EXAMEN SED, "Problemas tipo" para la primera parte de la asignatura

De esta parte, un examen podría incluir entre 2 y 3 problemas cortos; o bien un problema extenso.

Ejemplos de PROBLEMAS CORTOS:

1) Un PIC recibe una señal PWM en la línea RB0 configurada como entrada de interrupción externa. Sabemos que el periodo de dicha señal puede ser de hasta 0,5 segundos. Redacta la rutina de interrupción externa para que, interrumpiendo el mínimo tiempo posible la actividad normal de procesador, ponga a '1' o a '0' la salida RA0 si el tiempo en alta es mayor o menor que el tiempo en baja en cada periodo de la señal PWM. Utilizad el TIMER0 para las medidas de tiempo, indicando cómo se debe configurar si la frecuencia de reloj es de 4MHz

.

2) Mediante un bus I2C el PIC18F4520 tiene conectados un chip de memoria EEPROM 24C64 (dirección I2C: 1010XXX) y un convertidor DAC (dirección I2C: 101000X) conectado a un altavoz. Cuando el DAC es seleccionado para escritura, el siguiente byte que reciba a través de bus lo convierte directamente a tensión analógica (con un retardo que podemos despreciar). La EEPROM contiene los valores muestreados y digitalizados de una señal, que ocupan todas sus posiciones. El PIC debe estar normalmente en modo de bajo consumo, y cada vez que recibe una petición de interrupción externa se deberá reproducir en el altavoz la señal grabada en la EEPROM.
   a) Redacta el bucle de programa que implemente la funcionalidad descrita.
   b) Suponiendo que se llega al límite del modo I2C fast (SCK a 400KHz), estima el tiempo mínimo entre cada dos muestras consecutivas reproducidas en el altavoz, y por tanto el límite frecuencia de la señal digitalizada que podríamos reproducir correctamente.

NOTAS: No se pide que redactéis la rutina de atención interrupción externa. Se supone que el DAC es de 8 bits.

3) Atenuador de señal PWM: Un PIC recibe una señal PWM en la línea RB0 configurada como entrada de interrupción externa, y debe generar en su salida RA0 otra señal PWM del mismo periodo, pero cuyo tiempo en alta sea la mitad del que tiene la señal de entrada.
   a) Redacta la rutina de interrupción que debe decodifica la señal de entrada.
   b) Especifica la configuración de temporizador o temporizadores que utilices.
   c) Redacta el bucle de programa principal que genera la señal de salida.

NOTA: no se conoce el periodo de la señal de entrada, sólo sabemos que su valor es inferior a 50ms. La frecuencia de reloj del sistema es de 4MHz.

4) Un microcontrolador PIC 18F4520 tiene conectado un chip de memoria EEPROM-I2C de 32kbits, modelo 24C32. El PIC debe mantenerse el mayor tiempo posible en modo de bajo consumo, despertando cada 32 segundos (aproximadamente) para captar el valor de una señal analógica de su entrada RA0, y guardar su valor en la siguiente posición de la EEPROM. Cuando se llene la EEPROM comenzará a re-escribir desde la posición 0.

a) Asumiendo que el chip EEPROM tiene como dirección I2C: 1010111; redacta el bucle de programa, especificando además las directivas o parámetros de configuración que deban fijarse para que el sistema tenga el comportamiento descrito.
b) Estima qué porcentaje del tiempo total de funcionamiento se mantiene el PIC en modo de bajo consumo.

.

5) Un PIC-18F4520 ejecuta un programa consistente en la visualización continua de los 10 valores (int8) de un vector. La visualización se hace en una pantalla LCD de 16x2 caracteres (como la utilizada en prácticas), mostrando cada valor durante aproximadamente 1 segundo.
Se quiere que si se produce cualquier reset en el PIC, excepto por conexión normal de la alimentación (POR), el PIC arranque y rescate dichos valores de las posiciones 0 a 9 de su EEPROM interna, mientras que si se ha producido un POR, deberá mostrar todos los valores iguales a 0xFF.
Redacta el programa para que el sistema tenga el comportamiento descrito.

