Examen Argumentativo

SoC FJ2025

Name: David Martínez Martín del Campo Student ID: A01645721

**Microcontroladores Parte 1 (10 puntos)**

1. ¿Por qué es importante desactivar las interrupciones cuando se configuran los registros de un periférico o cuando se hace una modificación al vuelo? Realiza una investigación sobre cómo y porqué se realiza un test inicial de los periféricos durante el proceso de arranque del sistema. Justifica tu respuesta **(2.5 puntos)**

**Al configurar registros de control de un periférico o al hacer modificaciones en tiempo de ejecución, es importante desactivar las interrupciones ya sea de forma general para el microcontrolador o para el periférico especifico. Esto se hace para evitar que una ISR se dispare mientras los registros del periférico se encuentran en algún estado inestable lo que podría causar lectura de datos inválidos, escritura en registros incorrectos o corrupción en el estado del periférico.**
**Durante el proceso de arranque, el firmware (bootloader en sistemas embebidos) ejecuta rutinas de diagnóstico conocidas como Power-On Self-Test (POST). Parte de este proceso incluye inicializar y verificar el estado de los periféricos críticos, como timers, UARTs, ADCs o GPIOs.**

2. ¿Puedes describir con tus propias palabras porque se usa el mapa de memoria en un microcontrolador? ¿Cómo se usa para direccionar datos de y hacia los periféricos mediante el código de la aplicación? (2.5 puntos)

**El mapa de memoria de un micro define la organizacion del espacio de direcciones de memoria y que regiones de este estaran asignadas a RAM, ROM, registros de perifericos, y EEPROM. Este mapa se usa para que el CPU pueda acceder a los recursos del sistema de forma estructurada y predecible.**

**En arquitecturas con mapeo de memoria a periféricos (memory-mapped I/O), los registros de control y datos de los periféricos están asignados a direcciones específicas dentro del mapa de memoria. El CPU accede a estos registros igual que accede a una posición de memoria RAM: mediante instrucciones estándar de lectura y escritura (LDR, STR, MOV).**

3. ¿En qué casos se utiliza una resistencia pull-up y cuándo una configuración pull-down para una entrada determinada? Investiga y justifica tu respuesta **(2.5 puntos)**

**Una resistencia pull-up genera una entrada inactiva alta estable o un 1 lógico por lo que la señal digital que recibiría el sistema cuando esta entrada se interrumpa será baja o 0, Un ejemplo de esto puede ser un botón cuyo funcionamiento pasa alta de forma inactiva y baja cuando se presiona.**

**En el caso de una resistencia pull-down, lo que sucede es exactamente lo contrario, la resistencia va directamente a GND por lo que su señal inactiva es baja y en el ejemplo, el botón pasaría 1 al ser presionado.**

4. Realiza lo mismo (una investigación rápida) para el drive strength que se configura usando el bit 6 del registro de control de pin de la KL25, ¿para qué se usa? **(2.5 puntos)**

**En la MKL25Z4 el bit 6 del PORTx_PCRn se llama DSE (Drive Strength Enable). Y configura drive strength bajo cuando está en 0 (por defecto) y drive strength alto cuando está en 1.**

**La utilidad de estas configuraciones es permitir que los pines que se están configurando puedan entregar corientes mayores o menores, lo cual se vuelve muy útil cuando se conectan dispositivos con alta capacitancia, se necesita una conmutación más rápida en señales digitales, o cuando el pin controla directamente un transistor, LED, o carga moderada.**

