SISTEMAS BASADOS EN MICROPROCESADORES (SBM)

Grado en Ingeniería Informática
Escuela Politécnica Superior – UAM

PARCIAL 2
CURSO 2016-17

ENUNCIADO

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 paralelo. Los microcontroladores dispone de varios puertos de 8 bits, uno de los cuales, configurado como entrada, será utilizado para recibir los datos a grabar desde el puerto paralelo del PC. Así mismo, otro pin de salida de otro puerto del microcontrolador será utilizado para devolver una señal de confirmación de "dato recibido y grabado sin problemas", que estará conectado a un pin de entrada del puerto paralelo del PC que actúa como señal de protocolo. El valor que devuelve la tarjeta de grabación es un "1" lógico para indicar que no hay problemas. 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 paralelo con conectores DB-25 en sus extremos (para distinguirlos sin problemas, el del PC será macho y el de la tarjeta hembra). Este cable se conectará al puerto paralelo del PC y al puerto paralelo (emulado) de la tarjeta de programación. Los pines utilizados serán los D0-D7 y el pin de entada adicional que actuará como señal de protocolo del conector DB-25.

Los microcontroladores de esta familia disponen de un programa cargador (bootloader) grabado en su memoria ROM interna, capaz de comunicarse a través de uno de sus puertos (paralelo) 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 ,carácter a carácter, 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 en C del Programa de Aplicación (mc-prog.exe)

```
#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 *);
extern int far GrabarCodigo (char);

void main(void)
 {
  char NomFichCod[100];
  char cod;
  FILE *FPCod;

  /* Lectura de argumentos en la llamada al programa*/
  if (LeerArgumentos(NomFichCod)==1)
   {
    printf("No hay argumentos de entrada.\n");
    Printf("Recuerde: mc-prog fichero\n");
    printf("fichero : fichero que contiene el código a grabar\n");
    exit(0);
   }

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

  /* 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 del Programa de Aplicación (mc-prog.exe)

```
_codigo_rutinas segment byte public
      assume cs:_codigo_rutinas

_DetectarDriver proc far
      .........
_DetectarDriver endp

_DesinstalarDriver proc far
      ............
_DesinstalarDriver endp

_LeerArgumentos proc far
      push bp si bx cx dx ds

      mov  bp,sp
      mov  ds,[bp+18]   ;Segmento de la dirección del array de caracteres
      mov  bx,[bp+16]   ;Offset de la dirección del array de caracteres

; Codigo de lectura de argumentos
      xor dh,dh
      ;ES apunta al PSP si no se ha cambiado
      mov dl,es:[80h] ;Número de bytes en los parámetros
      cmp dx,0 ;Si hay más de 1 seguimos
      jne _hay_par

      mov ax,01h ;Error al no haber parámetros de entrada
      jmp _salir_leer
_hay_par:
      ..................... (P4)
      xor ax,ax
_salir_leer:
      pop ds dx cx bx si bp
      ret
_LeerArgumentos endp

_GrabarCodigo proc far
      ............
_GrabarCodigo endp

public _DetectarDriver
public _DesinstalarDriver
public _LeerArgumentos
public _GrabarCodigo

_codigo_rutinas ends
end
```

Driver de comunicación con la tarjeta de programación

```
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
      flag_error    db 0
      flag_grabando db 0
      dato          db ? ;Nibble a grabar
      dirbasep      dw 03BCh ;Dir. base del puerto LPT1
      contador      dw 0
      refresco      dw 2000

;Rutinas de Servicio
;Interrupciones Hardware

;Rutina de servicio del RTC
rutina_rtc proc far
      .........
rutina_rtc  endp


;Interrupciones Software
;Interrupción software 60h
rutina_driver proc near
      push bx

      ;Desinstalar el driver
      cmp ah,02h
      jne driver_grabar

      ;Desinstalar el driver
      call desinstalar

      jmp driver_fin

driver_grabar:
      cmp ah,01h
      jne driver_presencia

      ;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_presencia:
      cmp ah,00h
      jne driver_fin

      ;Codigo de presencia a devolver
      mov ax, ABCDh

driver_fin:
      pop bx
      iret
rutina_driver endp

;Rutinas auxiliares del driver
;Rutina para grabar el dato (nibble) recibido
;utilizando puerto serie para enviarlo al programador
grabar proc near
      push bx dx

      xor bx,bx
      mov dato,al
      mov dx,dirbasep ;Acceder al reg. de datos del LPT1
      mov bl,dato
      out dx,bl ;Enviar dato (nibble)por el LPT1
      add dx,1

;Protocolo
_esperar_señal:
      cmp flag_error,0
      jne _dato_nook
      in bl,dx
      test bl,01000000b
      jz _esperar_señal
      jmp _dato_ok

