Cuestión (20 minutos – 2 puntos)

Un microcontrolador de la familia STM32L152RB debe disponer de una memoria de datos de 256M x 16 bits. En esta memoria, la parte más baja es memoria ROM y la parte más alta es memoria RAM. También se sabe que hay 64M de memoria ROM y 192M de memoria RAM.

Para implementar esta memoria se dispone de los siguientes chips:

- Chips ROM de 32M x 16 bits
- Chips RAM de 64M x 8 bits
- Chips RAM de 128M x 16 bits

a) Indica cuántos chips de cada tipo hacen falta para implementar la memoria.
b) Dibuja el mapa de memoria del sistema, indicando la primera y última dirección de cada zona en hexadecimal. El diseño del microcontrolador obliga a que el chip de RAM más grande tenga que ir colocado en la parte más alta de la memoria.
c) Diseña el decodificador de direcciones válido para esta memoria utilizando un decodificador 2:4 con salidas activas por nivel alto y las puertas lógicas AND, OR y NOT que consideres necesarias. Considera que las señales de Chip Select (CS) con activas por nivel bajo.

SOLUCIÓN:

a) Mirando las necesidades de direccionamiento de cada zona de la memoria y el tamaño de cada chip, el número de chips de cada tipo sería el siguiente, colocados exactamente así de abajo a arriba:

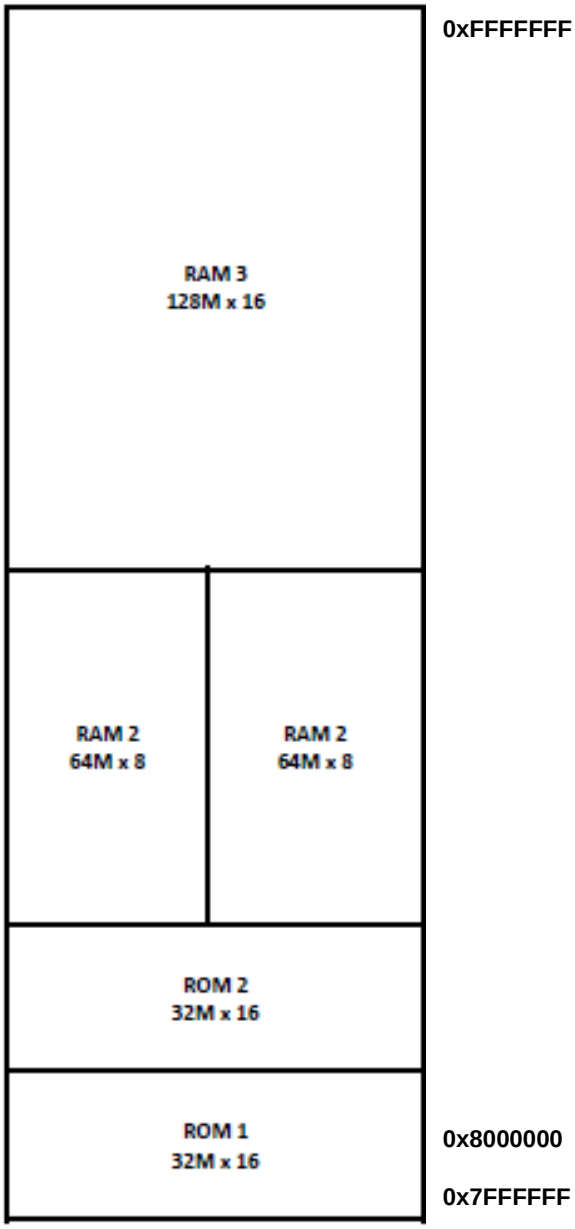
- *Chips ROM de 32M x 16 bits -> 2*
- *Chips RAM de 64M x 8 bits -> 2*
- *Chips RAM de 128M x 16 bits ->1*

b) Para el mapa de memoria, se tiene que la memoria ROM va debajo, la memoria RAM va arriba y el chip más grande de memoria RAM debe ir en la parta más alta de la memoria, por lo que la única solución es colocar los 2 chips de ROM en la parta más baja de la memoria uno encima del otro para alcanzar los 64M x 16 necesarios, los 2 chips de RAM pequeños uno al lado del otro para obtener 64M x 16 y el chip de RAM más grande encima del todo para tener 128M x 16, que sumados a los anteriores dan el total de 192M x 16 necesarios.

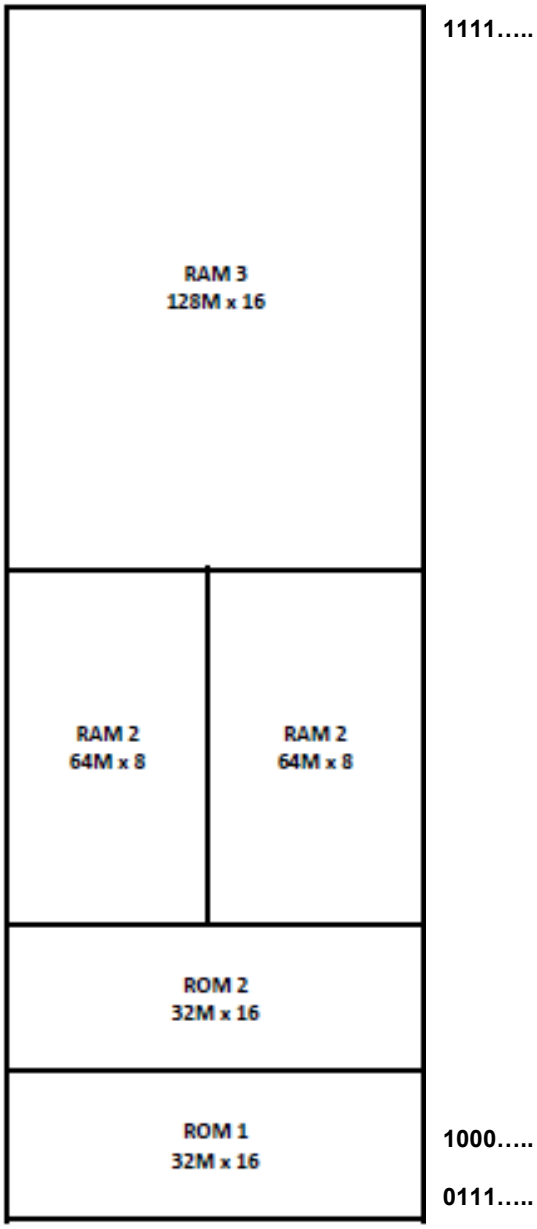
Para el cálculo de las direcciones hexadecimales de cada zona, como hay 256M de direcciones en total, hacen falta 28 bits de direcciones (2^{28} = 256M), del A0 al A27, por tanto las direcciones de memoria en hexadecimal van de la 0x0000000 (fila 1) hasta la 0xFFFFFFF (fila 256M).