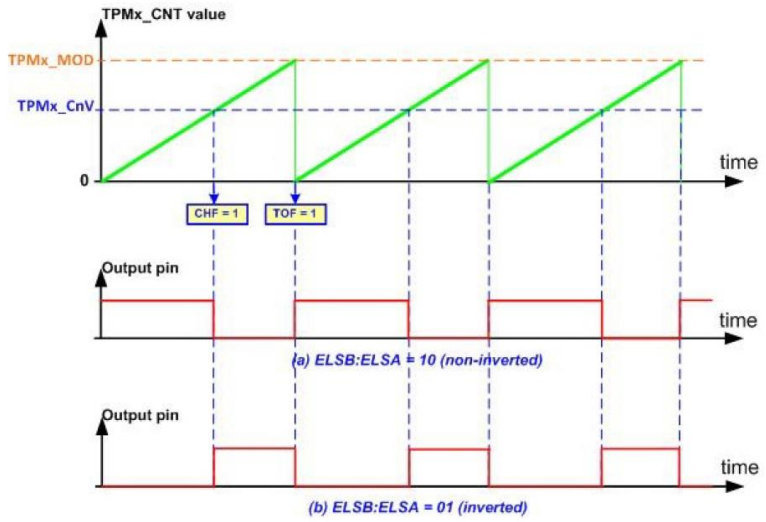
**Microcontroladores Parte 2 (10 puntos)**

5. Por que es útil introducir PWM con deadband para el control de motores, busca algunos ejemplo de uso y discute tus hallazgos **(2.5 puntos)**

**Un dead time o deadband PWM es útil en el control de motores, especialmente en configuraciones push-pull o puente H, donde dos transistores (alto y bajo) controlan cada rama del motor y se usan para evitar que ambos transistores de una misma rama conduzcan simultáneamente, lo que causaría un cortocircuito entre Vcc y GND (llamado shoot-through). El deadband introduce un pequeño retraso entre el apagado de un transistor y el encendido del otro.**

**Algunos casos de uso podrian ser motores DC en configuracion de puente H, motores trifasicos o convertidores DC-DC conmutados.**

6. En una aplicación PWM determinada, necesitamos una frecuencia de salida PWM de 60 Hz. Utilizando la frecuencia del módulo TPMx de 41,98 MHz, descubra el valor del registro TPMx_MOD. Cual es el valor de TMPx_CnV para un duty cycle de 37.5%? En qué casos es útil usar el PWM invertido y no invertido? **(2.5 puntos)**




**Frecuencia de salida:**

$$TPMx - MOD = \frac{fclk}{fPWM} - 1 = \frac{41.98 \cdot 10^6}{60} - 1 \approx 699666$$

**CnV con duty cycle de 37.5%:**

$$TPMx - CnV = Duty \cdot (TPMx - MOD + 1) = 0.375 \cdot 699666 \approx 262375$$

**El PWM inverso se usa para drivers activos en bajo o salidas complementarias mientras que el no inverso para control estándar de cargas activas en alto.**

7. Si deseamos configurar el timer como contador de eventos es necesario habilitar la función adecuada en el port control register (PCR), asumiendo que el pin ya fue configurado para este fin, como debemos configurar el timer para que genere una interrupción cada vez que detecta que un opto-acoplador detecta que pasa una objeto 50 veces (5 puntos)

```
/* counter clock must be present */
SIM->SOPT2 |= 0x01000000;
TPM0->SC = 0; /* disable timer while configuring */
TPM0->SC = 0x80; /* prescaler /1 and clear TOF */
TPM0->MOD = 0x????; /* max modulo value */
TPM0->CNT = 0; /* clear counter */
TPM0->SC |= 0x????; /* enable timer and use LPTPM_EXTCLK, enable interrupts,
enable ???? detection */
```

**Para realizar esta configuración primero usamos un pin externo configurado para la función de TPM_CLKINx (fuente externa de reloj del TPM).**

**El registro MOD define cuántos eventos se deben contar antes de generar una interrupción por desbordamiento.**

**El TPM debe estar configurado para usar LPTPM_EXTCLK como fuente de reloj.**

**El codigo incluyendo la configuracion se veria algo asi**
**SIM->SOPT2 |= 0x01000000;  // Selecciona LPTPM_EXTCLK como fuente de reloj para TPM**

