlo al programador
grabar proc near
      push es
      push dx

      mov dato,al
      mov dx,dirbasep
      add dx,5
THR_lleno:
      in al,dx
      test al,00100000b
      jz THR_lleno
      mov dx,dirbasep
      mov al,dato
      out dx,al
      add dx,5    ; Apunta a LSR
esperar_eco:
      cmp flag_error,0
      jne dato_nook
      in al,dx
      test al,00000001b
      jz esperar_eco
      mov dx,dirbasep
      in al,dx
      cmp dato,al
      je dato_ok
dato_nook:
      mov al,01h
```

```
      jmp fin_grabar
dato_ok:
      mov al,00h
      mov flag_grabando,0
fin_grabar:
      pop dx
      pop es
      ret
grabar endp

;Rutinas de instalación / desinstalación del driver

desinstalar proc near
      ;Salvar contexto en la pila
      push ax
      push es

      ;Desintalación del driver utilizando la función 49h de la int 21h
      mov es,cs:[2Ch]
      mov ah,49h
      int 21h

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

      ;Recuperar el contexto de la pila
      pop es
      pop ax
      ret
desinstalar endp

instalar proc near
      ;Inicializar valor de ES
      mov ax,0
      mov es,ax
      ;Instalar los nuevos vectores de interrupción
      ;Vector 70h asociado al RTC
      cli
      mov es:[70h*4], offset rutina_rtc
      mov es:[70h*4+2], seg rutina_rtc
      ;Vector 60h asociado a la interrupción software
      mov es:[60h*4], offset rutinas_driver
      mov es:[60h*4+2], seg rutinas_driver
      sti
      ;Programar PIC maestro habilitando interrupciones PIC esclavo
      ........................
      ;Programar PIC esclavo habilitando interrupciones del RTC
      ........................
      ;Programar frecuencia del RTC
      mov al,0Ah
      out 70h,al
      mov al, frec_RTC ;Pregunta P3
      out 71h,al
      ;Activar el PIE del RTC
      ........................
```

```
      sti
      ;Instalar el driver utilizando la int 27h
      mov dx,offset instalar
      int 27h
instalar endp

code ends
end driver_start
```

EXAMEN DE PROBLEMAS ETC-II (FEB03)
PARTE I

Nombre : _____

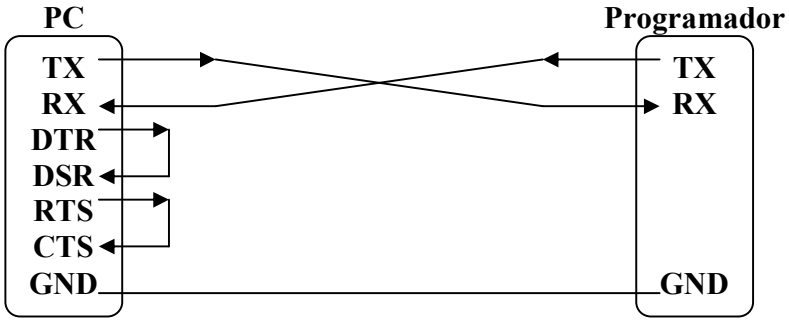
Apellidos : _____

DNI : _____ **Grupo :** _____

P1. Escriba el código en ensamblador de la rutina GrabarCodigo (char) del programa principal teniendo en cuenta que devuelve un código de error que será 1 si se ha producido algún error durante el proceso de grabación y 0 en caso contrario. El parámetro de entrada es el código (nibble) a grabar en una determinada posición de la memoria del microcontrolador. Esta función deberá comunicarse con el *driver* mediante la interrupción software 60h. EL código (nibble) a grabar deberá pasarse a través del registro AL al *driver*. Para indicar al *driver* que queremos grabar dicho nibble de código deberemos escribir en AH un 2. El resultado de la ejecución de dicha petición en el *driver* será indicado por éste a través de AL, que valdrá 0 si se ha efectuado la grabación con éxito o un 1 si ha habido error. (2 puntos)

