UNIVERSIDAD DE HUELVA
ÁREA DE SISTEMAS
7-SEPT-2004

ESCUELA POLITÉCNICA SUPERIOR
INFORMÁTICA INDUSTRIAL II
TIEMPO: 2 Horas

CUESTIONES:

C1. Enumera y comenta brevemente el proceso de desarrollo de software que lleva a la realización de un programa para un sistema basado en microcontrolador empleando lenguaje C. (0.75 puntos).

Solución:

Para el desarrollo software empleando lenguaje C se realizan las siguientes fases:

- ***Partición del problema** en módulos y asignación de tareas a cada módulo.*
- *Escritura del código fuente de cada módulo con un **editor de texto** y de su correspondiente fichero cabecera '*.h' donde aparecen listadas las funciones que ese modulo exporta a los demás.*
- *Escritura del fichero principal donde se encuentra la función 'main' donde se incluyen el fichero de cabecera de cada módulo.*
- ***Compilación** de cada fichero por separado.*
- ***Linkado** de todos los ficheros conjuntamente y con el fichero de inicio STARTUP.*
- ***Programación** de EPROM o descarga del programa a la placa de desarrollo.*
- ***Depuración** del programa sobre la placa de desarrollo o mediante un simulador software.*

C2. Explica que finalidad y que herramientas se emplean en la fase de depuración de un programa diseñado para un sistema basado en microcontrolador. (0.75 puntos).

*La principal finalidad de la fase de depuración es encontrar **errores semánticos en el programa** o sea fallos en el funcionamiento del programa. Estos errores sobre funcionamiento no pueden ser encontrados por el analizador sintáctico del compilador en la fase de compilación.*

Las herramientas que se pueden emplear son las siguientes:

- ***Programador de EPROM**: permite la carga del programa en el sistema target final.*
- ***Depuradores remotos o monitores**: permite un testeo limitado del programa funcionando en el target final.*
- ***Emuladores en circuito**: permiten un testeo casi ilimitado del funcionamiento del microcontrolador.*
- ***Simuladores**: efectúan una simulación por software del programa funcionando en un sistema ideal.*
- ***Analizador lógico**: permite testear señales digitales concretas, como peticiones de interrupción y señales externas al microcontrolador.*

C3. ¿Cada cuanto tiempo se incrementa el temporizador en el 8051? Justifica la respuesta. ¿Cómo se calcula el valor de recarga en el modo 1(contador de 16 bits) para temporizar 10 mseg (0.75 puntos).

***Cada microsegundo** funcionando el microcontrolador a 12 MHz. El valor de recarga del contador se calcula **restando de 65535 el tiempo que queremos temporizar** en microsegundos, puesto que el contador contará hacia arriba desde este valor hasta alcanzar el máximo valor de cuenta y entonces señalará este evento dando curso a la interrupción del temporizador. El valor de recarga para 10 milisegundos sería (65535-10000), o sea: 55535 o bien 0xD8EF.*

C4. Indica en que casos resulta necesario emplear los registros de ajuste de prioridad de interrupciones (IP) en el 8051. (0.75 puntos).

Cuando deseamos que una determinada interrupción de las seis posibles tenga una prioridad mayor. Por ejemplo, puede que tengamos una interrupción periódica del timer que necesite ser cursada para mantener un contador de tiempo con precisión antes que cualquier otra.

Se dispone de dos niveles de prioridad. Existe un bit para cada interrupción que permite marcar cada una de ellas para que tenga una prioridad normal (bit a '0') o una prioridad mayor (bit a '1'). Además en el caso de que varias de ellas tengan ese bit con el mismo valor se emplea la siguiente tabla de precedencia:

Prioridad	Fuente
(más alta) 1	/INT0
2	Timer 0
3	/INT1
4	Timer 1
(más baja) 5	Puerto Serie (RI, TI)

PROBLEMAS:

P1. (2 puntos) El siguiente código fuente C para 8051 a 12 MHz tiene varios errores, tanto de sintaxis como de semántica. Encontrarlos y corregir el código fuente.

