UNIVERSIDAD DE HUELVA
ÁREA DE SISTEMAS
12-FEB-2004

ESCUELA POLITÉCNICA SUPERIOR
INFORMÁTICA INDUSTRIAL II
TIEMPO: 2 Horas, 30 Minutos

CUESTIONES:

C1. Enumera y comenta brevemente las fases de que se compone el análisis de un sistema microcomputador. (0.75 puntos).

Solución:

Las fases en general son las siguientes:

- *Para comenzar realizar un estudio del **esquema eléctrico** y de la **documentación del sistema.***
- *Entender el **propósito del sistema**.*
- *Encontrar los principales **componentes** y la **función** que realiza cada uno ayudándose de las **hojas de características** del fabricante.*
- *Estudiar como **fluyen los datos** que procesa el sistema a través de sus componentes.*
- *Por último, realiza un **diagrama de bloques** del sistema.*

C2. Explica como se realiza y porqué resulta útil el empleo de librerías en un programa. ¿Cuál es la diferencia entre "librería" y "device driver" ? (0.75 puntos).

Solución:

*El empleo de librerías en lenguaje C y en otros lenguajes se realiza dividiendo el problema de programación a **resolver en pequeños subproblemas** que se resuelven de forma independiente y resulta útil para poder centrarse en cada problema de forma más cómoda y también para poder reutilizar el código ya realizado.*

*El empleo de librerías en C se hace en el **entorno de manejo del compilador**, donde cada módulo se puede añadir al proyecto en realización y compilarse de forma separada. Por convención los prototipos de las funciones que se pueden usar del módulo se declaran en un fichero"**cabecera**" con el mismo nombre del módulo y con extensión .h.*

*El "device driver" es una **librería especializada** en resolver el problema de gestionar un periférico concreto. Por ejemplo la librería I2C.C que se encarga de los periféricos I2C es un device driver del subsistema I2C.*

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 50 mseg (0.75 puntos).

Solución:

El temporizador se incrementa cada microsegundo si la frecuencia de reloj es de 12MHz. Los temporizadores se encuentran conectados al oscilador de reloj del microcontrolador a través de un divisor de frecuencia con un factor de 12.

El valor de recarga en el modo 1 para temporizar 50 mseg se calcula teniendo en cuenta que el contador se incrementa cada microsegundo. De esta forma deberá completar 50.000 cuentas antes de desbordarse, por tanto tendrá que cargarse con el valor (65535-50000)=15535. Este valor en hexadecimal es: 3CAF. Por tanto las instrucciones de carga serán:

```
TH0=0x3C;
TL0=0xAF;
```

O bien:

TH0=(65535-50000)>>8;
TL0=(65535-50000)&0x00FF;

C4 Indica en que circunstancias es posible en el caso del 8051 interrumpir a una interrupción, o sea, que se produzca un anidamiento de interrupciones. (0.75 puntos).

Solución:

La circunstancia solo puede ser una: Una interrupción de mayor prioridad que la actual se dispara porque en ese momento está activa.

PROBLEMAS:

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

```
#include <reg51.h>
/**************************************
 Generación de una señal periódica
**************************************/
//Definicion salida
sbit SALIDA=P1.0;
bit flipflop=0;

/**********************************************
 Rutina de servicio del Timer0
**********************************************/
void RSI_Timer0(void) interrupt 1 {
      SALIDA=~SALIDA;   //Complementa P1.0. Pasa 1a0, y 0a1
                        //El bit TFO se borra automáticamente RETI
}

/**********************************************
 Rutina de servicio de interrupción 0
**********************************************/
void RSI_Int0(void) interrupt 1 {
      if (flipflop)     TR0=1;      //arranca timer
      else TR0=0;                   //para timer
      flipflop=~flipflop;
}

/**************************************
 Rutina de inicio
**************************************/
void inicio(void) {
      PT0=1;      //Asigna prioridad alta al Timer 0
      ET0=1;      //Habilita interrupción del Timer 0
      EX0=1;      //Habilita interrupción Ext 0
      EA=1;       //Habilita bit de interrupción general
      TMOD=0x01   //Ml=l y MO=O (Modo 2),
                        // GATE=O y C/T=0 del Timer 0
      TL0=(255-100)&0x00FF;   //La interrupción tiene lugar cada 10 useg
      TH0=(255-100)>>4;       //Pone valor de recarga en TH0
      TR0=1;      //Pone en marcha el contador
}

/**********************************************
```

```
 Rutina Principal
**********************************************/
void main(void) {

inicio();
while (0) {

      }
}
```

Solución:

Errores de Sintaxis:
Línea 6: error C141: syntax error near '.0', expected ';' : La forma de definir un puerto de E/S en el compilador de C de KEIL es esta: `sbit SALIDA=P1^0;`

Línea 37: error C141: syntax error near 'TL0' : Realmente el error de sintaxis está en la línea anterior, falta un punto y coma detrás de la instrucción: TMOD=0x01.