- *El primer chip de memoria ROM empieza en la fila 1 (0x0000000) y llega a la fila 32M (2^{25} = 0x1FFFFFF)*
- *El segundo chip de memoria ROM empieza en la fila 32M+1 (0x2000000) y llega a la fila 64M (2^{26} = 0x3FFFFFF = 0x2000000 + 0x1FFFFFF)*
- *Los dos chips más pequeños de memoria RAM tienen el mismo direccionamiento y empiezan en la fila 64M+1 (0x4000000) y llega a la fila 128M (2^{27} = 0x7FFFFFF = 0x4000000 + 0x3FFFFFF)*
- *El chip más grande de memoria RAM empieza en la fila 128M+1 (0x8000000) y llega a la fila 256M (2^{28} = 0xFFFFFFF = 0x8000000 + 0x7FFFFFF)*

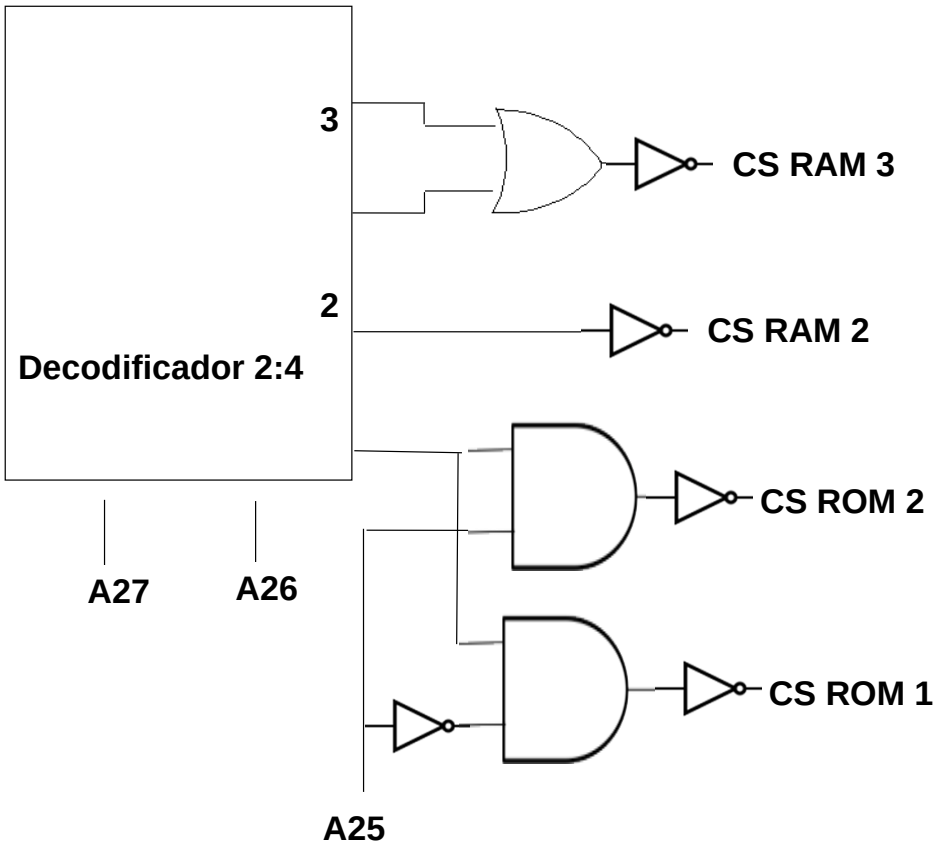
El mapa de memoria final es el siguiente:



c) Para el decodificador de direcciones, lo mejor es pasar el mapa de memoria a binario y fijarnos en los 3 últimos bits de las direcciones iniciales y finales de cada zona, que son las que nos van a diferenciar cada uno de los chips.



Tal como se ve, si colocamos los bits más significativos del bus de direcciones (A27 y A26) como entradas del decodificador 2:4, podemos diferenciar perfectamente la zona RAM3, la zona RAM2 y las zonas ROM (usando las puertas lógicas necesarias para obtener los niveles altos o bajos necesarios de CS), pero para diferenciar entre ambas zonas ROM hace falta incluir el bit A25 y las puertas lógicas necesarias. Considerando todo esto y teniendo en cuenta que el decodificador 2:4 tiene salidas activas por nivel alto y las señales de Chip Select (CS) con activas por nivel bajo, la solución para el decodificador de direcciones es la siguiente usando puertas AND, OR y NOT adicionales necesarias:


3
2
Decodificador 2:4
A27
A26
A25
CS RAM 3
CS RAM 2
CS ROM 2
CS ROM 1

Problema 1 (60 minutos – 4 puntos)

Se desea instalar en una señal de tráfico luminosa un sistema que controle el brillo de sus LEDs de manera que, cuando la intensidad de luz ambiente sea mayor el brillo de la señal de tráfico también sea mayor y, cuando la luz ambiente baje (por ejemplo, de noche), el brillo de la señal sea menor para no deslumbrar.