Solución:

```
#include <reg51.h>
```

```
Error sintáctico: Falta definir ON y OFF
#define ON 1
#define OFF 0
```

```
/*****************************************
 Control del calefactor
*****************************************/
/* entradas sensores de temperatura*/
sbit SEL=P1^3;
sbit ST1=P1^0;
sbit ST2=P1^1;
sbit ST3=P1^2;
/* salidas */
```

```
sbit CALEFACTOR=P1.4;
Error sintáctico,debe ser: sbit CALEFACTOR=P1^4;
```

```
/* Función que devuelve temperatura aproximada en función del sensor
que se haya activado */
unsigned char Temperatura(void) {
      /* ST3 se activa cuando es mayor de 30 */
      if (ST3) return 31;
      /* ST2 se activa cuando es mayor de 25 */
      if (ST2) return 26;
      /* ST1 se activa cuando es mayor de 20 */
      if (ST1) return 21;
      return 19;
}
/* estructura para guardar max y min */
struct control {
      unsigned char min;
      unsigned char max;
      };
```

```
/* tabla de temperaturas a las que cambia el estado del calefactor */
```

```
struct control tabla_temp[1]={{20,25},{25,30}};
```
Error sintáctico, ***debe ser:*** *struct control tabla_temp[2]= {{20,25},{25,30}};*

```
void main(void) {
unsigned char temp;

/* activa el calefactor */
CALEFACTOR=ON;
/* en función de la entrada SEL selecciona el rango de temperaturas adecuado que se ha fijado en la tabla "tabla_temp" */
temp=Temperatura();
```

```
if (temp<tabla_temp[max].SEL) CALEFACTOR=OFF;
else if (temp>tabla_temp[SEL].min) CALEFACTOR=ON;
```
Error sintáctico y semántico, ***debe ser :*** *if (temp>tabla_temp[SEL].max) CALEFACTOR=OFF;*
else if (temp<tabla_temp[SEL].min) CALEFACTOR=ON;

```
}
```

P2. La empresa de aparatos de medida MEDIDATROL S.A. está desarrollando un nuevo modelo de polímetro controlado por PC. Para el desarrollo del equipo se ha contratado a la empresa VALOREFICAZ S.A. el desarrollo de un módulo de adquisición de señal para efectuar medidas AC de valor eficaz (rms) de alta precisión. El módulo devuelve en todo momento por un bus de 16 bits conectado al puerto P0 y al puerto P1 el valor de la medida en código binario, siendo el dato en P1 el byte de mayor peso. El rango de funcionamiento básico del módulo es de 0 a 60.000 uV AC rms, aunque éste es variable. Las librerías proporcionadas por el equipo desarrollo son las siguientes:

LCD.C / LCD.H	
`void lcd_init(void)`	Inicializa LCD
`void lcd_puts(unsigned char *)`	Escribe cadena texto
`void lcd_putch(unsigned char)`	Escribe carácter ASCII

ADC.C/ADC.H		
Función	Objetivo	Parámetro
`void adc_init(void)`	Inicializa Convertidor	Ninguno
`void adc_ajusta_rango(unsigned char)`	Cambia el rango de medida	Rango de medida

El módulo devuelve un valor mayor de 60.000 cuando el rango ha sido sobrepasado. En ese caso el programa de control debe cambiar de rango de forma automática. En cualquier caso el módulo dispone de entrada de alta impedancia y no se daña si se sobrepasa el rango de medida.

Rango	Código binario	Código Rango
0 a 60mV	0 a 60000	0
0 a 600mV	0 a 60000	1
0 a 6V	0 a 60000	2
0 a 60V	0 a 60000	3
0 a 600V	0 a 60000	4

El programa principal debe sacar en pantalla al comienzo durante 2 segundos un mensaje de copyright y comenzar inicializando los módulos y activando el rango de mayor sensibilidad. Cuando va siendo necesario por la medida debe ajustar el rango más conveniente. En todo momento debe sacar de forma escueta por la pantalla LCD el rango y el valor de la medida con 5 dígitos de precisión con su unidad de medida (mV,V) y punto decimal correspondiente

(dispone de pantalla de 1x16 caracteres). La pantalla no se debe refrescar a mayor velocidad de 50ms para evitar su parpadeo.

Se pide realizar el diagrama de flujo (1 punto) y la función main del programa (2 puntos).

