**Cuestión (20 minutos – 2 puntos)**

Se tiene una arquitectura Harvard, con capacidad de 32Mpalabras de programa y 16MB de datos en palabras de 16 bits, con una capacidad de direccionar únicamente palabras. Dicha arquitectura contiene 16 registros internos propósito general y filosofía Load & Store. Con esta información, conteste a las siguientes preguntas:

1) (70%) Diseñe una codificación de los distintos tipos de instrucciones microprocesador, minimizando en lo posible el tamaño de la instrucción ajustando a números enteros de palabras. Los tipos de instrucciones que debe tener el microprocesador, son:

- 7 instrucciones de transferencia de datos entre memoria y registros internos, con direccionamiento directo.
- 5 instrucciones de transferencia de datos entre memoria y registros internos, con direccionamiento indexado.
- 14 instrucciones aritmético/lógicas de operar entre registros, siendo el resultado el mismo registro que uno de los operandos.
- 14 instrucciones aritmético/lógicas de operar entre registros, con uno de los operandos dado por direccionamiento inmediato, y dando el resultado en el otro operando.
- 6 instrucciones de control con direccionamiento inherente.
- 7 saltos condicionales con direccionamiento relativo a contador de programa, siendo el desplazamiento relativo de más/menos 1Mpalabra.

2) (30%) Indique una respuesta justificada a cada una de las siguientes preguntas:

a) Tipos de buses internos, y tamaño de cada uno de ellos
b) Tamaño del Registro de Instrucción
c) Tamaño del Contador de Programa
d) Tamaño de los Registros internos

**Problema 1 (60 minutos – 4 puntos)**

Para evitar la aglomeración de vehículos en un aparcamiento público, se va a desarrollar un sistema básico de control de acceso basado en un microcontrolador que ha utilizado durante el curso. A continuación se muestra el esquema del sistema y se describen las funcionalidades requeridas.

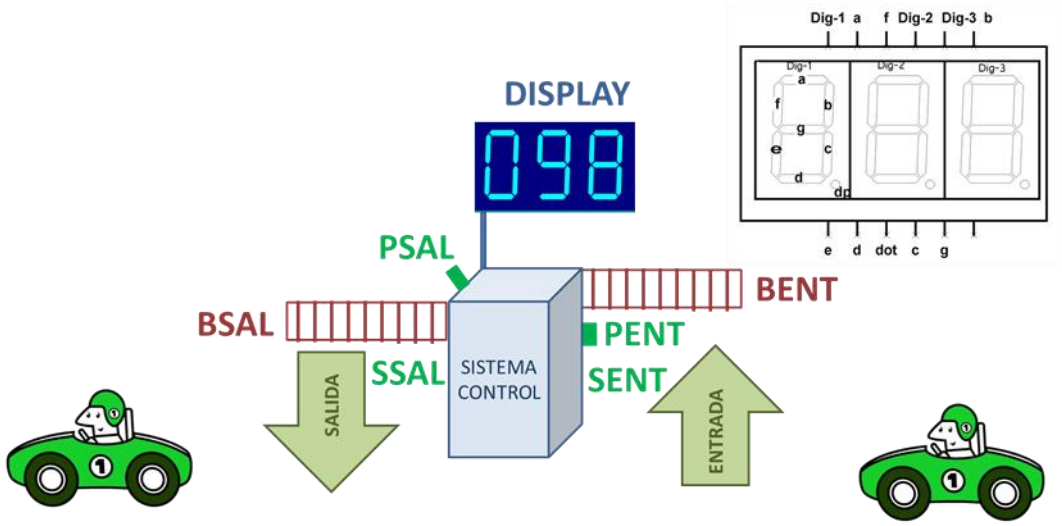


Figura 1. Esquema del sistema de control de acceso al aparcamiento y detalle de los pines del display

Cuando el sistema empieza a funcionar el aparcamiento está vacío y supondremos que cada vehículo que entra ocupa una única plaza de aparcamiento.

En la entrada, el sistema dispone de un display de siete segmentos de cátodo común de 3 dígitos (DISPLAY), en el que se representa el número de plazas libres en el aparcamiento en cada instante.

En la entrada, también hay un pulsador (PENT) para activar la elevación de la barrera de entrada (BENT). Cuando se pulsa PENT se debe levantar la barrera de entrada durante un tiempo, TBENT (supondremos que en ese tiempo entra un solo vehículo), en el caso de que haya plazas libres en el aparcamiento. Si algún conductor pulsa el botón de entrada con el aparcamiento completo la barrera permanecerá bajada y el conductor tendrá que volver a pulsar PENT cuando haya alguna plaza libre para que la barrera se eleve.

En la salida hay un pulsador de salida (PSAL). Cuando se pulsa, se eleva la barrera de salida (BSAL) durante un tiempo, TBSAL, en el cuál supondremos que sale un solo vehículo.