Para implementar el prototipo se va a utilizar la placa de desarrollo con el microcontrolador de referencia del curso, una placa LightHz de Mikroelectronika que incorpora un integrado TSL230BR y una placa basada en el integrado A6260. La placa LightHz con el integrado TSL230BR genera una señal digital cuya frecuencia es proporcional a la cantidad de luz ambiente que recibe. El integrado A6260 se utiliza para entregar a los LEDs de la señal de tráfico la corriente necesaria, conectando su señal de entrada EN a una señal PWM, la corriente suministrada a los LEDs es proporcional al ciclo de trabajo de la misma. Al final de este enunciado se da un código de referencia, en base al cual ha de responder a las siguientes preguntas:

1. Represente el esquema del hardware utilizado para la aplicación, indicando claramente qué pines del microcontrolador se conectan a los elementos que componen el sistema (20%)

2.- Identifique qué timers se utilizan en este programa. Indique para cada uno de ellos si funciona como TOC, TIC o PWM y cuál es su estado (en marcha o parado) tras la rutina de configuración. (10%)

3.- Represente el diagrama de flujo de la función declarada como "`void espera (unsigned int segundos);`". ¿Para qué se utiliza esta función en el programa principal? (10%)

4.- En este programa ¿se utilizan variables globales? En caso afirmativo diga cuales. Justifique la necesidad de utilizar variables globales. (5%)

5.- El programa configura un pin de salida. Identifique cual es este pin y dibuje la forma de onda que genera el microcontrolador en este pin durante los primeros instantes tras el Setup(). (15%)

6.- El integrado TSL230BR es un sensor de luz con salida en frecuencia. ¿Cuál es la frecuencia máxima que puede medirse en base al programa? (20%)

7.- Suponiendo que la luz ambiente es tal que la frecuencia de la señal digital generada por el sensor de luz es de 100kHz, indique cuál sería el ciclo de trabajo de la señal PWM que se aplica al integrado A6260. (20%)

CÓDIGO PROGRAMA CONTROL BRILLO LEDs

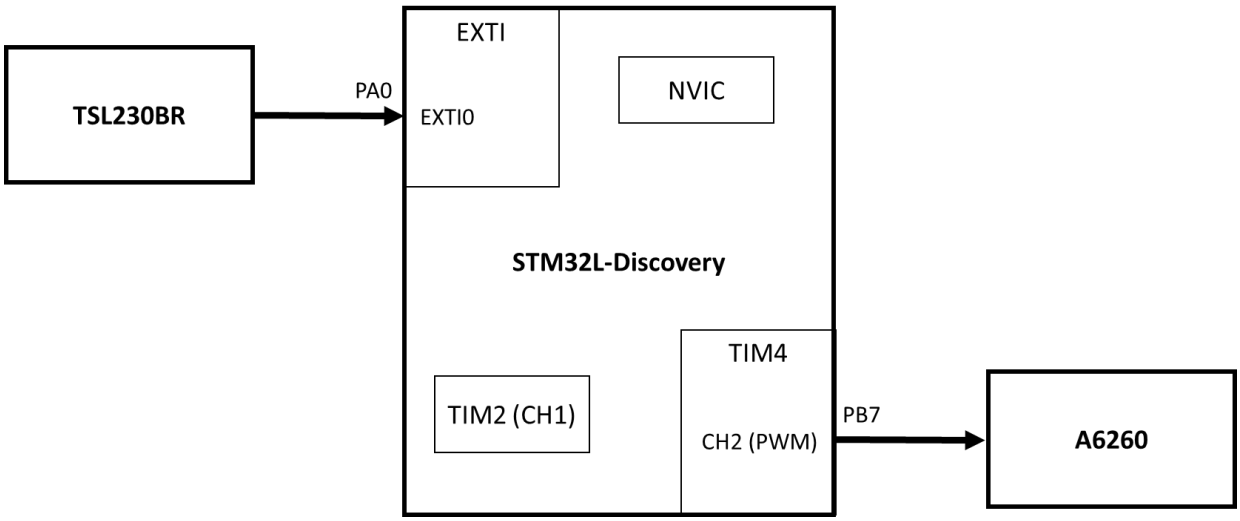
```
#include "Biblioteca_SDM.h"
#include "stm32l1xx.h"

#define TMR0_1segundo 1000

void Setup(void);
void espera (unsigned int segundos);
unsigned cuenta = 0;

void EXTI0_IRQHandler(void){
      if (EXTI->PR!=0){
      cuenta++;
      EXTI->PR = 0x01;
      }
  }

void main(void){
   Init_SDM();
   Setup();
   while(1) {
      espera(1000);
      cuenta = 0;
      EXTI->PR = 0x01;
      EXTI->IMR |= 0x01;
      espera(2);
      EXTI->IMR &= ~(0x01);
      TIM4->CCR2 = cuenta/8;
   }
}

void Setup(void){
   GPIOB->MODER|=0x00000001 << (2*7 +1);
   GPIOB->MODER&=~(0x00000001 << (2*7));
   GPIOB->AFR[0]|=(0x02 << (7*4));

   TIM2->CR1 = 0x0000;
   TIM2->CR2 = 0x0000;
   TIM2->SMCR = 0x0000;
   TIM2->PSC = 32000;
   TIM2->CNT = 0;
   TIM2->ARR = 0xFFFF;
   TIM2->CCR1 = TMR0_1segundo;
   TIM2->DIER = 0x0000;
   TIM2->CCMR1 = 0x0000;
   TIM2->CCER = 0x0000;
   TIM2->EGR |= 0x0001;

   GPIOA->MODER &= ~(1 << (0*2 +1));
   GPIOA->MODER &= ~(1 << (0*2));
   EXTI->FTSR |= 0x01;
   EXTI->RTSR &= ~(0x01);
   SYSCFG->EXTICR[0] = 0;
   NVIC->ISER[0] |= (1 << 6);

   TIM4->CR1 = 0x0080;
   TIM4->CR2 = 0x0000;
   TIM4->SMCR = 0x0000;
   TIM4->PSC = 0;
   TIM4->CNT = 0;
   TIM4->ARR = 32000;
   TIM4->DIER = 0x0000;
   TIM4->CCMR1 = 0x6800;
   TIM4->CCER = 0x0010;
   TIM4->CCR2 = 16000;
   TIM4->CR1 |= 0x0001;
   TIM4->EGR |= 0x0001;
   TIM4->SR = 0;
}
```

```
void espera (unsigned int segundos){
   unsigned int t = 0;
   TIM2->SR = 0;
   TIM2->CR1 |= 0x0001;
   while (t < segundos){
      TIM2->CCR1 = TIM2->CNT + TMR0_1segundo;
      while ((TIM2->SR&0x0002)==0);
      TIM2->SR &= ~(0x0002);
      t++;
   }
   TIM2->CR1 &= ~(0x0001);
}
```