```
_GrabarCodigo proc far
  push bp
  mov bp,sp
  mov ax,[bp+6]
  mov ah,02h
  int 60h
  xor ah,ah
  pop bp
  ret
_GrabarCodigo endp
```

P2. Dibuje las conexiones en los terminales de los conectores DB-9 del cable de interconexión entre el PC y la tarjeta del programador. Tenga en cuenta sólo los terminales Tx,Rx,GND,RTS,CTS,DTR,DSR de cada conector. (1 punto)



P3. Implemente un mecanismo de control en el *driver* que permita la detección de problemas hardware en las comunicaciones (rotura del cable, no conexión de la tarjeta del programador, etc.). Utilice el RTC del PC para diseñar un temporizador que nos permita determinar que ha transcurrido un tiempo máximo (*time out*) sin haber recibido respuesta de la tarjeta del programador y que por tanto, indica que hay problemas en la grabación. La temporización se realizará en base a las interrupciones periódicas que puede generar el RTC. El tiempo máximo de espera será de 2 segundos. Incorpore las variables necesarias para la implementación de esta rutina. Considere que la parte de instalación y desinstalación del vector de interrupción asociado al RTC están ya implementadas en el driver. Asigne valores a las variables "frec_RTC" y "refresco" definidas en el *driver* para conseguir las condiciones establecidas para el time out. Utilice la variable contador ya definida. (3 puntos)

```
rutina_rtc proc far
      sti
      push ax
      ;Leer el registro C del RTC
      mov al,0Ch
      out 70h,al
      in al,71h
      cmp flag_grabando,1
      jne rutina_rtc_fin
      ;Decrementar el contador
      dec contador
      jnz rutina_rtc_fin
      ;Poner el flag de error a 1 (han transcurrido 2 segundos)
      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 ax
      iret
rutina_rtc  endp
```

Una posible solución para los valores de las variables es:

La variable **frec_RTC** tendrá el valor **26h** para que se generen 1024 interrupciones por segundo. Al llegar a 2000 interrupciones (valor de la variable **refresco**) habrán transcurrido de forma muy aproximada 2 segundos, que es el tiempo máximo de espera antes de considerar que hay problemas en la grabación.

EXAMEN DE PROBLEMAS ETC-II (FEB03)
PARTE II

Nombre : _____

Apellidos : _____

DNI : _____ **Grupo :** _____

P4. Escriba el procedimiento *configurar* de inicialización del puerto serie del PC en el *driver* que será llamado desde el programa principal con la rutina InicializarPuertoSerie (int) antes descrita. Tenga en cuenta la información sobre las comunicaciones serie dadas en el enunciado de este ejercicio. (2 puntos)

```
configurar proc near
      push es
      push ax
      push dx

      mov dx,40h
      mov es,dx
      cmp al,1
      jne COM2
      mov ax,es:[0]
      jmp inicia_RS232

COM2:
      mov ax,es:[2]

inicia_RS232:
      mov dirbasep,ax
      mov al,10000011b ;DLAB=1,8 bits,1 bit parada, sin paridad
      mov dx,dirbasep
      add dx,3 ;Acceder a LCR
      out dx,al
      mov al,0Ch ;Byte menos significativo de velocidad
      mov dx,dirbasep
      out dx,al
      inc dx
      mov al,00h ;Byte más significativo de velocidad
      out dx,al ;Velocidad a 9600 baudios
      mov al,00000011b ;DLAB=0,8 bits,1 parada,sin paridad
      mov dx,dirbasep
      add dx,3 ;Acceder a LCR
      out dx,al

      pop dx
      pop ax
      pop es
      ret
configurar endp
```

P5. Analize el código y dibuje el organigrama de la rutina auxiliar *grabar* del *driver.* La variable flag_error es actualizada (se pone a 1) cuando transcurren 2 segundos desde que empezó el proceso de grabación de un dato. Su actualización la lleva a cabo un mecanismo de control software que se encarga de realizar ese medición de tiempo utilizando el RTC y que es objeto de la pregunta 3 de este exámen. Indique claramente el mecanismo de verificación de los datos grabados que implementa esta rutina. (2 puntos)