Ambas barreras se elevan cuando se pone un 1 lógico en el punto de conexión de la barrera al sistema de control (cuando en este punto de conexión hay un 0 lógico la barrera permanece bajada).

El sistema dispone también de dos sensores (uno en la entrada, SENT, y otro a la salida, SSAL) que permiten detectar si hay un obstáculo para mantener la barrera correspondiente elevada si el coche no la ha atravesado en el tiempo dado. La tensión de salida de estos sensores puede variar entre 0 y 5V. Una tensión de salida del sensor mayor que 1,3V indica que hay un obstáculo y la barrera correspondiente no debe cerrarse hasta que no desaparezca dicho obstáculo.

El código desarrollado para la aplicación es el que se muestra al final de este enunciado.

**DATOS: Se configura el oscilador del microcontrolador con $F_{osc}$ = 8MHz. El micro está alimentado entre 0 y 5V**

**Se pide:**

1. Represente el esquema del hardware para esta aplicación. Dibuje el microcontrolador como un bloque, indicando claramente qué pines del microcontrolador se conectan a cada una de las entradas o salidas de todos los elementos que componen el sistema (cada elemento debe estar claramente identificado en el diagrama con el mismo nombre que se les ha asignado en la figura 1). Además, rellene la siguiente tabla. (20%)

| PIN MICRO | HW CONECTADO | TIPO (E/S, DIGITAL/ ANALOGICO) | JUSTIFICACIÓN |
|---|---|---|---|
| | | | |

2. Represente el diagrama de flujo del programa principal y de la/s rutina/s de atención a la interrupción, incluyendo una descripción de lo que hace cada función que aparece en el código. (30%)

3. ¿Con qué recurso del microcontrolador se controla el tiempo de elevación de la barrera de salida y durante cuánto tiempo se mantiene elevada la barrera de salida (TBSAL) en el caso de que el sensor de salida no detecte ningún obstáculo? Justifique su respuesta incluyendo todos los cálculos necesarios para obtenerla. (15%)

4. ¿Cuál es número total de plazas que tiene disponible el aparcamiento según el código y cuál sería el máximo número de plazas que se podrían gestionar con el código suministrado? Razone su respuesta (10%)

5. Indique la configuración del ADC utilizada y cuál es el tiempo de conversión y el tiempo entre conversiones. Indique la modificación necesaria para que las conversiones se hagan más frecuentemente. (15%)

6. En el código falta una línea de código en el lugar señalado en el mismo. Escriba a continuación la línea de código que falta. Justifique su respuesta. (10%)

## CÓDIGO PROGRAMA CONTROL APARCAMIENTO

```
#include "Biblioteca_SDM.h"
#include "stm32l1xx.h"

unsigned int tiempo_BENT_up = 0;
unsigned int tiempo_BSAL_up = 0;
unsigned char barrera_entrada_subida = 0;
unsigned char barrera_salida_subida = 0;
unsigned char obstaculo_ENT = 0;
unsigned char obstaculo_SAL = 0;
unsigned char num_vehiculos=0;

void setup(void);
void representar(void); // ESTA FUNCIÓN REPRESENTA EN EL DISPLAY EL Nº DE VEHÍCULOS EN
                        // EL APARCAMIENTO (NO SE INCLUYE SU CÓDIGO)

void main(void){
   setup();
   while(1){
      representar();
      if ((EXTI->PR & (1<<0)) != 0) {
         EXTI->PR |= 1<<0;
         if ((obstaculo_ENT == 0) && (barrera_entrada_subida == 0) && (num_vehiculos < 152)){
            barrera_entrada_subida = 1;
            GPIOB->BSRR |= 1<<3;
            tiempo_BENT_up = 0;
            num_vehiculos++;
            }
         }
      if ((EXTI->PR & (1<<1)) != 0) {
         EXTI->PR |= 1<<1;
         if ((obstaculo_SAL == 0) && (barrera_salida_subida == 0)){
            barrera_salida_subida = 1;
            GPIOB->BSRR |= 1<<2;
            tiempo_BSAL_up = 0;
            num_vehiculos--;
            }
         }
      if ((barrera_entrada_subida == 1)  && (obstaculo_ENT ==0) && (tiempo_BENT_up > 20)){
         barrera_entrada_subida = 0;
         GPIOB->BSRR |= (1<<3)<<16;
         }
      if ((barrera_salida_subida == 1) && (obstaculo_SAL==0) && (tiempo_BSAL_up > 20)){
         barrera_salida_subida = 0;
         GPIOB->BSRR |= (1<<2)<<16;
         }
      }
   }

void setup(void){
   GPIOA->MODER |= 0x03 << (1*2);
   GPIOA->MODER |= 0x03 << (5*2);
   GPIOA->MODER &= ~(0xFFFFF000);
   GPIOA->MODER |= 0x55555000;
   GPIOB->MODER &= 0xFFFFFF00;
   GPIOB->MODER |= 0x05 << (2*2);
   SYSCFG->EXTICR[0] = 0x0011;
   EXTI->IMR |= 0x03;
   EXTI->RTSR |= 0x03;
   EXTI->FTSR &= ~(0x03);
   ADC->CR1 = 0x02000020;
   ADC->CR2 = 0x00000401;
   ADC->SQR1 = 0x00100000;
   ADC->SQR5 = 0x01;
   TIM3->CR2 = 0;
   TIM3->SMCR = 0;
   TIM3->CCMR1 = 0x0000;
   TIM3->CCER = 0;
   TIM3->CNT = 0;
   TIM3->PSC = 7999;
   TIM3->ARR = 0xFFFF;
   TIM3->CCR1 = 1000;
```

```
   TIM3->DIER = 0x0002;
   TIM3->CR1 = 0x0001;
   NVIC->ISER[0] |= 1 << 18;
   NVIC->ISER[0] |= 1 << 29;
   }

void TIM3_IRQHandler(void) {
   TIM3->SR &= ~(0x1 << 1);
   tiempo_BENT_up++;
   tiempo_BSAL_up++;
   TIM3->CCR1 = 1000;
   ADC->CR2 |= 1<<30;
   }

void ADC1_IRQHandler(void) {
   [redacted]
   if ((ADC->SQR5 & 0x001F)==1) {
      ADC->SQR5 = 0x05;
      if (ADC->DR > 67) obstaculo_ENT = 1;
      else obstaculo_ENT = 0;
      }
   else if ((ADC->SQR5 & 0x001F)==5) {
      ADC->SQR5 = 0x01;
      if (ADC->DR > 67) obstaculo_SAL = 1;
      else obstaculo_SAL = 0;
      }
   }
```