_dato_nook:
      mov al,01h ;Grabación errónea (time-out)
      jmp _fin_grabar

_dato_ok:
      mov al,00h ;Grabación realizada con éxito
      mov flag_grabando,0

_fin_grabar:
      pop dx bx
      ret
grabar endp

;Rutinas de instalación / desinstalación del driver
;Funcion que recupera los vectores de interrupción y desactiva el RTC
desinstalar proc near
      ..........
desinstalar endp
```

```
instalar proc near
      xor ax,ax
      mov es,ax

      cli

      ;Guardar vectores de interrupción inciales
      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

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

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

      sti

      ;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

      mov dx,offset instalar
      int 27h
instalar endp

code ends
end driver_start
```

SISTEMAS BASADOS EN MICROPROCESADORES (SBM)
Grado en Ingeniería Informática - Escuela Politécnica Superior – UAM

PARCIAL 2 - CURSO 2016-17

PREGUNTAS

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

P1. Analizar el código del enunciado e implementar la rutina de servicio de las interrupciones del RTC que forma parte del driver. La función de la misma es generar un mecanismo de "time-out" que se dispara cuando va a comenzar la grabación de un dato (nibble) (variable global *flag_grabando* a 1) y que contabiliza si han transcurrido N segundos antes de que dicha variable se haya puesto a 0 (fin de la grabación del dato). Con cada interrupción se decrementa la variable global *contador* y cuando llega a 0, la rutina pone otra variable global *flag_error* a 1 (indica que se ha producido time-out) y en caso contrario, *flag_error* será 0. (3 p.)

```
rutina_rtc proc far
      push ax

      sti

      ;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
      cmp contador,0
      jne rutina_rtc_fin

      ;Poner el flag de error a 1 (han transcurrido 2000 int o 2 s)
      mov flag_error, 1
      mov flag_grabando, 0

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
```

P2. Analizando el código del enunciado, implemente la rutina _GrabarCodigo en ensamblador, sabiendo que es una rutina interfaz entre el programa de aplicación (.exe) y el driver (.com) y que es llamada desde el programa principal (main()) escrito en C. (2 p.)

```
_GrabarCodigo proc far
      public _GrabarCodigo

      push bp
      push cx
      mov bp,sp
      mov cx,bp[8]
      mov ax,cx ; En AL queda el código (nibble) a grabar
      mov ah,01h
      int 60h
      pop cx
      pop bp
      ret
_GrabarCodigo endp
```

P3. Analizando el código del driver indique cuál es el tiempo de "time-out" establecido para decidir si la grabación de un dato ha sido correcta o errónea. Justifique la respuesta. (1.5 p.)

El RTC es programado en la parte de instalación del driver para generar 1000 interrupciones por segundo, escribiendo el valor h en los bits RS del registro A.

El valor de la variable global contador es inicializado con el valor de la variable global refresco (2000) cada vez que se dispara el mecanismo de time-out antes de proceder a la grabación de un dato por el driver. La rutina de servicio del RTC cuenta estas 2000 interrupciones antes de indicar que se ha producido un error de grabación. Por tanto, habrán transcurrido 2 s.

P4. Analizando el código de la aplicación (mc-prog.exe) complete el código de la rutina _LeerArgumentos, escrita en ensamblador, que es llamada desde el programa principal en C, justo en la zona marcada por la línea de puntos donde se indica (P4).Suponed que ES apunta al PSP. No es necesario que escriba todo el código de nuevo, sólo las líneas que faltan para completar la rutina (2 p.)

```
                        ;DS contiene el segmento de dirección del array

                        mov   si, 0h
copiar
                        mov   cl, es:[si+81h]

                        ;bx contiene el offset de la dirección del array
                        mov   [bx][si], cl

                        inc   si
                        cmp   si, dx

                        jnz   copiar
```

P5. Analizando el código de la rutina *grabar* del driver indique qué bit (nombre del mismo en el registro al que pertenezca) del LPT1 se está utilizando como señal de protocolo (entrada). (1.5 p)

Analizando el código de la rutina grabar comprobamos que tras enviar el dato (escribiéndolo en el registro de datos del LPT1), la dirección base (dirbasep) se incrementa en 1, con lo que la lectura que se lleva a cabo después del envío (in bl,dx) es del registro de estado. Tras leer su contenido, hacemos un comprobación del valor del bit 6 (test bl,01000000b) y si el resultado es 0, seguimos esperando y comprobando ese bit. Cuando es 1 (el grabador indica que la grabación se ha producido con éxito), si no se ha producido el time-out (flag_error = 1), damos por bueno el envío (grabación) del dato (_dato_ok), devolviendo en AL un 0. Por tanto, el bit que se utiliza como señal de protocolo de entrada es el #ACK del registro de Estado.