**TPM0->SC = 0x00;  // Desactiva temporizador mientras se configura**
**TPM0->MOD = 49;  // Cuenta hasta 49 (0–49) => 50 eventos**
**TPM0->CNT = 0;  // Reinicia el contador**
**TPM0->SC |= 0x88;  // Habilita TPM:**
**// - bit 7: TOF (clear)**
**// - bit 3: TOIE (enable overflow interrupt)**
**// - bits 1-0: CMOD = 01 => LPTPM_EXTCLK como reloj**

```
SIM->SOPT2 |= 0x01000000;   // Selecciona LPTPM_EXTCLK como fuente de reloj para TPM

TPM0->SC = 0x00;            // Desactiva temporizador mientras se configura
TPM0->MOD = 49;             // Cuenta hasta 49 (0–49) => 50 eventos
TPM0->CNT = 0;              // Reinicia el contador
TPM0->SC |= 0x88;           // Habilita TPM:
                            // - bit 7: TOF (clear)
                            // - bit 3: TOIE (enable overflow interrupt)
                            // - bits 1-0: CMOD = 01 => LPTPM_EXTCLK como reloj
```

**Sistemas Operativos en Tiempo Real (25 puntos)**

8. **(6 puntos)** Describe con tus palabras qué es un sistema de tiempo real y cuales son los dos tipos principales de RTOS por su tipo de respuesta en el tiempo. Explica la diferencia entre ellos.

**En pocas palabras, RTOS es un sistema operativo en el que el tiempo determinado en que una tarea se comlpleta tiene la misma importancia que el**

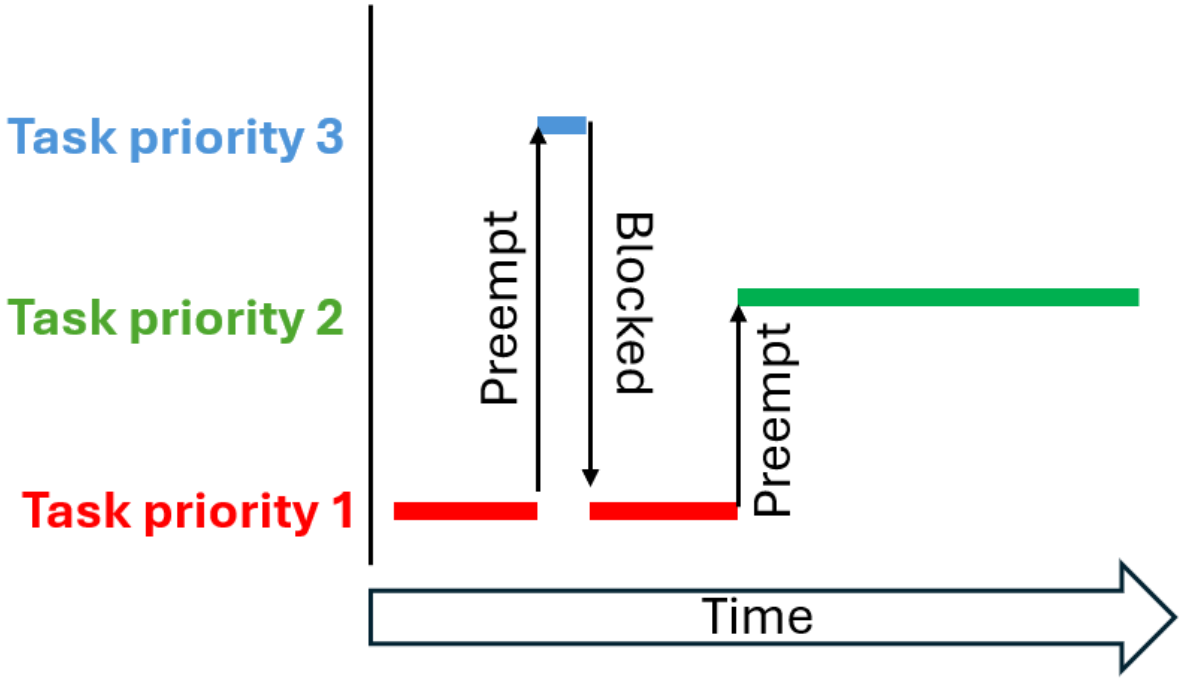
**resultado lógico de la misma, es decir que un RTOS le da tanta importancia al cuándo como al que.**

**Los dos tipos de RTOS son hard RTOS en el que las respuestas de las tareas siempre deben caer en el tiempo especificado sin margen de error (para tareas que requieren un nivel muy alto de precisión).**
**Y soft RTOS que define un tiempo esperado en que las tareas deben cumplirse pero que permite que funcione dentro de un margen de error sin consecuencias críticas.**

9. **(5 puntos)** Explica que es el cambio de contexto entre tareas de FreeRTOS, utilizando una analogía de tu vida cotidiana.

