UNIVERSIDAD DE HUELVA
ÁREA DE SISTEMAS
12-FEB-2002

ESCUELA POLITÉCNICA SUPERIOR
INFORMÁTICA INDUSTRIAL II
TIEMPO: 3 Horas

CUESTIONES:

C1. Comentar las diferencias básicas en el hardware y en el software de un sistema basado en microprocesador y un sistema basado en microcontrolador (0.5 puntos).

Diferencias básicas:

- ***Buses**. El sistema basado en microcontrolador (SBMC) no suele contener buses de direcciones, datos y control.*
- ***Periféricos**. El SBMC suele contener varios en el propio chip.*
- ***Velocidad**. El SBMC suele ser bastante más lento.*
- ***Aplicación**: El SBMC se emplea en tareas sencillas de control donde no es necesaria una gran capacidad de cálculo.*
- *Etc…*

C2. ¿Cuál es el objetivo que se persigue con la modularización de programas? Explica que se pone en el fichero *.h y como se utiliza este fichero en el código del programa principal. (0.5 puntos).

*El objetivo básico es la **reutilización del código**, otros pueden ser: generar documentación, reducir tiempos de compilación, etc. En el fichero *.h se coloca **como mínimo** la lista de prototipos de funciones que se exportan, esto es, que se permite llamar desde otro módulo o el módulo principal, precedido cada uno por la palabra clave **extern**.*
En el modulo principal se inserta la línea #include "modulo.h" y se llama a cada función exportada a medida que vaya haciendo falta.

C3. Comenta la diferencia entre el tratamiento del puerto serie con o sin interrupciones. Indica el mecanismo de funcionamiento en ambos casos. (0.5 puntos).

*La diferencia principal es que al usar interrupciones el programa principal se desliga de la tarea de tener que **estar esperando** a que llegue un carácter o esperando a que se termine de enviar el carácter precedente antes de mandar el siguiente. Mediante interrupciones el programa principal escribe en memoria lo que quiere enviar o recoge de memoria lo que ha llegado y no se preocupa de cómo llega o se envía. Esto es función de la **rutina de servicio de interrupción**. La rutina de servicio de interrupción lee de la memoria los nuevos caracteres a enviar y espera a que el HW termine llamándola para enviar el siguiente carácter, mientras esto ocurre el programa principal se está ejecutando y realiza procesamiento útil.*

Si no se emplean interrupciones el programa principal se va a quedar esperando a que llegue un carácter del puerto serie cuando necesite una entrada de datos. También se queda esperando a que se termine de transmitir totalmente un carácter antes de enviar el siguiente y en este tiempo el programa principal no suele realizar procesamiento útil. Ese funcionamiento puede resultar inconveniente dependiendo del programa que estemos realizando.

C4. Enumerar los diferentes sistemas de interconexión (cableado) en RED LOCAL y compararlos indicando sus ventajas e inconvenientes. (0.5 puntos).

Los sistemas de cableado básico son:

- ***Par trenzado**. Es bastante inmune al ruido al estar trenzado, es barato, se emplea en las redes locales.*
- ***Cable coaxial**. Dispone de una malla de cobre que apantalla el conductor de señal y lo hace inmunes a ruidos, tiene un gran ancho de banda por lo que puede transmitir gran cantidad de datos. Se suele emplear para señales moduladas como las de CATV (televisión por cable).*

- ***Fibra Óptica***. *Tiene una atenuación de la señal muy baja al emplearse haces de luz coherente (láser) para la transmisión por lo que se emplea cuando se quiere transmitir señales a muy larga distancia.*
- ***Ondas de Radio***. *Tiene un alcance reducido pero la ventaja de que no es necesaria una infraestructura de cableado previa.*

C5. Enumera el nombre y la localización física de los espacios de memoria que se pueden emplear en el C51 de KEIL para el 8051. (0.5 puntos).

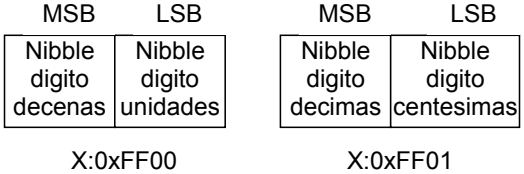
Los espacios de memoria son:

- ***CODE****: Zona de código, se emplea para el código y tablas constantes, se encuentra dentro del chip de 0x0000 hasta 0x0FFF en el 8051 y de 0x1000 a 0xFFFF fuera del chip opcionalmente.*
- ***DATA****: Zona de datos interna, se emplea para las variables del programa, se encuentra dentro del 8051, desde 0x00 hasta 0x7F, son 128 bytes.*
- ***IDATA****; Zona de datos interna accesible mediante direccionamiento indirecto. Es la zona especial que solo tiene el 8052 accesible mediante direccionamiento indirecto (MOV @R1,A), se utiliza para la pila principalmente, desde 0x80 a 0xFF.*
- ***PDATA****: Zona de datos paginada, son 256 bytes de ram externa desde 0x00 a 0xFF que se acceden sin utilizar más bits de direcciones, usada con ram externa.*
- ***XDATA****: Zona de datos externa, se emplea para grandes cantidades de datos, es lenta, son 64Kbytes desde 0x0000 hasta 0xFFFF. Esta fuera del chip y viene definida por los componentes de RAM instalados.*
- ***BDATA****: Zona de datos direccionable a bit, es la zona del 8051 direccionable a nivel de bit, se emplea para guardar banderas o variables de tipo bit.*

PROBLEMAS:

P1. Un distribuidor de combustibles nos ha pedido que realicemos un diseño de una maquina expendedora de combustible. Esta maquina dispone de tres tipos de combustible y controla a tres bombas que dispensan cada combustible. La manguera dispone de un sensor de llenado en su extremo.

El equipo dispone además de un contador externo que se va actualizando, accesible en las direcciones X:0xFF00(MSD) y X:0xFF01(LSD) de memoria externa que en BCD empaquetado contienen la cuenta actual del número de litros que se están sirviendo. Se pone a cero cuando se cuelga la manguera automáticamente.



El sistema debe comenzar a funcionar al levantar la manguera, activando la bomba adecuada al pulsar el gatillo y contabilizando los litros (multiplica por precio y saca a pantalla). Emplear aritmética entera con precisión de dos decimales en todos los casos (multiplicar por 100 y usar enteros). Si el usuario no pulsa los botones de premarcación el sistema para cuando se deja de pulsar el gatillo o se detecta deposito lleno.

Se muestra continuamente en pantalla el número de litros consumidos y su precio. El precio por litro lo debéis introducir vosotros en una tabla fija en memoria de código y mostrarlo al comenzar. Al colgar la manguera el sistema queda en reposo a la espera que se levante de nuevo la manguera.

Las funciones de C de la librería "surtidor.h" que se pueden usar son las siguientes;

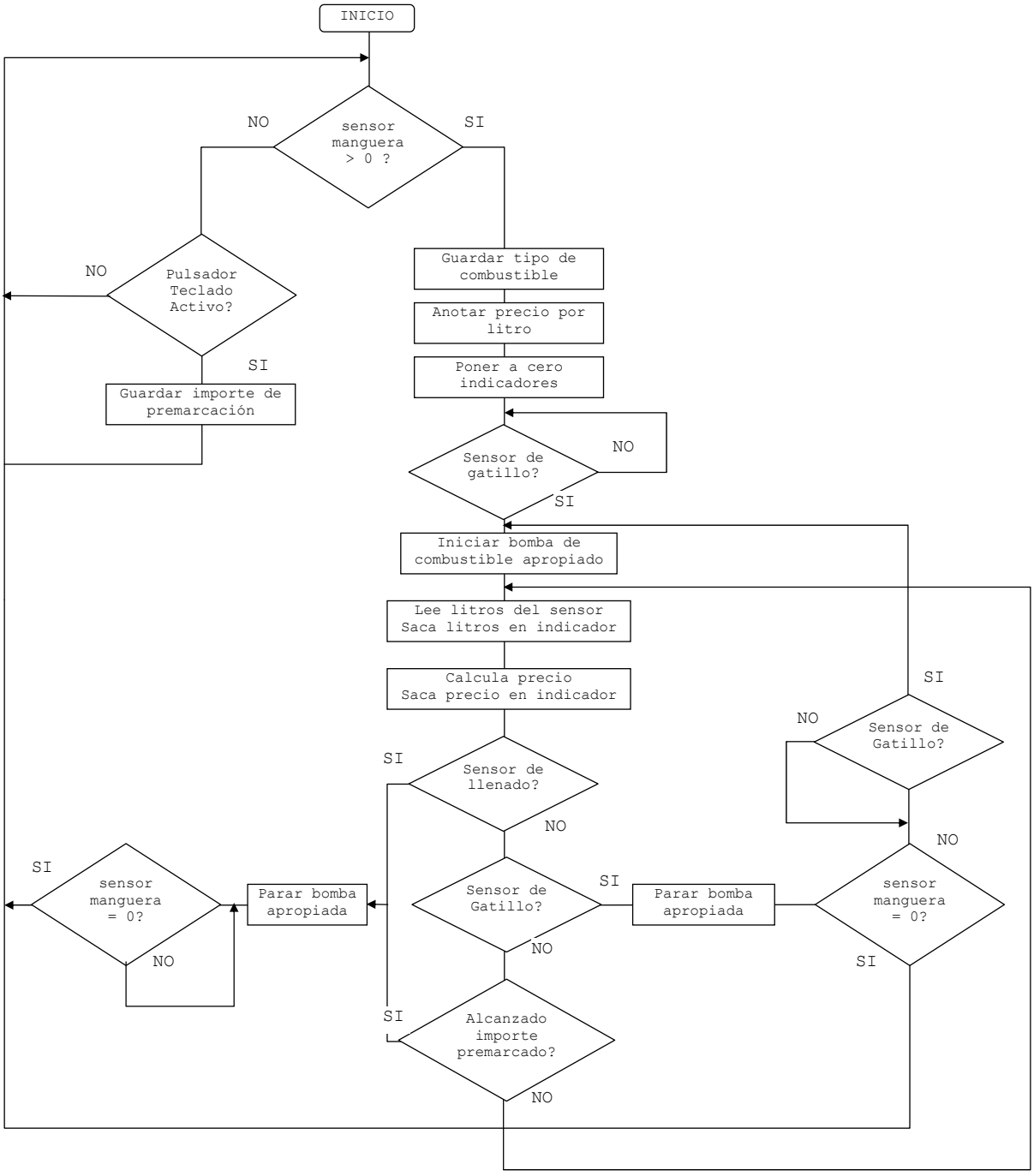
```
void sacar_litros(unsigned int);
// saca los litros en pantalla en BCD empaquetado con cuatro dígitos ##.##
void sacar_precio(unsigned char);
// saca el precio en pantalla en euros por litro en BCD empaquetado con dos 0.##
void sacar_precio_total(unsigned int);
// saca el precio en pantalla en euros en BCD empaquetado con cuatro dígitos ##.##
void iniciar_bomba(unsigned char);  //arranca la bomba de combustible adecuada 1=SP95, 2=SP97, 3=GasOil
void parar_bomba(unsigned char);  //para la bomba de combustible adecuada 1=SP95, 2=SP97, 3=GasOil
bit sensor_lleno(void);  // devuelve 1 cuando esta lleno el deposito
bit sensor_gatillo(void);  // devuelve 1 cuando esta pulsado el gatillo de la manguera
unsigned char sensor_manguera(void); //devuelve 0 si no se ha levantado manguera
// 1 cuando Sin Plomo 95
// 2 cuando Sin Plomo 97
// 3 cuando Gasoil
```