Solo se pueden usar las funciones comentadas, aunque se pueden escribir las funciones auxiliares que se crea necesario y emplear la librería de temporización con fichero de cabecera “delay.h” (DelayMs, DelayS) para los retardos.

Solución:

No es necesario el empleo de interrupciones en el diseño del programa. El diagrama de flujo tendrá dos partes bien diferenciadas:

- ***Inicialización** de los módulos y salida por pantalla del mensaje inicial.*
- ***Bucle sin fin** donde se lee la entrada continuamente, se ajusta el rango y se actualiza el display. El rango se puede ajustar antes o después de actualizar el display, aunque lo más lógico es hacerlo antes de actualizar el display.*

*El formato de display más lógico sería aquel que diese las siguientes informaciones: **Rango**, **valor** de la medida con punto decimal y **unidad** de medida. Algo así sería adecuado: “R-A 60.000 uV” que son 14 caracteres, como el display tiene 16 introduciremos un espacio delante para centrar la presentación en pantalla.*

Podemos hacer una función para actualizar la pantalla. Tomaría como parámetros el valor de la medida, y el rango y mostraría el valor. Tendría el siguiente prototipo: `void mostrar_lectura(unsigned char rango, unsigned int lectura);`

Resulta evidente que necesitamos almacenar en el programa el valor tomado de la medida y el valor del rango actual del voltímetro. También podemos realizar una función para ajustar el rango, esta función toma como entrada el valor actual de medida y el rango actual, para devolver el rango aconsejado, que puede ser menor (lectura menor de 60) o mayor (lectura mayor de 60.000) que el actual. Tendría el siguiente prototipo: `unsigned char nuevo_rango(unsigned int lectura, unsigned char rango_actual);`

*De esta forma, la función **main** podría ser tan sencilla como esto:*

```
#include “adc.h”
#include “delay.h”
#include “lcd.h”

void main(void) {
unsigned char rango=0,n_rango; /* rango inicial cero */
unsigned int lectura; /* ojo!! son 16 bits */

     inicializa_instrumento();  /* inicializo modulos */
     while (1) {
          lectura=P1;    /*guardo byte alto de la Word */
          lectura<<=8;   /* lo coloco en la parte alta */
          lectura+=P0;   /* completo Word con la parte baja */
/* calculo nuevo rango a partir de la lectura realizada */
          n_rango=nuevo_rango(lectura,rango);
/* ajusto el nuevo rango */
```

```
          adc_ajusta_rango(n_rango);
/* muestro lectura anterior realizada con el rango antiguo*/
          mostrar_lectura(rango,lectura);
/* actualizo el rango actual */
          rango=n_rango;
     }
}
```

Las funciones que realizan casi todo el trabajo son las siguientes:

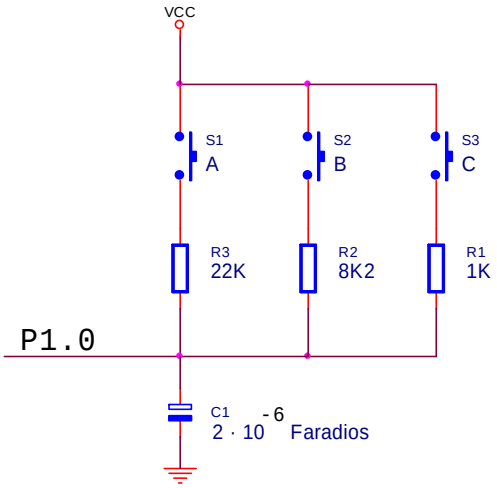
```
void inicializa_instrumento(void) {
     /* inicializa los modulos */
     adc_init();
     lcd_init();
     /* saca por pantalla saludo */
     lcd_puts(“ MEDIATROL S.A. “);
     DelayS(2); /* espera dos segundos */
     lcd_puts(“               “); /* borra */
}

unsigned int ajusta_rango(unsigned int lectura,unsigned char
rango_actual) {
     /* en los margenes extremos no cambia de rango */
     if ((lectura<60) && (rango>0)) return (rango_actual-1);
     if ((lectura>60000) && (rango<4)) return (rango_actual+1);
     else return rango_actual;
}

void  mostrar_lectura(unsigned  char  rango,  unsigned  int
lectura) {
     /* espera un tiempo para evitar parpadeo */
     DelayMs(50);
     /* tratamos primero el caso extremo */
     if ((rango==4) && (lectura>60000)) {
          lcd_putst(“ERR: SOBRERRANGO“); /* medida > 600V */
          return;
          }
     /* a continuación el caso normal */
     lcd_putc(‘ ‘);
     lcd_putc(‘R’);
     lcd_putc(‘-‘);
     lcd_putc(rango+’A’); /* saca A,B,C,D,E funcion de rango */
     lcd_putc(‘ ‘);
     lcd_putc(lectura/10000+’0’); /* decenas millar */
     if (rango==2) lcd_putc(‘.’); /* punto en rango 2 */
     lcd_putc((lectura/1000)%10+’0’); /* unidades millar */
     if ((rango==3) || (rango==0))
          lcd_putc(‘.’); /* punto en rangos 0 y 3 */
     lcd_putc((lectura/100)%100+’0’); /* centenas */
     if ((rango==1) || (rango==4))
          lcd_putc(‘.’); /* punto en rangos 1 y 4 */
     lcd_putc((lectura/10)%1000+’0’); /* decenas */
     lcd_putc(lectura%10000+’0’); /* unidades */
     lcd_putc(‘ ‘);
```

```
     if (rango>2) lcd_putc(‘ ‘); /* saca unidad de medida */
     else lcd_putc(‘m’);
     lcd_putc(‘V’);
     lcd_putc(‘ ‘); /* total hasta ahora 16 caracteres */
}
```