**Digamos que estoy preparando la comida, metí la carne al horno y puse un temporizador de 35 minutos para sacarla cuando sonara. Mientras la carne estaba en el horno utilice ese tiempo para empezar a preparar una salsa, 35 minutos después estoy cortando ajo y preparando una mezcla de especias cuando de repente suena mi alarma, entonces, la tarea que estaba en plano secundario (preparar la carne) y que es de mayor prioridad vuelve al primer plano, dejo el cuchillo y automáticamente me voy a sacar la carne del horno.**

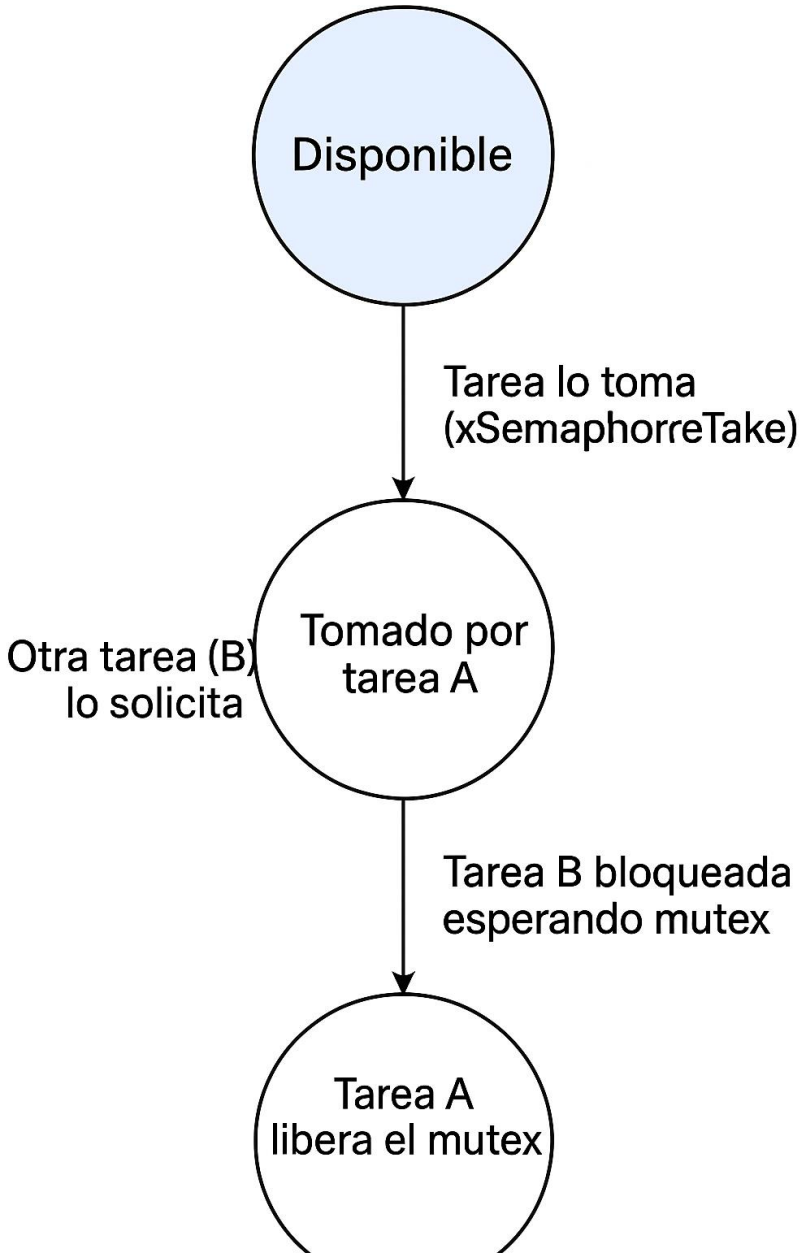
10. **(4 puntos)** Asumiendo que la tarea roja y la tarea azul requieren usar un recurso compartido, cuyo acceso está restringido mediante un semaphore de FreeRTOS, explica si el comportamiento del siguiente diagrama de tiempos es correcto y si es deseable. Justifica tu respuesta.