Se pide:

a) Realizar un diagrama de flujo del programa principal.
b) Realizar el programa principal que responde al diagrama de flujo anterior, realizar las llamadas necesarias a las funciones facilitadas y a otras que consideréis que resultan necesarias. ¿Cómo modularizarías el problema?
c) Realizar la función de lectura de litros de combustible de forma que devuelva un entero con los litros x 100: unsigned int lectura_combustible(void);

SOLUCIÓN:

a) El diagrama de flujo podría tener la siguiente forma:



b) El código fuente para implementar este diagrama de flujo está basado en dos bucles principales:

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

#define MSD XBYTE[0xFF00]
#define LSD XBYTE[0xFF01]

#define uchar unsigned char
#define uint unsigned int

/* Variables del programa */
```

```
uint marca, precio, litros, premarcado;
uchar seleccion;
uchar precio_por_litro[3]={78,90,88};

void main(void) {

while (1) {
      premarcado=9999;
      seleccion=sensor_manguera();
      if (seleccion>0) {
            // sacamos precio por litro para informar al usuario
            sacar_precio(pasobcd(precio_por_litro[seleccion]));
            // puesta a cero del display
            sacar_litros(0);
            sacar_precio_total(0);
            while(!sensor_gatillo()); // espera que se pulse gatillo
            do {
                  iniciar_bomba(seleccion);
                  // llamada a funcion que nos devuelve los litros
                  // contabilizados
                  litros=lectura_combustible();
                  sacar_litros(pasobcd(litros));
                  // calculo del precio en binario
                  precio=litros*precio_por_litro[seleccion];
                  sacar_precio_total(pasobcd(precio));
            }
while (!sensor_lleno() && !sensor_gatillo() && (precio<premarcado));
            // condiciones de parada para la bomba de combustible
            parar_bomba(seleccion);
      // caso especial del gatillo que se suelta y vuelve a pulsar
            if (!sensor_gatillo()) {
                  while (sensor_manguera()!=0)
                        if (sensor_gatillo()) do {
                              iniciar_bomba(seleccion);
                              litros=lectura_combustible();
                              sacar_litros(litros);