NOTA: La función del CCS restart_cause() devuelve un valor que indica la causa del último reset que se haya producido en el PIC. La función devuelve el valor NORMAL_POWER_UP cuando se ha producido un reset normal por conexión de la alimentación.

6) Mediante el convertidor A/D de un microcontrolador PIC 18F4520, en los canales AN0 y AN1, se captan dos señales analógicas con la precisión máxima que permite el convertidor (10 bits). El PIC tiene conectado un chip de memoria EEPROM-I2C de 32kbits, modelo 24C32, configurado con la dirección I2C: 1010011. El reloj del sistema es de 4MHz.
Cada vez que el PIC recibe un pulso en su entrada RB0, debe proceder leyendo los valores de las dos señales analógicas y guardando su suma en las siguientes posiciones de la memoria. Cuando se llene la EEPROM comenzará a re-escribir desde la posición 0. Además, si transcurren más de 8 segundos (aproximadamente) sin que se reciba un nuevo pulso en RB0, se pondrá en alta un bit de aviso en la salida RC0 y las siguientes escrituras se harán también re-escribiendo a partir de la posición 0.

c) Redacta la rutina de interrupción externa que implemente la funcionalidad descrita.
d) Redacta la rutina de interrupción del temporizador TMR0 de manera que, con un único desbordamiecto, realice el limitador de tiempo de 8 segundos. Indica cómo debe estar configurado el TMR0.

.

7) Un PIC18F4520 con reloj de 8 MHz debe captar cada 8 segundos (valor aproximado) el valor analógico de su entrada RA0 y actualizar el valor de la patilla RB1 dependiendo del valor medido, manteniéndose el resto del tiempo en modo de bajo consumo. Redacta el bucle que realice la operación indicada, actuando como un comparador con histéresis: RB1 pasa a valer '1' cuando el valor analógico supere los 3Volt., y vuelve al valor '0' cuando desciende de 2Volt.

8) Un microcontrolador PIC tiene conectados dos dispositivos I2C (D1 y D2), que tienen la dirección 101011X, disponiendo de un terminal A0 para fijar el bit menos significativo de su dirección I2C. Para escribir o leer un dato de estos dispositivos, estos deben recibir previamente las órdenes 0x55 o 0xAA respectivamente. Dibuja el esquema de conexión y configuración del sistema. Redacta un bucle de programa que iterativamente lea datos del dispositivo D1 y los escriba en el dispositivo D2.

Ejemplos de PROBLEMAS CORTOS con soluciones:

9) Se ha redactado una rutina de interrupción INT_RB para atender la lectura de un teclado matricial 4x4 conectado al puerto B de un PIC18F4520. Cada vez que se pulsa una tecla, la rutina efectúa su lectura y devuelve el control al programa principal. Queremos añadir una interrupción por TMR0 para que el micro pase a bajo consumo si transcurren más de 8 segundos sin pulsar alguna tecla. Redacta la rutina #INT_TMR0 de manera que se resuelva con un único desbordamiento de TMR0, suponiendo que el reloj del sistema es de 4MHz. Indica también las líneas de código extra que deban introducirse en la rutina #INT_RB y las de configuración del temporizador en el programa principal.

SOLUCIÓN: con un divisor (pre-scaler) de 128, el desbordamiento a los 8 segundos (aproximadamente) se consigue cargando 3036 inicialmente, en TMR0. Esto deberá hacerse cada vez que se ejecute la rutina de atención de #INT_RB.

