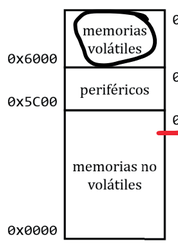
Enero 2019

Ejercicio 3	Mapas de memoria	**1,6 puntos**

Un determinado microprocesador, cuya longitud de palabra es de 8 bits, presenta el siguiente mapa de memoria y periféricos:



Las direcciones generadas por este procesador (y las indicadas en el mapa anterior) son direcciones de byte.

En este mapa se desean ubicar los siguientes recursos:

- Un par de memorias EEPROM de 32 Kbit de capacidad para formar un bloque de 64 Kbit a partir de la dirección más baja posible del mapa. Estas memorias tienen el CS activo a nivel bajo.
- Una memoria SRAM de 16 Kbit de capacidad en la dirección más baja posible del mapa. Esta memoria tiene el CS activo a nivel bajo.
- Un periférico genérico que ocupa 4 posiciones en el mapa. Este periférico tiene el CS activo a nivel alto y desea ubicarse en la posición más baja posible del mapa.

Apartado A. ¿Cuántos bits tiene el bus de direcciones de este microprocesador? Justifique la respuesta.

Apartado B. Indique, para ***cada chip*** de memoria, su organización y la anchura, en bits, de su bus de direcciones.

Memoria	Organización	Anchura de su bus de direcciones (bit)
EEPROM0/1	4K × 8	12
SRAM	2K × 8	11

Apartado C. Obtenga la expresión booleana de las señales de selección de las memorias EEPROM, $CSn_{EEPROM0}$ y $CSn_{EEPROM1}$. Recuerde que dichas memorias desean ubicarse en las posiciones más bajas posible del mapa. Emplee el tipo de decodificación necesario para minimizar esta lógica.

Apartado D. Obtenga la expresión booleana de la señal de selección de la memoria SRAM, CSn_{SRAM}. Recuerde que dicha memoria desea ubicarse en las posiciones más bajas posibles del mapa. Debe emplearse decodificación completa.

Apartado E. Si se desease hacer una decodificación completa del periférico ¿cuántos bits del bus de direcciones del microprocesador se necesitarían para obtener la señal CS_{PERIF}? Justifique la respuesta.

Ejercicio 4	Interrupciones
	1.6 puntos

Un determinado microcontrolador está basado en un *core* ARM *Cortex-M0*. Dicho *core* cuenta con un NVIC (*Nested Vectored Interrupt Controller*). Entre los periféricos incluidos en dicho microcontrolador se encuentran: un *timer*, un ADC, un DAC y un GPIO, todos ellos capaces de generar interrupciones mediante sendas señales de petición de interrupción IRQ (IRQ_{TMR}, IRQ_{ADC}, IRQ_{DAC} y IRQ_{GPIO} respectivamente). Además, una circuitería externa al microcontrolador tiene acceso a la señal de interrupción no enmascarable NMI.

El programa corriendo sobre el procesador ha configurado los niveles de prioridad de las interrupciones configurables según la siguiente tabla:

Interrupción	Nivel de prioridad
IRQ_{TMR}	196
IRQ_{ADC}	128
IRQ_{DAC}	64
IRQ_{GPIO}	0

y, además, ha enmascarado todas la enmascarables, excepto las asociadas a IRQ_{TMR} y IRQ_{DAC}.

A su vez, las rutinas de atención a las interrupciones, ISR, asociadas a las distintas interrupciones presentan los tiempos de ejecución dados en la siguiente tabla:

Interrupción	Tiempo de ejecución de su ISR (µs)
NMI	0.50
IRQ_{TMR}	0.75
IRQ_{ADC}	1.25
IRQ_{DAC}	1.75
IRQ_{GPIO}	1.00

Todas estas IRQ y NMI son activas por flanco. Suponga que el tiempo de latencia para todas las interrupciones es despreciable.