      precio=litros*precio_por_litro[seleccion];
                              sacar_precio_total(precio);
                        } while (sensor_gatillo());
                  parar_bomba(seleccion);
            }
      premarcado=9999;
      }
      if (marca=pulsador_pre()) {
            if (premarcado==9999) premarcado=0;
            premarcado+=marca;
            }
      }
}
```

He añadido las funciones:
// devuelve 1000 si se ha pulsado el botón de 10 euros o 100 si se ha pulsado el boton de 1 euro, 0 en caso contrario.
Unsigned int **pulsador_pre**(void);
// devuelve el valor de los litros con dos dígitos decimales de precisión multiplicado por 100, o sea si se están sirviendo 10.23 litros devolvería 1023. Este valor se ira incrementando

unsigned int **lectura_combustible**(void);
// pasa un valor unsigned int (16 bits) a BCD empaquetado 4 dígitos, por ejemplo el numero //1234 se convertiría en 0x1234. Es necesario para la representación en los displays del //surtidor.
Unsigned int **pasobcd**(unsigned int);

Para modularizar el problema se podría realizar los siguientes módulos:

- **Sensores**: incluiría las funciones sensor_lleno, sensor_gatillo, sensor_manguera, lee_litros y pulsador_pre.
- **Visualizadores y bomba**: incluiría las funciones sacar_litros, sacar_precio, sacar_precio_total, iniciar_bomba y parar_bomba.
- **Utilidades**: incluiría la función pasobcd.

c) Teniendo en cuenta los anteriores #define XBYTE, la función pedida se puede realizar de la siguiente forma:

```
unsigned int lee_litros(void){
      return (MSD&0xF0)*1000+(MSD&0x0F)*100+(LSD&0xF0)*10+(LSD&0x0F);
      }
```

Empleando máscaras y multiplicando por potencias de 10 para convertir de BCD a binario.

P2. (2,5 puntos) En el manual de un fabricante de placas de desarrollo del 8051 nos encontramos con el siguiente mapa de memoria:

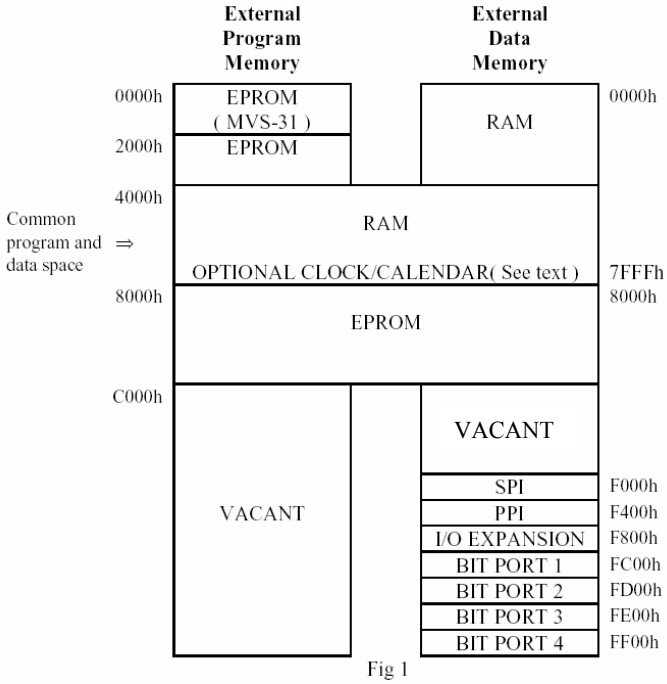


Fig 1

a) ¿Cual es el número de chips y tamaño de la RAM externa instalada?
b) Para los puertos de E/S externos se emplea en todo caso decodificación incompleta a partir de la dirección indicada en la tabla hasta la dirección del siguiente puerto (no se tienen en cuenta los 8 bits bajos). Los dispositivos SPI y PPI ambos tienen patillas /RD, /WR y /CS1 y /CS2. Para direccionar la zona "I/O Expansion" tan sólo tenemos que generar una señal /CSEXP. Realizar el decodificador de estos puertos y detalla la conexión de las señales indicadas a las señales /RD, /WR y A0-A15 del 8051 a través de este decodificador.

SOLUCIÓN:
a) Como resulta evidente hay dos módulos de RAM. Uno de ellos se puede acceder también a través de la zona de código porque se ha usado una AND entre las señales /PSEN y /RD para activar la RAM compartida. Pero esto no es lo que nos preguntan.

La primera RAM externa se extiende desde 0x0000 a 0x3FFF, luego ocupa 16Kbytes.
La segunda RAM externa se extiende desde 0x4000 a 0x7FFF, luego ocupa otros 16Kbytes. En total la RAM externa instalada es de 32Kbytes.

c) El primer rango es: 0xF000-0xF3FF, el segundo es 0xF400-0x7FF y el tercero es 0xF800-0xFBFF. El esquema quedaría de la siguiente forma: