UNIVERSIDAD CARLOS III DE MADRID (Dpto. de Tecnología Electrónica)
Sist. Dig. Basados en Microprocesador **(Gr. Ing. Telemática)**
21 de junio de 2013 EXAMEN CONV. EXTRAORD. (3 horas)

EJERCICIO 1 (2.0 puntos):

Diseñe un microprocesador con las siguientes características:

- 16KB de memoria de datos.
- 8KB de memoria de programa.
- Juego de 128 instrucciones de 16 bits.
- Bus de datos de 8 bits.
- Minimice la velocidad de carga de la instrucción (Fetch).

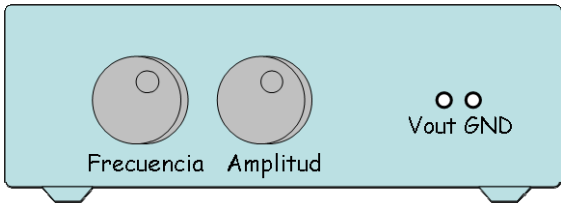
Se pide:

a) En caso de implementarse con arquitectura Von Neuman:
   a. Diagrama de bloques indicando el tamaño de los buses principales.
   b. Mapa de memoria.
   c. Tamaño del contador de programa y del registro de instrucción.

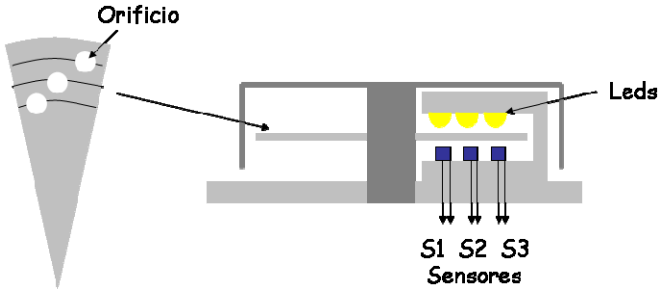
b) En caso de implementarse con arquitectura Harvard:
   a. Diagrama de bloques indicando el tamaño de los buses principales.
   b. Mapas de memoria.
   c. Tamaño del contador de programa y del registro de instrucción.

EJERCICIO 2 (4.0 puntos):

El generador de onda de la siguiente imagen produce una señal de triangular de frecuencia y amplitud variable construida con 10 muestras por periodo.



La amplitud de la onda varía entre 0 y 3V, y es controlada mediante una rueda de giro "sin fin" cuyo mecanismo se puede ver en la siguiente imagen.



La rueda dispone de un orificio cada 10º, es decir, dispone de un total de 36 orificios. Dichos orificios están situados a tres radios distintos, de forma que a su paso por los leds se excita únicamente el sensor asociado al mismo. Esto quiere decir que cuando la rueda gira en sentido horario los sensores ópticos se excitan siguiendo la siguiente secuencia S3-> S2 ->S1, mientras que cuando gira en sentido antihorario la secuencia de excitación es la siguiente S1 -> S2 ->S3.
El sentido horario implicará un incremento de 10mV en la amplitud de la tensión de salida por cada 10º girados, mientras que el giro en sentido antihorario supondrá -10mV por cada 10º de giro.

La rueda que controla la frecuencia, es una rueda asociada a un potenciómetro y regula la frecuencia entre 1KHz y 10KHz cuando la señal ofrecida por el potenciómetro vale 0 V y 3 V respectivamente.

Como parámetros de desarrollo se han marcado los siguientes:

1. La recogida de información de la rueda de control de frecuencia se gestionará con la interrupción del ADC. En esta interrupción, la variable global "tiempo_muestra" se actualizará con el tiempo que debe transcurrir entre dos muestras consecutivas emitidas por el dispositivo para una determinada frecuencia.
2. La recogida de información de la rueda de control de amplitud se realizará con interrupciones externas (EXT_INT). Utilizando la variable global "estado" que controlará el estado actual (S1, S2 y S3).
3. El muestreo de la señal de salida se realizará mediante la interrupción de un TOC que activará el flag "flagSacarDato".
4. El flujo principal del programa (main), usará las variables "flagSacarDato", "estado" y "amplitud" para generar la onda con el periférico correspondiente (gestionando también el control de la amplitud de la misma).

Se pide:

a) Diagrama de bloques.
b) Diagrama de flujo.
c) Código de configuración de los pines del micro (únicamente los GPIO, sin las EXTI, ni el NVIC).
d) Código de la RAI asociada a alguna de las interrupciones externas.

EJERCICIO 3 (4.0 puntos):

En las próximas hojas aparece el código de un programa a cargar en un sistema basado en un STM32L152RB, **al cual le puede faltar alguna función y/o alguna definición y/o contener algún error de programación**. El programa corresponde a la implementación de un velocímetro de bicicleta, capaz de mostrar por la pantalla la información sobre la distancia recorrida, la velocidad instantánea y la temperatura. De dicho sistema se conocen las siguientes características:

- Tiene dispositivos conectados cuya señal de entrada al STM32L152RB, es un valor cualquiera entre 0 y 3,3V, con una variación de velocidad menor de 1Hz.
- Puede tener algún otro dispositivo más conectado.

Se pretende que el alumno analice dicho código y, a partir de ahí conteste **razonadamente** a las siguientes preguntas (*algunas de las justificaciones se pueden realizar indicando las líneas de código donde se encuentra la evidencia*):

1. ¿Qué elementos (periféricos) del STM32L152RB se están utilizando? (10%)
2. Teniendo en cuenta que tiene una configuración de reloj que hace que el pclk de todos los periféricos vaya a 12MHz, indique la configuración de cada uno de los periféricos del microcontrolador cuando se encuentren en funcionamiento. Absténgase de simplemente decir el valor de cada registro de configuración; lo que se pide es la funcionalidad que se obtiene. Es imprescindible detallar la escala temporal (caso de que exista) que utilizan los periféricos. (10%)
3. De un significado breve a las siguientes variables del programa. No tienen por qué ser acrónimos de ningún tipo (20%):
    a. cero
    b. uno
    c. dos
    d. tres
    e. cuatro
    f. cinco
    g. seis
4. El código presenta al menos tres errores. Dos de esos errores está en los nombres de las funciones, mientras que el otro se encuentra en el funcionamiento del mismo. Encuéntrelos, justifíquelos y dé una solución a los mismos. (20%)
5. Analizando la pureza del código, un experto expone una crítica a la línea 35. ¿Podría decir qué crítica es, y cómo la solucionaría? (10%)
6. Realice el Diagrama de flujo de todas y cada una de las funciones utilizadas, así como del programa principal. No haga siempre una transposición directa del código en un diagrama de flujo, sino represente la funcionalidad obtenida, mediante dicho diagrama de flujo. (30%)

ANEXO I

```
1   #include "stm32l1xx.h"
2   #include "Biblioteca_SDM.h"
3   unsigned char radio=23;
4   unsigned uno, dos;
5   unsigned char cero;
6   unsigned tres;
7   unsigned cuatro;
8   unsigned cinco;
9   unsigned seis;
10
11  void RAI1 (void) {
12     EXTI->PR = 0x01;
13     NVIC->ICER[0] |= (1 << 6);
14     if ((EXTI->RTSR & 0x01) == 0) {
15        TIM4->CCR2 = TIM4->CNT + 2000;
16        TIM4->CR1 |= 0x0001;
17        EXTI->RTSR |= 0x01;
18        EXTI->FTSR &= ~(0x01);
19        }
20     else {
21        TIM4->CR1 &= ~(0x0001);
22        EXTI->FTSR |= 0x01;
23        EXTI->RTSR &= ~(0x01);
24        }
25     NVIC->ISER[0] |= (1 << 6);
26     }
27
28  void RAI2 (void) {
29     if (TIM4->SR & 0x0004 == 1) {
30        cero = 1;
31        TIM4->CR1 &= ~(0x0001);
32        TIM4->SR = 0x0004;
33        }
34     else {
35        seis = 2 * 3142 * radio;
36        tres += seis;
37        dos = TIM4->CCR1 - uno;
38        if (dos < 0) dos += 0xFFFFFFFF;
39        cinco = seis / dos;
40        uno = TIM4->CCR1;
41        TIM4->SR = 0x0002;
42        }
43     }
44
45  int main (void) {
46     Init_SDM();
47     Init_LCD();
48     cero = 0;
49     GPIOA->MODER |= 0x00000300;
50     GPIOA->MODER &= ~(1 << (0*2 +1));
51     GPIOA->MODER &= ~(1 << (0*2));
52     GPIOA->PUPDR &= ~(11 << (0*2));
53     GPIOB->MODER |= 0x00000001 << (2*6 +1);
```

```
54      GPIOB->MODER &= ~(0x00000001 << (2*6));
55      GPIOB->AFR[0] &= ~(0x0F << (4*6));
56      GPIOB->AFR[0] |= 0x02 << (4*6);
57      ADC1->CR2 &= ~(0x00000001);
58      ADC1->CR1 = 0x02000000;
59      ADC1->CR2 = 0x00000472;
60      ADC1->SMPR1 = 0;
61      ADC1->SMPR2 = 0;
62      ADC1->SMPR3 = 0;
63      ADC1->SQR1 = 0x00000000;
64      ADC1->SQR5 = 0x00000004;
65      ADC1->CR2 |= 0x00000001;
66      while ((ADC1->SR&0x0040)==0);
67      ADC1->CR2 |= 0x40000000;
68      EXTI->FTSR |= 0x01;
69      EXTI->RTSR &= ~(0x01);
70      SYSCFG->EXTICR[0] = 0;
71      EXTI->IMR |= 0x01;
72      NVIC->ISER[0] |= (1 << 6);
73      TIM4->CR1 = 0x0000;
74      TIM4->CR2 = 0x0000;
75      TIM4->SMCR = 0x0000;
76      TIM4->PSC = 12000;
77      TIM4->CNT = 0;
78      TIM4->ARR = 0xFFFF;
79      TIM4->CCR2 = 0;
80      TIM4->DCR = 0;
81      TIM4->DIER = 0x0006;
82      TIM4->CCMR1 = 0x0001;
83      TIM4->CCMR2 = 0x0000;
84      TIM4->CCER = 0x0003;
85      TIM4->CR1 |= 0x0001;
86      TIM4->EGR |= 0x0001;
87      TIM4->SR = 0;
88      NVIC->ISER[0] |= (1 << 30);
89      uno = 0;
90      while (1) {
91         if (cero!=0) {
92            uno = 0;
93            tres = 0;
94            }
95         while ((ADC1->SR & 0x0002)==0);
96         cuatro = (unsigned char)(ADC1->DR & 0x000000FF);
97         MuestraVelocimetro(cinco, tres, cuatro);
98         }
99      }
```