**Problema 2 (100 minutos – 4 puntos)**

Basándose en el microcontrolador que ha utilizado durante el curso con una frecuencia de reloj de 8MHz, se necesita que implemente un subsistema de control del suministro eléctrico de un sistema mayor, de tal forma que genere alarmas cuando dicho suministro no se ajuste a los requisitos del sistema. Para cumplir con esta función, al microcontrolador se le conectará un dispositivo externo que tiene como entrada la red eléctrica de suministro y como salida una señal cuadrada, $V_{cuadrada}$, del mismo periodo y ciclo de trabajo, *duty-cycle* (DC), que la señal de la red eléctrica, pero entre Vcc y masa (en condiciones normales, la frecuencia de la señal de la red eléctrica es de 50Hz con DC del 50%). El sistema admite frecuencias entre 40Hz y 60Hz y DC entre el 45% y el 55%

Partiendo de esta señal de entrada, el subsistema de control tiene que suministrar al mundo exterior la información a través de 2 LEDs (uno verde LV y uno rojo LR), un altavoz (al que le atacará una señal cuadrada de uno tono determinado) y una comunicación serie con baud rate de 19200, sin paridad, con 8 bits de datos y 1 bit de parada (19200,8,N,1). La información a ofrecer debe ser (por orden prioritario):

- Si no hay alarma: LV encendido, LR apagado, sin tono, sin comunicación serie
- Si frecuencia fuera de rango: LV apagado, LR encendido, tono de 1kHz, mensaje "HHMMSS_FR"
- Si DC fuera de rango: LV apagado, LR encendido, tono de 500Hz, mensaje "HHMMSS_DC"

Los mensajes sólo se envían una vez al empezar a ocurrir cada alarma. En dichos mensajes HH es la hora, MM es el minuto y SS es el segundo, cada uno de ellos tiene dos dígitos y por tanto se envía con dos caracteres ASCII. Para obtener dicha información, el subsistema cuenta con un Real Time Clock (RTC) al que se puede consultar la información a través de la siguiente función:

```
void ObtenerHora (unsigned char *hora, unsigned char *minuto, unsigned char *segundo);
```

Con estas especificaciones, y teniendo en cuenta que se valorará el nivel de optimización de la solución (es decir, menor uso de periféricos y menor tiempo de computación), conteste a las siguientes preguntas:

1. Enumere las tareas que realiza el microprocesador y los periféricos que utilizará para realizar cada una de ellas. Realice también el diagrama de bloques detallado del sistema, incluyendo todos los elementos del sistema mencionados en el enunciado y los pines de conexión del microcontrolador con cada uno de ellos (25%)

2. Describa como configuraría los periféricos utilizados para realizar la comunicación serie y generar la señal cuadrada que atacará al altavoz. Escriba el valor de los registros de configuración de estos periféricos. Justifique los valores utilizados (25%)

3. Indique los periféricos que se atenderán por interrupciones. Justifique su elección. Escriba las líneas de configuración necesarias para habilitar las fuentes de interrupción seleccionadas. (20%)

4. Diagrama de flujo del programa principal (utilice funciones para realizar las acciones necesarias en caso de que la frecuencia medida esté fuera de rango o el DC medido esté fuera de rango), diagramas de flujo de las funciones utilizadas en el principal y diagramas de flujo de la/s rutina/s de atención a la interrupción (30%)