. Rutina del temporizador:

```
#INT_TMR0
Tocadormir() {
        Sleep();
        }
```

En la rutina #INT_RB, debe incluirse:

```
SET_TIMER0 (3036);
```

Y en el main:

```
SETUP_TIMER_0(RTCC_INTERNAL |  RTCC_DIV_128);
```

---

10) Para un PIC18F4520 con reloj de 8 MHz, redacta un programa que capte continuamente, con una resolución de 8 bits, los valores de tres señales analógicas (voltajes entre 0V y 5V), que calcule su promedio y lo represente (en binario, con 8 bits) en las líneas del puerto B.
Estima la frecuencia máxima que pueden tener las señales para que su tratamiento sea correcto.

```
Int16 analog1, analog2, analog3;          //se supone ADC=8 en la directiva DEVICE del archivo 18F4520.h
long resultado;
setup_adc_ports(AN0_TO_AN3);          //RA0 a RA3 entradas analógicas (ver 18F4520.h)
setup_adc(adc_clock_div_8);    //Ajusta tiempo de conversión de cada bit  a 8*Tosc
set_adc_channel(1);
 delay_us(10);
analog1=read_adc();
set_adc_channel(2);
delay_us(10);
analog2=read_adc();
set_adc_channel(2);
delay_us(10);
analog3=read_adc();
resultado= (analog1+analog2+analog3)/3; // si las variables fueran int8, el resultado sería incorrecto
output_B(resultado); //aunque resultado es int16, su valor cabe en los 8 bits menos significativos que enviamos al puerto
```

- Cada valor de señal captado requiere aproximadamente un tiempo: Ta/d + Ts/h = 12ciclos*8*Tosc + 10us = 12us+10us
- Los 12 ciclos corresponden a una conversión completa: conversión de 10bits más dos ciclos adicionales que emplea el convertidor del PIC.
- Al tener que captar las tres señales, cada canal se muestrea con una frecuencia de 1/(22us*3) = 15151,5Hz
- Imponiendo que la frecuencia de muestreo de cada canal sea al menos el doble, la frecuencia de las señales deberá ser inferior a 7575,7Hz. En la práctica este valor será inferior, ya que estamos despreciando el tiempo de ejecución de varias instrucciones del bucle; en particular la que calcula el promedio.

11) Queremos realizar un sensor controlado con un PIC que tenga una funcionalidad similar al sensor de ultrasonidos utilizado en prácticas, pero que realice la medida con luz; con un emisor de infrarrojos y un detector de luz.

c) ¿Qué frecuencia debería tener el reloj del sistema para que el sensor pueda medir una distancia mínima de 15 cm?

En recorrer 30 cm (ida y vuelta) la luz invierte: 0.3m / 3*10^8m/s. El tiempo mínimo que puede medir el PIC corresponde a una cuenta de uno de sus temporizadores, es decir: 4*Tosc. Por tanto Fosc=4*10^9Hz (aproximadamente 4GHz)

d) Suponiendo que el PIC estuviese funcionando con la frecuencia calculada, y asumiendo un temporizador de 16 bits, sin pre-scaler, calcule la distancia máxima que podría llegar a medir correctamente.

Aproximadamente 15cm * 2^16 = 9830 metros

12) Un PIC tiene conectado un servomotor como los explicados en teoría y utilizados en prácticas (posiciones extremas entre -90º y +90º con anchuras de pulso entre 1 y 2 milisegundos). Redacta en C un bucle que mantenga el eje del servomotor con una orientación de +45º.

Para mantener la posición de +45º, el servo debe recibir continuamente pulsos de 1750us, separados unos 20ms (este valor no es crítico). Si el terminal de control del servo está conectado al pin RA0 (por ejemplo) y despreciando los retardos adicionales que introducen las instrucciones del bucle, tendríamos:

```
while (1) {
output_high(PIN_A0);
delay_us(1750);
output_low(PIN_A0);
delay_ms(20); //puede ponerse un valor menor
}
```

13) Con chips de memoria EEPROM I2C de 32Kbytes, que tienen la dirección de dispositivo I2C: 1010XXX, se desea configurar una memoria de 64Kbytes. Dibuja su esquema de interconexión con un microcontrolador PIC, y redacta dos funciones en C que implementen las operaciones de lectura y de escritura en dicha memoria.

