Informe de Laboratorio 1 - Periféricos Embebidos

Hecho por:
David Ramírez Betancourth (dramirezbe@unal.edu.co)
Cristian **COMPLETAR**

Introducción

En este laboratorio se exploró el funcionamiento de los periféricos embebidos en un microcontrolador STM32 Nucleo L476RG. Las pruebas realizadas tuvieron como objetivo medir diferentes parámetros asociados al tiempo de establecimiento de señales, procesamiento de eventos externos y transmisiones UART, utilizando herramientas como un analizador lógico y el programa Saleae. Esto permitió comprender los principios de funcionamiento y optimización de sistemas embebidos.

Objetivos

1. Configurar el botón PB1 (PC13) como entrada EXTi, el LED LD2 (PA5) como salida, y el USART2 (PA2, PA3) en modo asincrónico.
2. Usar un analizador lógico para capturar eventos y analizar tiempos de respuesta.
3. Medir tiempos de establecimiento de nivel lógico con y sin filtro RC.
4. Analizar el impacto de diferentes frecuencias del reloj del sistema en el tiempo de procesamiento.
5. Evaluar los tiempos de bit y de transmisión en paquetes UART con diferentes tasas de baudios.

Desarrollo

1. **Rebote Mecánico y Filtro RC**

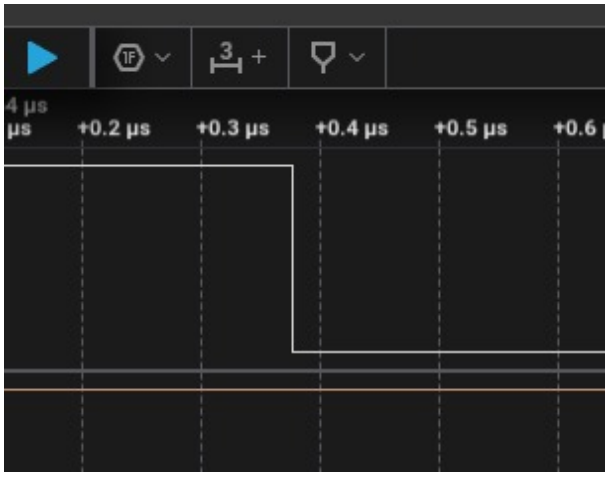
- Se inicializó el gpio para PB1 y LD2:

```
#define LED_PIN 5 // Pin 5 of GPIOA
#define BUTTON_PIN 13 // Pin 13 of GPIOC

void init_gpio_pin(GPIO_t *GPIOx, uint8_t pin, uint8_t mode)
{
    GPIOx->MODER &= ~(0x3 << (pin * 2)); // Clear MODER bits for this pin
    GPIOx->MODER |= (mode << (pin * 2)); // Set MODER bits for this pin
}

int main(void)
{
    configure_gpio();
    init_gpio_pin(GPIOA, 5, 1); // Set PA5 as output
    init_gpio_pin(GPIOC, 13, 0); // Set PC13 as input
}
```

- Se usó el programa Saleae para medir el tiempo de establecimiento del nivel lógico en un botón con y sin filtro RC.
    - Botón interno (PB1):



    - Botón externo sin filtro:

      Incluir imagen
- Los datos obtenidos mostraron qque el filtro RC reduce significativamente el ruido y el tiempo de rebote.