**En el diagrama se ve cómo una tarea de baja prioridad (roja) es interrumpida por una de alta (azul) y luego por una de prioridad media (verde). Si la tarea roja y la azul comparten un recurso protegido por un semáforo, puede ocurrir inversión de prioridades: la tarea azul queda bloqueada esperando el semáforo que tiene la tarea roja, pero esta no puede continuar porque la tarea verde la sigue interrumpiendo.**

**Para evitar este problema, podemos usar FreeRTOS para implementar herencia de prioridades en los mutex. Esto permite que la tarea roja herede temporalmente la prioridad de la azul, libere el recurso rápidamente y se evite el bloqueo indebido. Por eso, al usar recursos compartidos entre tareas con distinta prioridad, es fundamental usar mutex con herencia de prioridad, no semáforos binarios.**

11. **(3 puntos)** Dibuja el diagrama de estados de un Mutex.




Este diagrama fue dibujado utilizando chatGPT basado en un curso de DigiKey sobre los básicos de freeRTOS.

12. **(7 puntos)** Explica con tus palabras cómo funciona el esquema de manejo diferido de interrupciones en los sistemas de tiempo real. ¿Cuál de los recursos disponibles en

la API de FreeRTOS utilizarías para implementar este esquema y por qué? (Justifica tu respuesta)

**El manejo diferido de interrupciones es una técnica donde la ISR se limita a realizar el trabajo mínimo necesario, como capturar datos o señalar un evento, y pospone el procesamiento más pesado o prolongado para una tarea en modo usuario (fuera del contexto de interrupción).**

**Esto evita que la ISR bloquee otras interrupciones o afecte la latencia del sistema, ya que las ISRs tienen prioridad sobre las tareas y normalmente no permiten llamadas complejas del kernel.**

**Por lo general se utiliza una cola, una notificación a tarea o un semáforo binario. El más recomendado seria xTaskNotifyFromISR() ya que es la forma más ligera y rápida. Permite que una ISR despierte una tarea específica para continuar el procesamiento. Ideal si solo hay una tarea interesada en el evento.**

**Equipo de Laboratorio para Sistemas Embebidos (15 puntos)**

**14. (7.5 puntos)** Porque es importante limitar la corriente al momento de utilizar una fuente de voltaje durante el desarrollo de un sistema embebido?

**Es muy importante limitar la corriente al usar una fuente de voltaje en el desarrollo de sistemas embebidos para proteger el hardware ante cortocircuitos o errores de conexión, prevenir el sobrecalentamiento de componentes, evitar daños permanentes y mejorar la seguridad del desarrollador. Además, permite una depuración más segura al reducir el riesgo de fallos graves durante pruebas. Por ello, se recomienda usar fuentes con control de corriente ajustable.**

**15. (7.5 puntos)** Estás trabajando en el laboratorio con una tarjeta de desarrollo que comunica datos de un sensor de temperatura digital a través del protocolo I2C hacia un microcontrolador. Sin embargo, notas que los valores recibidos en el microcontrolador son erráticos o a veces nulos. Tienes acceso a los siguientes equipos de laboratorio:

¿Cuál(es) de estos equipos usarías para diagnosticar el problema y por qué?

- Multímetro digital
- Osciloscopio
- Analizador lógico
- Fuente de alimentación regulada
- Debugger de hardware (SWD/JTAG)
- Generador de funciones

¿Cómo abordarías el uso de estos equipos para diagnosticar el problema?