SOLUCIÓN:

1. Represente el esquema del hardware utilizado para la aplicación, indicando claramente qué pines del microcontrolador se conectan a los elementos que componen el sistema (20%)

Atendiendo al enunciado y al código se aprecia que hay un dispositivo de entrada a nuestro sistema, que es el TSL230BR que nos ofrece la señal variable en frecuencia, y tenemos que generar una señal PWM que ataque al A6260 que proporcione la corriente a la señal de tráfico.
Con los datos de la función Setup() se puede apreciar que la entrada ataque a la EXTI0 a través del PA0 y que la salida PWM sale utilizando el TIM4 con el canal 2, saliendo por el PB7.
Adicionalmente tenemos otro temporizador para medir segundos, sin salida externa, y el controlador de interrupciones para la EXTI.



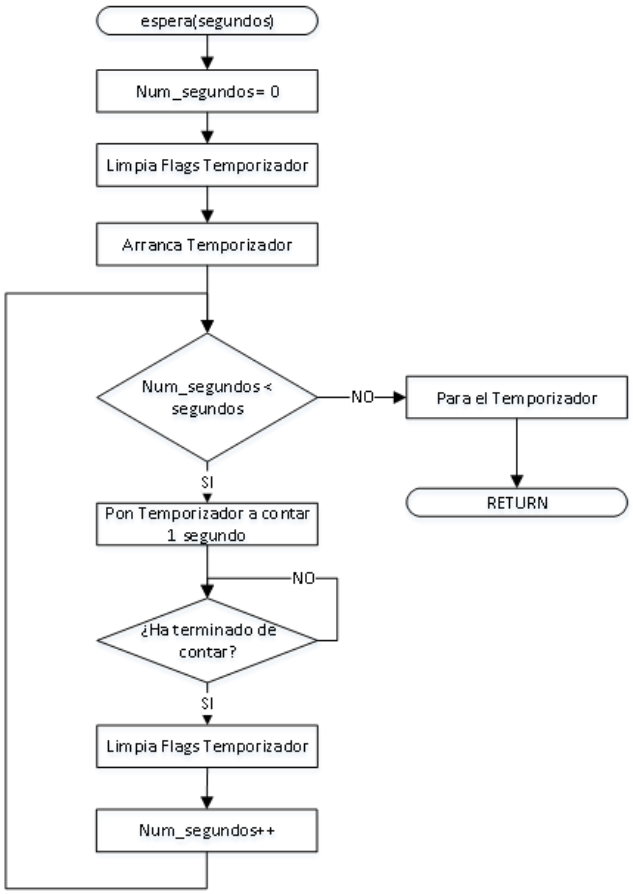
2.- Identifique qué timers se utilizan en este programa. Indique para cada uno de ellos si funciona como TOC, TIC o PWM y cuál es su estado (en marcha o parado) tras la rutina de configuración. (10%)

Se utilizan 2 timers. El TIM2 se utiliza como TOC, sin salida externa. Tras la configuración se encuentra parado.
El TIM4 se utiliza como PWM, sin pre-escalado, con periodo 32000 y semiciclo inicial de 16000. Tras la configuración se encuentra arrancado.

3.- Represente el diagrama de flujo de la función declarada como "void espera (unsigned int segundos);". ¿Para qué se utiliza esta función en el programa principal? (10%)

La función se utiliza para realizar una espera del número de segundos indicado en el parámetro, bloqueando el resto del programa, salvo lo gestionado por interrupciones.

Se utiliza para hacer una espera inicial, y luego para establecer un periodo determinado (2 segundos) de medida de flancos de bajada de la señal de entrada, para a partir de ahí, medir la frecuencia de la señal de entrada.



4.- En este programa ¿se utilizan variables globales? En caso afirmativo diga cuales. Justifique la necesidad de utilizar variables globales. (5%)

Sí. Se usa la variable "cuenta", que servirá para determinar el número de flancos que han ocurrido en la señal de entrada.
Es necesario, ya que es un dato que tiene que ser compartido entre el programa principal y la RAI, por lo que es un dato que nunca se puede pasar por parámetro.

5.- El programa configura un pin de salida. Identifique cual es este pin y dibuje la forma de onda que genera el microcontrolador en este pin durante los primeros instantes tras el Setup(). (15%)

El pin de salida es el PB7, tal y como se puede ver en las tres primeras líneas de código de la función Setup().
Dicha salida está asociada con la salida hardware del TIM4, que sacará una señal PWM.
Tras la ejecución del Setup(), la señal resultante es una señal cuadrada del 50% de duty cycle (ya que CCR1 es la mitad que ARR).
El periodo de la señal es, teniendo en cuenta un pclk de 32MHz, de:
32.000.000 / 32.000 = 1.000 Hz.
Por tanto el periodo es de 1ms.