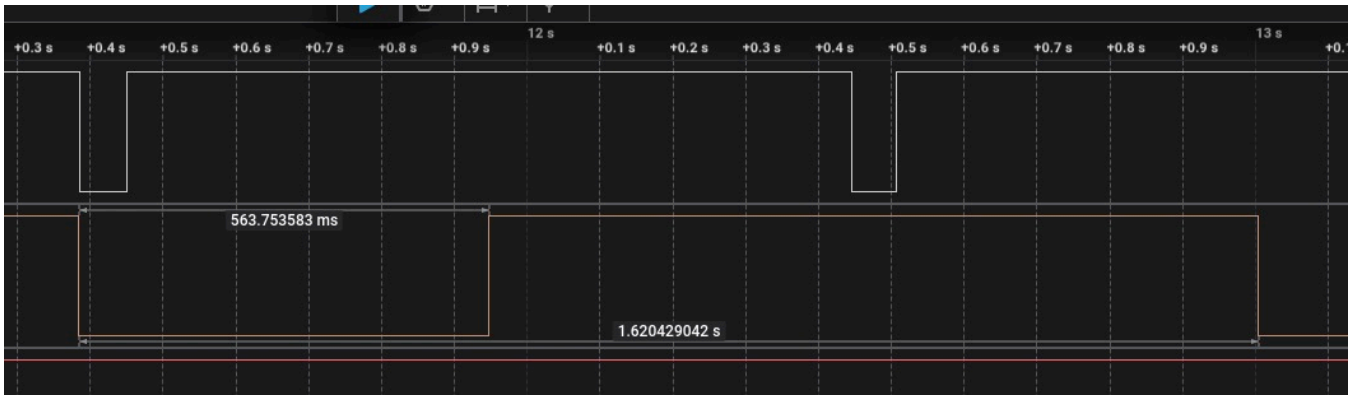
1. **Frecuencia del Reloj del Sistema**

- Se configuró el sistema para trabajar con frecuencias de reloj de 4 MHz.

```
void configure_systick_and_start(void)
{
    SysTick->CTRL = 0x4;     // Disable SysTick for configuration, use processor clock
    SysTick->LOAD = 3999;    // Reload value for 1 ms (assuming 4 MHz clock)
    SysTick->CTRL = 0x7;     // Enable SysTick, processor clock, no interrupt
}

int main(void) {
    configure_systick_and_start();
}
```

- Se midió el tiempo de procesamiento de un evento externo (cambio de nivel en LD2 al pulsar PB1) usando un analizador lógico (1.6204 segundos y 563.75 ms)



1. **Tiempo de bit y Transmisión de Paquetes UART**

- Configuramos USART2 para recibir y transmitir a un puerto serial:

```c
#define GPIOA ((GPIO_t *)0x48000000) // Base address of GPIOA
#define GPIOC ((GPIO_t *)0x48000800) // Base address of GPIOC

void configure_gpio_for_usart() {
    // Enable GPIOA clock
    *RCC_AHB2ENR |= (1 << 0);

    // Configure PA2 (TX) as alternate function
    GPIOA->MODER &= ~(3U << (2 * 2)); // Clear mode bits for PA2
    GPIOA->MODER |= (2U << (2 * 2));  // Set alternate function mode for PA2

    // Configure PA3 (RX) as alternate function
    GPIOA->MODER &= ~(3U << (3 * 2)); // Clear mode bits for PA3
    GPIOA->MODER |= (2U << (3 * 2));  // Set alternate function mode for PA3

    // Set alternate function to AF7 for PA2 and PA3
    GPIOA->AFR[0] &= ~(0xF << (4 * 2)); // Clear AFR bits for PA2
    GPIOA->AFR[0] |= (7U << (4 * 2));   // Set AFR to AF7 for PA2
    GPIOA->AFR[0] &= ~(0xF << (4 * 3)); // Clear AFR bits for PA3
    GPIOA->AFR[0] |= (7U << (4 * 3));   // Set AFR to AF7 for PA3

    // Configure PA2 and PA3 as very high speed
    GPIOA->OSPEEDR |= (3U << (2 * 2)); // Very high speed for PA2
    GPIOA->OSPEEDR |= (3U << (3 * 2)); // Very high speed for PA3

    // Configure PA2 and PA3 as no pull-up, no pull-down
    GPIOA->PUPDR &= ~(3U << (2 * 2)); // No pull-up, no pull-down for PA2
    GPIOA->PUPDR &= ~(3U << (3 * 2)); // No pull-up, no pull-down for PA3
}

int main(void) {
    configure_gpio_for_usart();
    UART_Init(USART2);

    UART_send_string(USART2, "Connected...\r\n");

    uint8_t buffer[1];

    UART_receive_it(USART2, buffer, 1);
```

```
    while (1) {
        if (rx_ready != 0) {
            UART_send_string(USART2, "Received: ");
            UART_send_string(USART2, (char *)buffer);
            UART_send_string(USART2, "\r\n");
            UART_receive_it(USART2, buffer, 1);
            rx_ready = 0;
        }
    }
}
```

- Usando el USART2, se transmitió el paquete “Connected...\r\n” a una terminal y se recibió el mensaje "000000000"
- Con un analizador lógico, se midieron los tiempos de bit, byte y transmisión y recepción total del paquete:

Incluir imagen

Conclusiones

1. El uso de un filtro RC es esencial para mitigar el ruido y mejorar la estabilidad de las señales generadas por botones mecánicos.
2. La transmisión UART es directamente proporcional a la tasa de baudios, permitiendo optimizar la velocidad de comunicación en sistemas embebidos dependiendo de las necesidades del proyecto.
3. Herramientas como el analizador lógico son indispensables para la validación y análisis de desempeño en periféricos embebidos.