P3. (2 puntos) Se pretende realizar una librería para la lectura de un teclado de tres teclas conectado al bit P1.0 del 8051. Para que esto sea posible el programa descarga el condensador poniendo a cero la salida y a continuación pone un uno en la salida y espera a que el condensador se cargue testeando el bit hasta que vale uno. Si conocemos el tiempo que ha tardado en cargarse podemos averiguar el pulsador que se ha pulsado. ESTE TIEMPO ES APROXIMADAMENTE IGUAL A LA CONSTANTE DE TIEMPO DEL CIRCUITO, O SEA R·C. Suponer que la resistencia de pull-up integrada en el 8051 tiene un valor aproximado de 47 Kohmios.



Realizar la función “`unsigned char lee_tecla(void)`” que devuelva el código ASCII de la tecla pulsada(A, B o C), si no se pulsa ninguna devuelve cero. Ignorar la posibilidad de que se puedan pulsar varias teclas a la vez. Emplear la función de retardo “DelayMs()” incluida en el fichero de cabecera “delay.h”. Tener en cuenta que la tolerancia de los componentes suele ser del 5%.

Solución:

Lo primero que hay que hacer es calcular los tiempos de carga del condensador que se obtienen con la pulsación de los pulsadores A, B, C o ninguno de ellos en paralelo con la resistencia de pull-up del 8051 de 47K. Las tolerancias apenas afectan a los tiempos pequeños.

TA=30 ms ± 1 ms
TB=13 ms ± 0.6 ms
TC=2 ms ± 0 ms
TN=94 ms.

El programa quedaría de la siguiente forma:

```
#include “delay.h”
sbit TECLADO P1^0;
struct tiempos {
     unsigned char min;
     unsigned char max;
```

```
}
/* tiempos max y minimos de todas las teclas en orden A,B,C
excepto el caso de ninguna tecla */
struct tiempos teclado[3]={{31,29},{14,13},{2,2}};
```

```
unsigned char lee_tecla(void) {
unsigned char tiempo,i;

TECLADO=0; /* descarga el condensador */
DelayMs(10); /* tiempo opcional de espera a descarga*/
TECLADO=1;  /* carga del condensador */
/* espera que termine la carga del condensador contando el
tiempo de carga aproximado*/
while (!TECLADO) {
     DelayMs(1);
     tiempo++;
     }
/* se desplaza por la tabla de limites y encuentra el primero
que coincida con los margenes de tiempo*/
for (i=0;i<3;i++)
     if ((tiempo<=teclado[i].max) && (tiempo>=teclado[i].min))
               return (i+'A');
/* si hemos llegado hasta aqui significa que o no se ha
pulsado nada o el calculo de tiempo no entra dentro de las
tolerancias marcadas en ese caso devolvemos 0 */
return 0;
}
```