6.- El integrado TSL230BR es un sensor de luz con salida en frecuencia. ¿Cuál es la frecuencia máxima que puede medirse en base al programa? (20%)

La frecuencia máxima tendrá que ver con el valor máximo que puede llegar a valer la variable "cuenta" durante los 2 segundos que se cuentan los flancos de EXTI0. A eso hay que añadirle la ejecución de la RAI, por lo que sólo se podrían detectar flancos que no caigan mientras se está ejecutando la RAI.
Como "cuenta" es un unsigned int, su límite de ciclos en 2 segundos es 2^{32}-1
Esta limitación llevaría a una frecuencia teórica máxima de 4,3GHz aproximadamente.
Pero la RAI tiene tres instrucciones en C. Suponiendo que cada instrucción en C ocupa como media unas 3 instrucciones en ensamblador, quiere decir que la ejecución de la RAI sería de 9 ciclos de reloj, por lo que la frecuencia máxima a medir sería la de pclk/9 = 3,6MHz aproximadamente.

7.- Suponiendo que la luz ambiente es tal que la frecuencia de la señal digital generada por el sensor de luz es de 100kHz, indique cuál sería el ciclo de trabajo de la señal PWM que se aplica al integrado A6260. (20%)

Si la señal de entrada es de 100KHz, tendrá 100.000 flancos de bajada a lo largo de 1 segundo, lo cual implica que en el intervalo de 2 segundos, serán 200.000, y por lo tanto ese será el valor de "cuenta".
El ancho del pulso alto pasa a ser ese valor dividido por 8 = 25.000.
*Y teniendo en cuenta que el periodo eran 32.000, el DC = 25.000 * 100 / 32.000 = 78,13%*

Problema 2 (100 minutos – 4 puntos)

Se quiere controlar el sistema de sensado de una cámara limpia para la fabricación de circuitos fotónicos integrados, utilizando el microcontrolador de referencia del curso. Este sistema dispone de un sensor de temperatura, un sensor de humedad y un sensor de fuego y debe permite transmitir la información de estos sensores a un PC instalado en la propia cámara limpia y al puesto de control central que se conecta con la cámara limpia mediante un enlace de fibra óptica de plástico. La conexión con el PC se realizará mediante una comunicación serie asíncrona y el envío de datos al puesto central se realizará utilizando una codificación FSK (Frequency Shift Keying), que se detalla más adelante. El sistema dispone también de un altavoz que emite una alarma sonora en el caso de que se detecte fuego en la cámara.

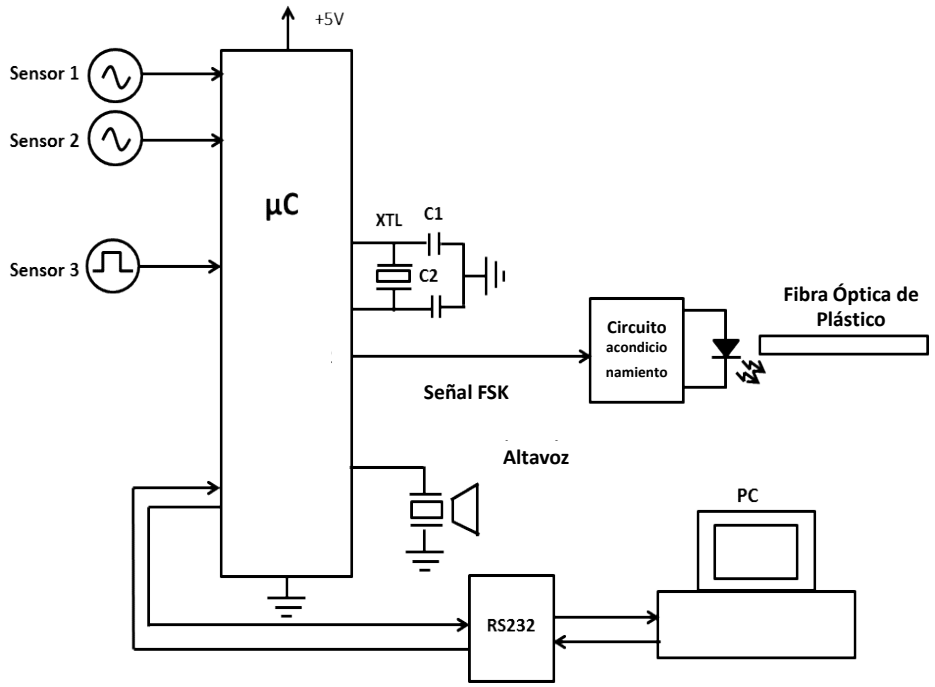


Fig 1.

A continuación se describen más detalladamente las características del sistema

- **El microcontrolador funciona con un oscilador de 8MHz.**
- **Sensores:** Los tres sensores del sistema tienen las siguientes características y prioridad en el sistema de control

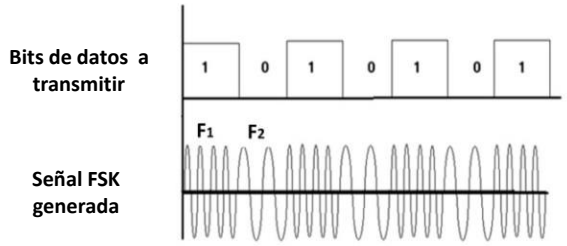
Sensor	Tipo	Resolución	Señal	Rango (V)	Nivel de Prioridad
1	Temperatura (0 – 99 ºC)	20 mV / ºC	analógica	0 - 2	No
2	Humedad Relativa (0 – 99 %)	20 mV / %	analógica	0 - 2	No
3	Fuego		digital	0 - 5	Alta

El sensor de incendios genera una señal digital a su salida. Cuando detecta fuego en la cámara< se produce un flanco de subida en esta señal. La señal permanece a nivel alto mientras el fuego está activo y cuando el fuego se extingue se produce un flanco de bajada en la señal de salida del sensor. Esta señal permanece a nivel bajo mientras no haya fuego en la cámara.