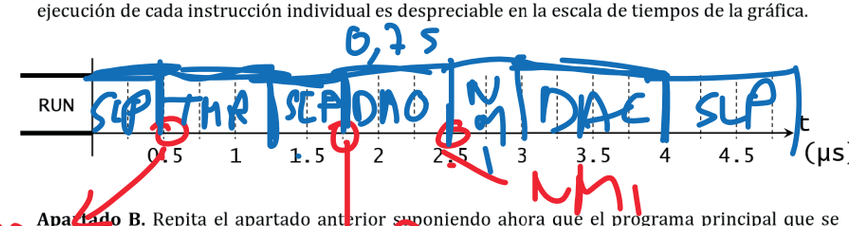
El programa principal que el procesador está ejecutando es:

```
        CPSIE   I       ; __enable_irq()
Loop
        WFI
        B       Loop
```

Apartado A. En estas circunstancias se producen flancos activos en estas líneas IRQ y NMI en los instantes dados en la siguiente tabla:

Interrupción	Instantes en los que ocurren flancos activos (µs)
NMI	2.50
IRQ_{TMR}	0.50
IRQ_{ADC}	3.50
IRQ_{DAC}	1.75
IRQ_{GPIO}	4.00

Complete el gráfico indicando para cada momento qué ISR (NMI, TMR, ADC, DAC o GPIO) está ejecutando el procesador, en qué intervalos de tiempo el procesador está dormido (estos márquelos como SLP, de *SLeeP*) y en qué intervalos de tiempo se está ejecutando el programa principal sin que el procesador duerma (márquelos como RUN). $t = 0$ s coincide con el instante en el que se ejecuta la instrucción `CPSIE I` del programa principal. Suponga que el tiempo de ejecución de cada instrucción individual es despreciable en la escala de tiempos de la gráfica.



Apartado B. Repita el apartado anterior suponiendo ahora que el programa principal que se ejecuta es:

```
        CPSIE   I       ; __enable_irq()
Loop
        NOP
        B       Loop
```

De nuevo, $t = 0$ s coincide con el instante en el que se ejecuta la instrucción `CPSIE I` del programa principal.

Apartado C. ¿En cuál de los dos casos anteriores es previsible un menor consumo de energía por parte del procesador? Justifique su respuesta.

Ejercicio 5	Periféricos
	1.6 puntos

Sobre el µcontrolador de una placa *mbed* LPC1768 (o Nucleo-l432kc) está corriendo el siguiente código.

```
#include "mbed.h"
#include "pinout.h"   // para A_PIN, ENA_PIN y PWM_PIN

#define     BAUDRATE  (9600)
#define     US        (1000000)

AnalogIn    a(A_PIN);
InterruptIn ena(ENA_PIN);
Serial      pc(USBTX, USBRX);
PwmOut      pwm(PWM_PIN);

bool        ena_rise_evnt;
bool        ena_fall_evnt;

void ena_rise_isr(void) {
  ena_rise_evnt = true;
}

void ena_fall_isr(void) {
  ena_fall_evnt = true;
}

void main(void) {
  int32_t   n = 13;

  pc.baud(BAUDRATE);
  pc.format(7, pc.None, 2);

  ena.rise(ena_rise_isr);
  ena.fall(ena_fall_isr);

  pwm.period_us( (6*US) / BAUDRATE );
  pwm.pulsewidth_us( US / BAUDRATE );

  while(true) {
    if(ena_rise_evnt) {
      ena_rise_evnt = false;
      n = a.read_u16() / 780;
      pwm.pulsewidth_us( (US*(n + 17)) / (21*BAUDRATE) );
    }
    if(ena_fall_evnt) {
      ena_fall_evnt = false;
      if(n >= 46) pc.putc(n);
    }
    __disable_irq();
    if(!ena_rise_evnt && !ena_fall_evnt) __WFI();
    __enable_irq();
  }
}
```


En el siguiente cronograma se muestra la evolución de las señales de entrada A y ENA en los pines A_PIN y ENA_PIN. En este cronograma el eje de abscisas está graduado en unidades de 1/9600 s ≈ 104.167 µs y $t = 0$ coincide con el instante en que se entra en `main()`. La tensión de alimentación del microcontrolador V_{DD} es 3.3 V y coincide con la de referencia del ADC.

Complete el cronograma para las señales PWM y USBTX (asociadas a los pines PWM_PIN y USBTX) y también para el valor de la variable n a lo largo del tiempo. Suponga que el tiempo necesario para la ejecución de cada una de las líneas de código del programa es despreciable en la escala de tiempos del cronograma.

Para este problema suponga que el objeto `PwmOut` se comporta de modo que, cada vez que se llama a cualquiera de sus métodos `period()`, `period_ms()` o `period_us()`, inmediatamente comienza la generación (flanco de subida) de un nuevo pulso, y todos los pulsos siguientes (sus flancos de subida) estarán espaciados el tiempo indicado en el parámetro empleado al llamar al método, y así hasta que de nuevo se llame a alguno de estos tres métodos.