DIBUJO: Dos chips de memoria con sus líneas SDA y SCK conectadas en paralelo a las correspondientes del PIC, y sus líneas de dirección difiriendo en al menos un valor. Por ejemplo, si uno de los chips (CHIP0) lo ponemos con sus terminales A0, A1 y A2 conectados a GND, y el otro chip (CHIP1) igual salvo A0 conectado a Vdd, las direcciones I2C de ambos chips serán: 1010000 y 1010001. En tal caso, para direccionarlos en I2C emplearemos los siguientes bytes de control: 0xA0 o 0xA2 para escritura en CHIP0 o en CHIP1, y 0xA1 o 0xA3 para lectura del CHIP0 o del CHIP1.

Para direccionar 64KBytes necesitamos 16 bits, de los cuales podemos establecer que el más significativo sea el usado para distinguir entre ambos chips de memoria. Así, las posiciones 0x0000 a 0x7FFF estarían en el CHIP0, y las posiciones 0x8000 a 0xFFFF estarían en el CHIP1.

Las funciones en C para lectura y escritura en la memoria de 64KBytes deben decidir a cuál de los chips debe accederse dependiendo de la dirección indicada como argumento. Una posible implementación de estas funciones es:

```
BYTE read_ext_eeprom64(long int address) {
BYTE data;
int i2c_address_read, i2C_address_write;
if(address>>15) {  // en CHIP1 cuando msb=1
i2c_address_write= 0xA2;
i2c_address_read= 0xA3; }
else{             //en CHIP0
i2c_address_write= 0xA0;
i2c_address_read= 0xA1; }
  i2c_start();
  i2c_write(i2C_address_write);
  i2c_write((address>>8)&0x7f);
  i2c_write(address);
  i2c_start();
  i2c_write(i2c_address_read);
  data=i2c_read(0);
  i2c_stop();
  return(data);
}
```

```
void write_ext_eeprom64(long int address, BYTE data) {
/*en esta version no estamos introduciendo espera hasta completar la escritura*/
int i2C_address_write;
if(address>>15)  // en CHIP1  cuando el bit más significativo es 1
i2c_address_write= 0xA2;

else        //en CHIP0
i2c_address_write= 0xA0;

i2c_start();
i2c_write(i2C_address_write);
i2c_write((address>>8)&0x7f);
i2c_write(address);
i2c_write(data);
i2C_stop;
}
```

Ejemplos de PROBLEMAS EXTENSOS:

1) Un vehículo móvil autónomo debe avanzar corrigiendo automáticamente su trayectoria, evitando los obstáculos que encuentre en su camino. Para ello el vehículo dispone de dos sensores de ultrasonidos SFR05 ubicados a ambos lados de su parte frontal. El vehículo está controlado por un PIC 18F4520 que mediante los terminales RA0 y RA1 gobierna su movimiento de acuerdo con la siguiente asignación:

RA0	RA1	
0	0	Parado
1	1	Avanza en línea recta
0	1	Avanza girando hacia la derecha
1	0	Avanza girando hacia la izquierda

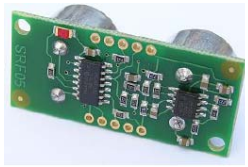
- Diseña el sistema de control de movimiento del vehículo, efectuando la lectura de los sensores de ultrasonidos media interrupciones:

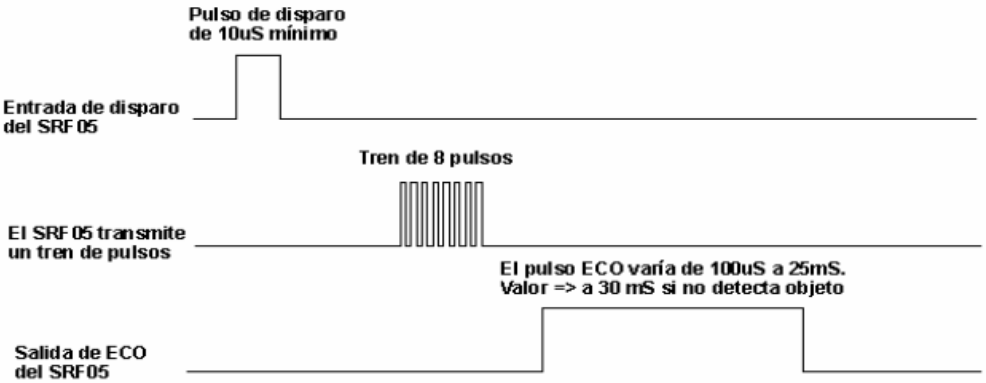
a.1) Indica las patillas del PIC elegidas para la conexión de los sensores.

a.2) Redacta el código del software de control.