- **Señal de alarma de fuego:** Cuando el sensor de incendios detecta fuego en la cámara el microcontrolador genera una señal cuadrada de 1ms de periodo para activar el altavoz. Cuando el fuego se extingue esta señal también desaparece.
- **La comunicación serie con el PC se realiza a 19200 baudios, 8 bits, con 1 bit de inicio y 1 bit de parada.** Esta comunicación debe permitir enviar y recibir datos del PC.
- **El envío de la información al puesto de control se realiza generando en uno de los pines del microcontrolador, una señal FSK cuya estructura se detalla a continuación.**

El generador FSK se utiliza para generar dos señales de diferentes frecuencias que representan los dos posibles valores binarios: Para transmitir un ‘1’ lógico se genera una señal de una frecuencia f1 (100kHz) durante un tiempo (período de bit = 80µs) y para generar un ‘0’ lógico una señal de frecuencia f2 (50kHz). Estas señales pueden ser sinusoidales (como en el dibujo de la fig. 2a) o cuadradas.

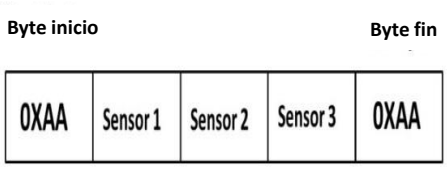
La estructura de los datos de los sensores que se envían al puesto de control central cada 2 segundos es la que aparece en la Fig. 2b.



Características en el Sistema de la cámara limpia

- ✓ f1 = 100 kHz (señal cuadrada9
- ✓ f2 = 50 kHz.(señal cuadrada)
- ✓ Ciclo de trabajo del 50%
- ✓ Período de bit = 80 µs.

a



Características del paquete de datos a transmitir al puesto central de control

- El paquete empieza tiene un byte de inicio y uno de fin, ambos codificados con **0XAA**. Todos los bytes del paquete se empiezan a transmitir por el bit LSB.
- La información del sensor digital de incendios (Sensor 3) se transmite con **0XB4** y **0XB6** cuando su salida está a nivel alto y a nivel bajo, respectivamente.

b

Fig 2

El diagrama de flujo que se ha desarrollado para el programa principal es el que se muestra a continuación:

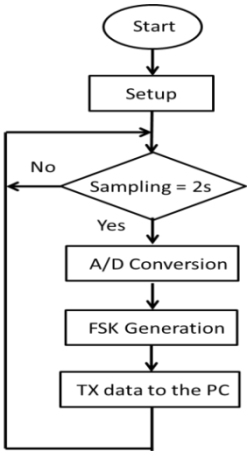


Fig. 3

Se pide:

1. Haga una tabla indicando las tareas que realiza el microcontrolador, los periféricos utilizados y los pines de conexión del microcontrolador (indicando si funcionan como analógicos o digitales y si son de entrada o salida). (20%)
2. Indique como configurar el convertidor A/D y la comunicación serie para esta aplicación. Indique de forma justificada los valores necesarios en todos los registros de configuración. (20%)
3. La rutina "A/D Conversion" que aparece en el diagrama de flujo de la Fig. 3 selecciona el canal analógico de entrada y realiza la conversión A/D de cada uno de los canales. Represente el diagrama de flujo de esta rutina. Considere que los resultados de las conversiones A/D se almacenan en un vector declarado como : char ADresult [ ]; (15%)
4. La rutina "setup" que aparece en el diagrama de flujo de la Fig. 3 realiza la configuración inicial de todos los recursos del microcontrolador. Indique razonadamente qué recurso/s del microcontrolador atenderá mediante interrupción y cuál debe ser el contenido de los registros asociados a la/s interrupción/es en esta rutina de inicialización. (10%)
5. Realice el diagrama de flujo y el código C de la/s rutina/s de atención a la interrupción (20%)
6. Realice el diagrama de flujo de la rutina "FSK Generation" que aparece en el diagrama de flujo de la Fig.3. (15%)

SOLUCIÓN:

1. **2 puntos**

TAREA	PERIFERICO	PIN
Sensor 1 para consultar temperatura	ADC - Canal 0	PA4 (entrada analógica)
Sensor 2 para consultar humedad	ADC - Canal 1	PA5 (entrada analógica)
Sensor 3 para consultar fuego	GPIO	PA0 (entrada digital - Prioridad alta)
Comunicación con el PC	USART	PB6 (salida digital con AF - RxD) PB7 (entrada digital con AF - TxD)
Periodo de muestreo en el prog. princ.	TIMER 0 (2 segundos)	Sin HW asociado
Periodo del bit para la señal FSK	TIMER 1 (80 µs)	Sin HW asociado
Señal FSK para el puesto de control	TIMER 2 (con PWM)	PA6 (salida digital con AF - PWM)
Alarma de fuego	TIMER 3 (con PWM)	PA7 (salida digital con AF - PWM)

2. **2 puntos**

ADC:

// Configuración del ADC: 2 canales (PA4 y PA5), 12 bits de resolución, sin interrupciones, conversión continua y modo scan.

```
GPIOA->MODER |= 0x00000F00;    // PA4 y PA5 como analógico (…..00/1111/00……)
ADC1->CR2 &= ~(0x00000001);    // ADON = 0 (ADC apagado)
ADC1->CR1 = 0x00000100;        // OVRIE = 0 (deshabilitada la habilitación por interrupción)
                               // RES = 00 (resolución = 12 bits, por ejemplo)
                               // SCAN = 1 (modo scan deshabilitado, para leer los 2 canales)
                               // EOCIE = 0 (deshabilitada la interupción por EOC)
ADC1->CR2 = 0x00000412;        // EOCS = 1 (activado el bit EOC al acabar cada conversión)
                               // DELS = 001 (retardo de la conversión hasta que se lea el dato
                               // anterior, por ejemplo)
                               // CONT = 1 (conversión continua, más fácil)
ADC1->SMPR1 = 0;               // Sin sampling time (4 cycles)
ADC1->SMPR2 = 0;
ADC1->SMPR3 = 0;
ADC1->SQR1 = 0x00100000;       // 2 elementos solo en la secuencia
ADC1->SQR5 = 0x000000A4;       // Los elementos son los canales AIN4 y AIN5 (…../00101/00100)
ADC1->CR2 |= 0x00000001;       // ADON = 1 (ADC activado)
```