*Nota: Justifica cada selección y provee una descripción detallada de tu proceso.*

**El primer paso sería utilizar el multímetro digital para verificar que el sensor esté correctamente alimentado y que existan resistencias pull-up en las lineas correspondientes. Luego utilizaría el osciloscopio para observar la forma de onda en ambas líneas del bus I2C, buscando niveles lógicos correctos, tiempos de subida adecuados, y posibles interferencias o condiciones de bloqueo. A continuación, emplearía un analizador lógico, configurado con decodificación I2C, para capturar y analizar las tramas transmitidas, verificando direcciones, datos, y cualquier error de protocolo.**

**Drivers en Capas para Sistemas Embebidos (15 puntos)**

**16. (5 puntos)** Explica las ventajas de una arquitectura de software embebido por capas en comparación con un diseño monolítico. Justifica tu respuesta con un ejemplo de mantenimiento de código o migración de hardware.

**Una arquitectura de software embebido por capas tiene varias ventajas claras a diferencia de un diseño monolítico, ya que permite separar el sistema en módulos independientes como drivers, lógica de aplicación e interfaz. Esto facilita el mantenimiento, la reutilización de código y la portabilidad, ya que los cambios en una capa no afectan al resto del sistema. Por ejemplo, al migrar un sistema de medición de temperatura de un microcontrolador A a uno B, solo sería necesario modificar la capa de acceso al hardware si el diseño es por capas, manteniendo intacta la lógica de aplicación. En cambio, en un sistema monolítico, los cambios requerirían revisar gran parte del código, aumentando complejidad y riesgo de errores.**

**17. (10 puntos)** Implementa una arquitectura de software embebido por capas para el siguiente código que realiza una comunicación mediante UART, de tal manera que la funcionalidad original del codigo se mantenga.

El diseño debe incluir las siguientes funciones:

- MCAL_UART_Init
- MCAL_UART_SendChar
- HAL_UART_Init
- HAL_UART_SendChar
- HAL_UART_SendString
    - Utiliza esta función para mandar “Hi”, en lugar de mandar cada carácter por separado.

```c
#include "MKL25Z4.h"

void UART0_Init(void) {
    SIM->SCGC4 |= 0x400;        // Habilita el reloj para UART0
    SIM->SCGC5 |= 0x0200;       // Habilita el reloj para el puerto A

    PORTA->PCR[1] = 0x0200;     // UART0_RX
    PORTA->PCR[2] = 0x0200;     // UART0_TX

    UART0->C2 = 0;              // Desactiva UART0 para configuración
    UART0->BDH = 0;
    UART0->BDL = 0x1A;          // Baud rate: 115200 (aproximado)
    UART0->C4 = 0x0F;
    UART0->C2 = 0x08;           // Habilita solo transmisión
}

void UART0_SendChar(char c) {
    while (!(UART0->S1 & 0x80)); // Espera hasta que el buffer esté vacío
    UART0->D = c;
}

void main(void) {
    UART0_Init();
    UART0_SendChar('H');
    UART0_SendChar('i');
    UART0_SendChar('\n');
    while (1);
}
```

Haz uso de los siguientes tipos de datos y estructuras de configuración

```c
#ifndef UART_TYPES_H
#define UART_TYPES_H

#include <stdint.h>
#include "MKL25Z4.h"

typedef enum {
    UART_PARITY_NONE,
    UART_PARITY_EVEN,
    UART_PARITY_ODD
} UART_parity;

typedef enum {
    UART_STOPBITS_1,
    UART_STOPBITS_2
} UART_stop_bits;

typedef struct {
    UART0_Type *base;      // UART instance (e.g., UART0)
    uint32_t baud_rate;
    UART_parity parity;
    UART_Stop_bits stop_bits;
} UART_Config;


#endif
```

Para este ejercicio, está permitido escribir valores fijos (es decir, no configurables) a cualquier registro que no esté relacionado a la estructura de configuración UART_config. Cualquier parámetro relacionado a esta estructura debe ser configurable mediante esta estructura.

**La respuesta al ultimo ejercicio esta en el ZIP de la entrega.**