Errores Semánticos:
Línea 20: Las definiciones de ambas rutinas de interrupción apuntan al mismo vector. La interrupción de Int 0 debe apuntar al vector 0x03, o sea, la definición de la función será: `void RSI_Int0(void) interrupt 0.`

Debajo de la línea 23: Puesto que la interrupción trabaja por nivel (IT0=0 por defecto al arrancar el microcontrolador) se debe poner a cero la bandera de interrupción dentro de la RSI. Habrá que añadir: `IE0=0;` *si no, volvería otra vez a entrar en la interrupción porque sigue activado el flag.*

Línea 35: Si el temporizador trabaja en modo 2 se debe cargar 0x02 en TMOD: `TMOD=0x02;`

Línea 36 y 37: Si se emplea el temporizador en modo auto-recarga para 10 microsegundos se deben cargar otros valores en TH0 y TL0: `TL0=TH0=(255-10);` *porque el contador se incrementa cada microsegundo a Fclk=12 MHz.*

Línea 48: El bucle principal del programa, que está vacío, no es tal, debería ser: `while (1)`

P2. (3 puntos) La empresa de golosinas PEPE S.A. está desarrollando un expendedor de caramelos. Para el desarrollo del expendedor se ha contratado a la empresa MONEY S.A. el desarrollo de un monedero que devuelve por un bus de 8 bits conectado al puerto P0 el peso de la moneda en gramos cuando se ha introducido una moneda y también controla el LCD, el dispensador de caramelos y los cajones de monedas. Las librerías proporcionadas por el equipo de MONEY S.A. 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 caracter ASCII

MON.C/MON.H		
Función	Objetivo	Parámetro
`void mon_init(void)`	Inicializa Monedero	Ninguno
`void mon_dispensa(bit)`	Activa dispensador caramelo	1=abre, 0=cierra
`void mon_abre_cajon(unsigned char)`	Abre cajon de monedas	Código de cajón
`void mon_cierra_cajon(unsigned char)`	Cierra cajon de monedas	Código de cajón

El módulo de pesado de moneda coloca en todas las patillas del puerto 0 del 8051 un nivel lógico cero cuando no se ha introducido ninguna moneda. Cuando se introduce alguna moneda se obtiene en el puerto 0 el peso de la moneda que se haya introducido en gramos **en código BCD**. Hay que tener en cuenta las tolerancias de las monedas que se indican en la tabla. Cualquier moneda que no cumpla con el peso de la tabla será rechazada.

Moneda	Peso	Código Cajón
5 céntimos	5 gramos ±2	0
10 céntimos	14 gramos ±2	1
20 céntimos	19 gramos ±2	2
50 céntimos	25 gramos ±2	3

El programa debe emplear la función "mon_abre_cajon" de forma inmediata para abrir el cajón correspondiente. Para ello se le pasa como parámetro para que la moneda sea aceptada. Si no se abre el cajón a tiempo, la moneda será devuelta al cliente (cuando por ejemplo se introduce una moneda que supere el resto del precio del caramelo). El cajón deberá estar abierto durante 1seg, luego deberá cerrarse con la función "mon_cierra_cajon".

La función mon_dispensa abre el dispensador con un 1 y con un 0 lo cierra. Para una dispensación correcta del caramelo se necesita abrir durante 500ms.

La aplicación realiza las siguientes tareas en el orden indicado:

1.- Escribir en pantalla el nombre de la empresa.
2.- Esperar 1s.
3.- Escribir en pantalla mensaje pidiendo dinero (el caramelo cuesta 85 céntimos).
4.- Esperar entrada de monedas y contabilizarlas abriendo los cajones correspondientes.
5.- Si se introduce una moneda que supere el importe restante será rechazada (basta con no abrir ningún cajón).
6.- Al introducir importe exacto abrir el dispensador 0.5 seg.
7.- Esperar un tiempo.
8.- Volver a empezar.

Se pueden escribir las funciones auxiliares que se crea necesario y emplear la librería de temporización (DelayMs, DelayS) para los retardos. Se pide realizar el diagrama de flujo y la función main del programa.

Solución:

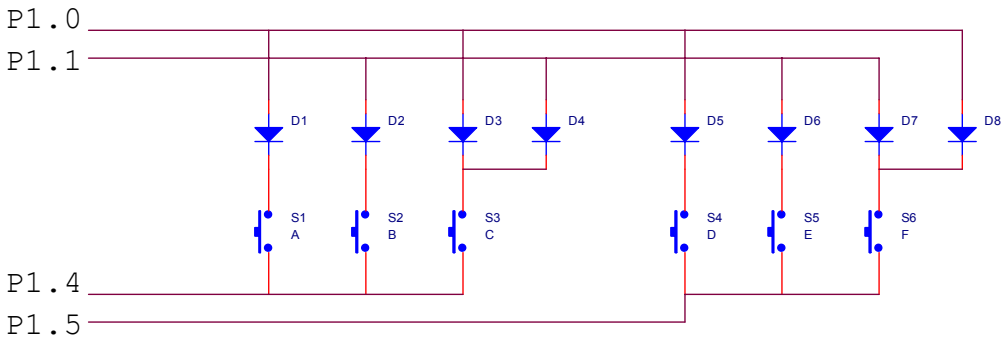
```
#include <reg51.h>
#include "mon.h"
#include "delay.h"
#include "lcd.h"

#define TOTAL 85
// Tabla de codigos, ojo codigo 0 no es ningun cajon y 1 para
// el cajon 0
unsigned char code cod_a_valor[]={0,5,10,20,50};

unsigned char pide_moneda(void) {
unsigned char peso;
```

```
      // espera entrada de moneda
      while (P0==0);
      peso=P0;
      //moneda de 5 centimos
      if ((peso>0x02)&&(peso<0x08)) return 1;
      //moneda de 10 centimos
      if ((peso>0x07)&&(peso<0x13)) return 2;
      //moneda de 20 centimos
      if ((peso>0x17)&&(peso<0x23)) return 3;
      //moneda de 50 centimos
      if ((peso>0x47)&&(peso<0x53)) return 4;
      // devolvemos el código especial 0 que significa
      // moneda defectuosa
      return 0;
}

void main(void) {
unsigned char importe,cod_moneda;
// Inicializa subsistemas
lcd_init();
mon_init();

while(1) {
      // Saca mensajes de bienvenida
      lcd_puts("PEPE S.A.");
      DelayS(1);
      lcd_puts("Introduzca moneda");
      lcd_puts("Solo ");
      lcd_putch((TOTAL/10)+'0');
      lcd_putch((TOTAL%10)+'0');
      lcd_puts(" centimos");
      lcd_puts("Introduzca importe exacto");

      importe=0;   //pongo a cero el importe inicial
      do {
          cod_moneda=pide_moneda(); // codigo de moneda
          // Acepta moneda si (importe + valor moneda) < TOTAL
          // si cod_moneda es 0 significa moneda defectuosa
          if (((importe+cod_a_valor[cod_moneda])<TOTAL)
              && (cod_moneda!=0)) {
              mon_abre_cajon(cod_moneda-1);
              DelayMs(500);// Abre cajon 500ms
              mon_cierra_cajon(cod_moneda-1);
              importe+=cod_a_valor[cod_moneda];
          }
          // caso contrario no abre cajon y moneda rechazada
          // Saca por pantalla importe introducido hasta ahora
          lcd_puts("Introducido ");
          lcd_putch((importe/10)+'0');
          lcd_putch((importe%10)+'0');
          lcd_puts(" centimos");
      } while (importe!=TOTAL);
```

```
        // El usuario ha conseguido introducir importe exacto
        lcd_puts("Gracias por su compra");
        mon_dispensa(1);
        DelayMs(500);// Abre dispensador 500ms
        mon_dispensa(0);
        DelayS(3);
        //Espera un tiempo para que el usuario coja producto
        }
}
```

El diagrama de flujo es bastante simple a partir del código fuente y del enunciado del problema.

P3. (2 puntos) Se pretende realizar una librería para la lectura de un teclado de seis teclas conectado a los bits P1.0, P1.1, P1.4 y P1.5 del 8051. Para poder dar servicio a más pulsadores empleando tan solo 4 puertos de E/S se han colocado diodos. Recordar que los diodos conducen cuando la corriente fluye por ellos en sentido directo.



Realizar la función "`unsigned char lee_tecla(void)`" que devuelva el código ASCII de la tecla pulsada. Ignorar la posibilidad de que se puedan pulsar varias teclas a la vez.

```
#include <reg51.h>

sbit COL_0=P1^4;
sbit COL_1=P1^5;
sbit FIL_0=P1^0;
sbit FIL_1=P1^1;

//Al leer filas podemos obtener 4 codigos:
// 00 significa pulso tecla con dos diodos
// 01 significa pulso tecla con diodo en P1.1
// 10 significa pulso tecla con diodo en P1.0
// 11 significa no pulso ninguna tecla
unsigned char code
 tabla_tecla[]={'C','B','A',0,'F','E','D',0};

unsigned char lee_tecla(void) {

// Filas P1.0 y P1.1
// Columnas P1.4 y P1.5
```

```
// Escaneo por columnas poniendo ceros

COL_0=0;
COL_1=1;
// si no se pulso nada
if ((FIL_0) && (FIL_1)) {
      // cambio de columna
      COL_0=1;
      COL_1=0;
      // Si no se pulso tampoco nada
      if ((FIL_0) && (FIL_1)) {
          //Devuelvo 0 ninguna tecla
          return 0;
      } else {
          //Tecla de la Columna 1
          return tabla_tecla[(P1&0x03)+4];
      }
} else {
      //Tecla de la Columna 0
      return tabla_tecla[P1&0x03];
}
// aqui no se va a llegar nunca
}
```