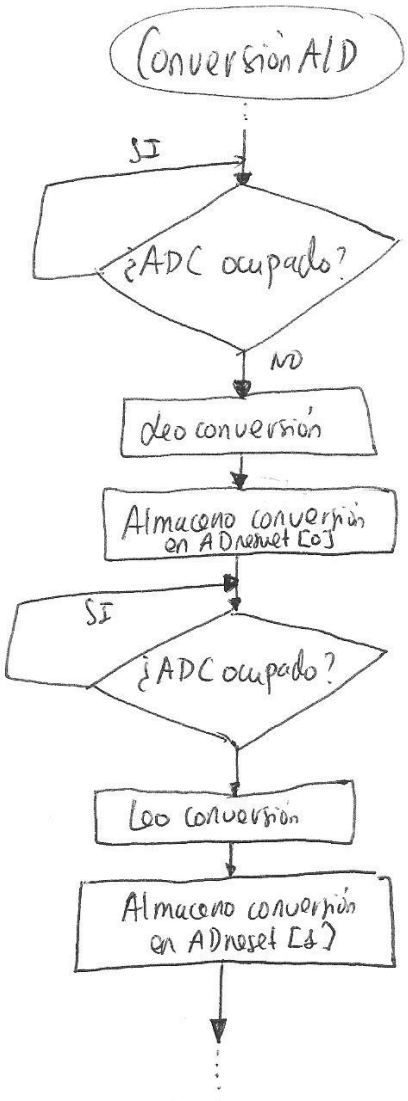
USART:

// Configuración del USART: Elegimos el envío y recepción por espera activa para simplificar el código, aunque empeoramos el rendimiento del micro, no dicen nada al respecto, pero también valdría recepción por interrupciones.

```
// PB7 y PB6 como AF - USART
GPIOB->MODER |= (0x01 << (2*7+1)); // (10 la posición de PB6 en MODER)
GPIOB->MODER &= ~(0x01 << (2*7));
GPIOB->MODER |= (0x01 << (2*6+1)); // (10 la posición de PB7 en MODER)
GPIOB->MODER &= ~(0x01 << (2*6));
GPIOB->AFR[0] = 0x77000000;        // AF7 = 0111 -> La función AF es la función USART
                                   // 0111 en la posición del PB6 y del PB7 en AFR[0])
```

```
// Comunicación 19200,8,N,1 con PCLK = 32MHz (el enunciado dice que el oscilador externo es de
// 8Mhz, pero lo que interesa es el PCLK y 32MHz se pueden conseguir sin problemas utilizando los
// preescalados adecuados del subsistema de reloj, tal como se explicó en clase)

USART1->CR1 = 0x0000000C;  // OVER8 = 0 -> Oversampling = 16 -> DIV_fraction[3:0] en el USARTx->BRR
usa 4 bits
                                    // UE = 0 –> Deshabilitación de la USART
                                    // M = 0 –> Tamaño de palabra: 8 bits
                                    // PCE = 0 -> Control de paridad deshabilitado
                                    // TXEIE, TCIE, RXNEIE = 0 -> Sin IRQs en la comunicación
                                    // TE = 1 -> Habilitación de la transmisión
                                    // RE = 1 -> Habilitación de la recepción

USART1->CR2 = 0x00000000;           // STOP = 00 –> Números de bit de parada = 1
USART1->BRR = 0x00000683;           // 19200 baudios -> USARTDIV = 104,1875 (mirar tabla de
                                    // teoría para 32Mhz y OVER8 = 0)
                                    // Mantisa = 104 = 0x68;
                                    // Fracción = 0,1875*16 = 3 = 0X03;

// Habilitación del USART
USART1->CR1 |= 0x01 << 13; // UE = 1 en el bit 13 del CR1
```

3. **1,5 puntos**



4. **1 punto**

- Para las señales analógicas no hace falta utilizar interrupciones porque son de prioridad baja
- Según lo indicado en el enunciado, el sensor de fuego debe tener prioridad alta, luego debe configurarse a una entrada digital pero como interrupción externa.
- La comunicación con el PC se ha indicado en el punto 2 que se harán por espera activa, tanto el envío como la recepción para simplificar
- Para las funciones de los 2 temporizadores de contaje de tiempo (TIM 0 y 1) se puede elegir o no usar interrupciones, como no nos dicen nada al respecto se elige no utilizarlas para simplificar.
- Para las funciones de los 2 temporizadores de las señales de PWM (TIM 3 y 4), se puede elegir o no usar interrupciones, como no nos dicen nada al respecto se elige no utilizarlas para simplificar.

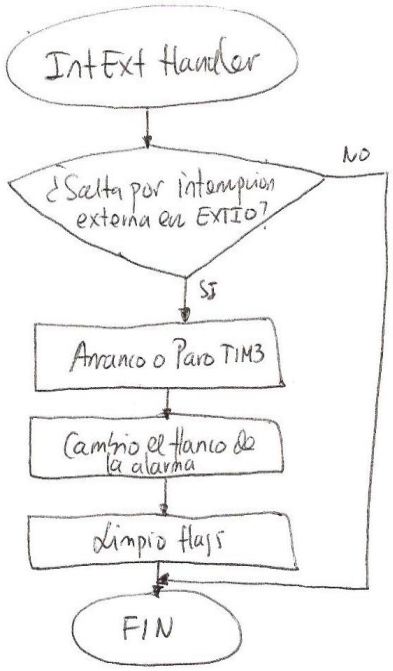
Por tanto sólo se utilizan interrupciones externas para el sensor de fuego. A continuación se muestran la configuración de sus registros asociados.