NOTA: En ausencia de obstáculos, el vehículo debe avanzar en línea recta.

Recordamos el funcionamiento del sensor SRF05:





La duración del pulso ECO de salida varía entre 100µS y 25mS en función de la distancia entre las cápsulas del módulo SRF05 y el objeto. La velocidad del sonido es aproximadamente 29.15 µs/cm. Como el sonido realiza un recorrido de ida y vuelta, el tiempo total por unidad de longitud es de 58.30µs/cm. Así pues, el rango mínimo que se puede medir es de 1.7 cm (100µS/58) y el máximo de 431 cm (25mS/58). Debe respetarse un tiempo mínimo de 50 ms entre dos pulsos de disparo consecutivos del mismo sensor.
El retardo entre el pulso de disparo y el inicio del pulso de ECO puede variar de un módulo SRF a otro.

2) Un PIC18F4520 utiliza las líneas RB0 y RB1 para recibir dos señales digitales de entrada, de las que debe estar continuamente calculando la anchura de los pulsos que llegan, y visualizar las anchuras de los últimos pulsos recibidos en una pantalla LCD16x2 (como la utilizada en prácticas). Ambas líneas las debe gestionar mediante interrupciones, procurando que el error de medida sea mínimo. La anchura máxima de los pulsos es de unos 50 ms.

Para minimizar el consumo de potencia, haremos que el PIC pase automáticamente a modo de reposo (sleep) cuando transcurra un tiempo sin detectar nuevos pulsos en las líneas de entrada. El sistema utiliza un reloj de 4MHz, y se quiere que pase automáticamente y se mantenga en bajo consumo cuando transcurran unos 5 segundos sin que le lleguen pulsos. La llegada de nuevos pulsos hará que el PIC pase de nuevo al modo de medida.

Redacta el programa para que el sistema funcione en la forma indicada. Se supone que la pantalla está conectada al puerto B del PIC, pero se comunica dejando libres las líneas RB0 y RB1.

3) Para implementar un sencillo control electrónico de estabilidad (ESP) de un automóvil, un PIC 18F4520 recibe en las entradas RB0 y RB1 dos señales de pulsos que codifican las revoluciones de las dos ruedas delanteras. Cuando la velocidad es de 100Km/hora, si la adherencia con el suelo es perfecta, el decodificador de cada rueda produce 10 pulsos/segundo, y este valor cambia de manera lineal con la velocidad, e igual para las dos ruedas. Sin embargo, si se produce un deslizamiento o una pérdida de adherencia temporal en alguna de las ruedas, el sistema EPS debe actuar corrigiéndolo.
Redacta un programa que monitorice continuamente el giro de ambas ruedas y actúe frenando aquella que se desvíe en revoluciones, cada vez que supere en más de un 10% a la contraria.

Observaciones:
- Se puede suponer que las señales que activan los frenos de la rueda izquierda y la derecha son respectivamente las salidas RA0 y RA1 del PIC. Cada freno actúa durante todo el tiempo que la correspondiente salida esté en alta, haciendo que disminuya progresivamente el número de revoluciones de la correspondiente rueda.
- La anchura de los pulsos no es relevante, sólo su cadencia. Conviene detectarlos en las líneas RB0 y RB1 mediante interrupciones; y utilizar un temporizador para medir los tiempos entre impulsos en cada línea. Suponiendo que el PIC emplea un cuarzo de 4MHz, y que usamos temporizadores de 16 bits para medir las frecuencias de impulsos, elige los pre-scalers adecuados para el sistema ESP desempeñe su función al menos en el rango de velocidades: 20 a 200Km/h