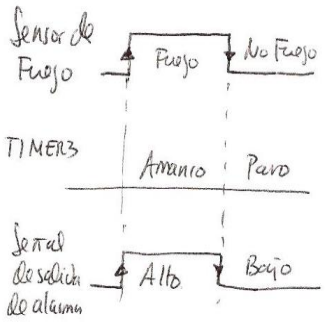
```
// PA0 como entrada (00)
  GPIOA->MODER &= ~(1 << (0*2 +1));
```

```
  GPIOA->MODER &= ~(1 << (0*2));

// Entradas sin pull-up, pull-down (00)
  GPIOA->PUPDR &= ~(11 << (0*2));

// Configuración de EXTI0 por flanco de bajada
  EXTI->FTSR |= 0x01;          // Un '1' habilita el evento por flanco de bajada en EXTI0
  EXTI->RTSR &= ~(0x01);     // Un '0' inhabilita el evento por flanco de subida en EXTI0
  SYSCFG->EXTICR[0] = 0;   // La EXTI0 la provoca el bit 0 del GPIOA (PA0)
  EXTI->IMR |= 0x01;           // Un '1' habilita la EXTI0, no la enmascara
  NVIC->ISER[0] |= (1 << 6);  // No hay espera activa del evento. Habilito la EXTI0 en el NVIC (posición 6).
                              // Si se activa PA0, o sea hay una interrupción
                               // externa en PA0 (EXTI0), salto al NVIC, consulto la EXTI0 en el NVIC
```

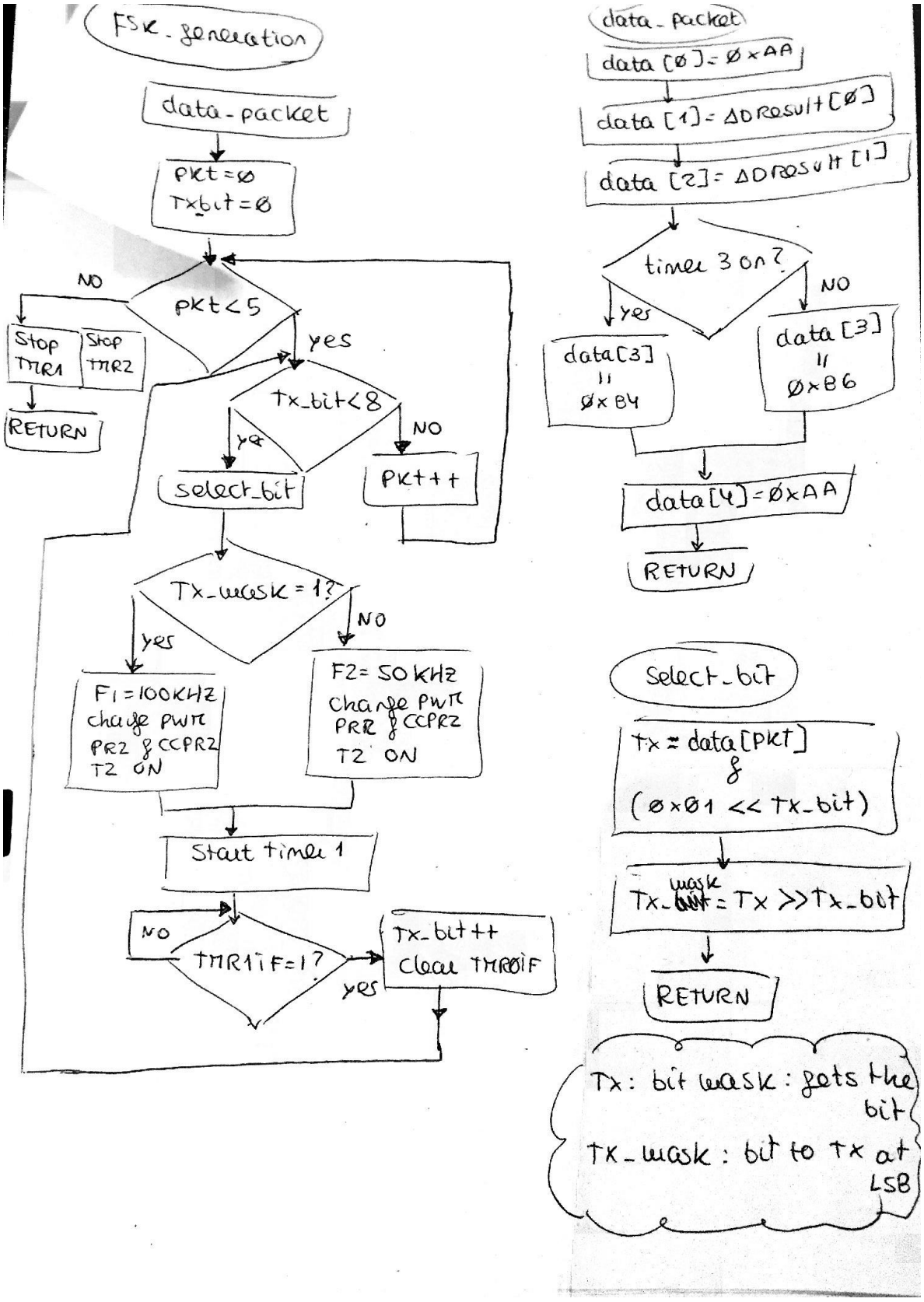
5. **2 puntos**

Sólo hay diagrama de flujo para la rutina de interrupción EXTI0_IRQHandler asociada al PA0





6. **1,5 puntos**


FSK_generation
data_packet
pkt=0
Txbit=0
pkt<5
NO
yes
Stop TMR1
Stop TMR2
RETURN
Tx_bit<8
yes
NO
select_bit
pkt++
Tx_mask=1?
yes
NO
F1=100KHz
change PWM PR2 & CCPR2
T2 ON
F2=50KHz
change PWM PRR & CCPR2
T2 ON
Start timer 1
NO
TMR1IF=1?
yes
Tx_bit++
Clear TMR0IF
data_packet
data[0]=0xAA
data[1]=ADResult[0]
data[2]=ADResult[1]
timer 3 on?
yes
NO
data[3] = 0xB4
data[3] = 0xB6
data[4]=0xAA
RETURN
Select_bit
Tx = data[pkt] & (0x01 << Tx_bit)
Tx_mask = Tx >> Tx_bit
RETURN
Tx: bit wask: gets the bit
Tx_wask: bit to Tx at LSB