Department of

Electrical & Electronics Engineering

Abdullah Gül University

Lab Experiment 5 Report

EE3002 - EMBEDDED CONTROL SYSTEMS DESIGN CAPSULE

Submitted on: Submitted by: Pınar Yılmaz

Lab Partners: Ahmed Tamer

Lab Instructors: Atıf Kerem Şanlı

Grade: / 100

OBJECTIVE

The goal of this lab to use the Timer2 to blink an LED connected to PC13. This lab experiment demonstrates how to set up and use the Timer and GPIO peripherals to produce periodic LED toggling.

BACKGROUND

This lab focuses on toggling LED using Timer2 as a counter. In this task, Timer 2 is set up to function as a counter. It creates periodic 1-second intervals that toggle an LED connected to pin PC13. To achieve a 1-second overflow, the prescaler and auto-reload value (ARR) are chosen based on the microcontroller's clock frequency which is 8M Hz for HSI.

DESIGN AND TEST PROCEDURES

Design Steps

1. **GPIO Clock Configuration:**
   - The system clock was chosen as HSI.
2. **GPIO Pin Configuration:**
   - GPIOC's Pin 13 is configured as an output, with a maximum speed of 2 MHz.
3. **Timer2 Configuration:**
   - Timer2 is set up to produce periodic interrupts. The prescaler and auto-reload value are chosen to set a time base of approximately 1 second.

Code Explanation

1. System Clock Initialization

For system clock configuration HSI was enabled and waited for it to ready. The system clock was arranged as HSI.

```
void SystemClock_Config(void) {
    RCC->CR |= RCC_CR_HSION; // Enable HSI
    while (!(RCC->CR & RCC_CR_HSIRDY)); // Wait until HSI is ready

    RCC->CFGR &= ~RCC_CFGR_SW;      // Clear SW bits
    RCC->CFGR |= RCC_CFGR_SW_HSI;   // Select HSI as system clock
    while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_HSI); // Wait for HSI
}
```

2. GPIO Configuration

- The clocks for GPIO C was enabled.

```
void GPIO_Init(void) {
    RCC->APB2ENR |= RCC_APB2ENR_IOPCEN; // Enable GPIOC Clock
```

- PC13 was set to "Output mode, max speed 2 MHz" via the GPIOC->CRH register:

```
    GPIOC->CRH &= ~GPIO_CRH_MODE13; // Clear mode
    GPIOC->CRH |= GPIO_CRH_MODE13_0; // Output mode 2 MHz
    GPIOC->CRH &= ~GPIO_CRH_CNF13; // Push-pull
}
```

3. Timer2 Initialization

- The clock for Timer2 is enabled by setting bit in the RCC_APB1ENR register.

```
void TIM2_Init(void) {
  RCC->APB1ENR |= RCC_APB1ENR_TIM2EN; // Enable Timer2 Clock
```

- The Prescaler (PSC) is adjusted to 999, dividing the clock frequency by 1000. The Auto-Reload Register is set to 7999, to create a time base of approximately 1 second. Since HSI 8 MHz, this results in a toggle every 1 second.

```
TIM2->PSC = 999;
TIM2->ARR = 8000-1;
```

4. Blinking LED

- The program enters an infinite loop, checking the UIF flag in the TIM2_SR register.

```
while (1) {
        // Wait for Timer2 overflow
        if (TIM2->SR & TIM_SR_UIF) { // Check update interrupt flag (UIF)
```

- When the timer overflows (UIF = 1), it toggles PC13.

```
GPIOC->ODR ^= GPIO_ODR_ODR13; // Toggle LED state
```

Full Code

```
#include "stm32f10x.h" // Include CMSIS library for STM32F1

/* Function Prototypes */
void SystemClock_Config(void);
void GPIO_Init(void);
void TIM2_Init(void);

/* Main Function */
int main(void) {
   /* Configure System Clock */
   SystemClock_Config();

   /* Initialize GPIO and Timer */
   GPIO_Init();
   TIM2_Init();

   /* Start Timer2 */
   TIM2->CR1 |= TIM_CR1_CEN; // Enable Timer2

   /* Infinite loop */
   while (1) {
      // Wait for Timer2 overflow
      if (TIM2->SR & TIM_SR_UIF) { // Check update interrupt flag (UIF)
         TIM2->SR &= ~TIM_SR_UIF; // Clear UIF flag

         // Toggle LED on PC13
         GPIOC->ODR ^= GPIO_ODR_ODR13;
      }
   }
}
/* System Clock Configuration */
void SystemClock_Config(void) {
   RCC->CR |= RCC_CR_HSION; // Enable HSI
   while (!(RCC->CR & RCC_CR_HSIRDY)); // Wait until HSI is ready

   RCC->CFGR &= ~RCC_CFGR_SW;      // Clear SW bits
   RCC->CFGR |= RCC_CFGR_SW_HSI;   // Select HSI as system clock
   while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_HSI); // Wait for HSI
}


/* GPIO Initialization */
void GPIO_Init(void) {
   /* Enable GPIOC Clock */
   RCC->APB2ENR |= RCC_APB2ENR_IOPCEN;

   /* Configure PC13 as output for LED */
   GPIOC->CRH &= ~GPIO_CRH_MODE13; // Clear mode
   GPIOC->CRH |= GPIO_CRH_MODE13_0; // Output mode 2 MHz
   GPIOC->CRH &= ~GPIO_CRH_CNF13; // Push-pull
}
```

```
/* Timer2 Initialization */
void TIM2_Init(void) {
   /* Enable Timer2 Clock */
   RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;

   /* Configure Timer2 */
   TIM2->PSC = 999;
   TIM2->ARR = 8000-1;
   TIM2->CR1 |= TIM_CR1_ARPE; // Enable auto-reload preload
}
```

RESULTS AND DISCUSSION

PC13’s LED blinks at a frequency of 1 Hz, indicating that Timer 2 was correctly set up with a prescaler of 999 and an auto-reload value of 7999. This configuration caused a timer overflow every 8000 clock cycles and as HSI is 8 MHz clock, it resulted in a 1-second interval for the LED to toggle.It was an easy lab experiment. The task was generally about utilizing what we had learned.

CONCLUSIONS

This lab successfully demonstrates how to toggle an LED using Timer 2. The timer's overflow event is used to toggle an LED connected to PC13. The use of hardware peripherals, such as timers, provides precise timing.

REFERENCES

[1] STMicroelectronics, "STM32F10xxx Reference Manual," [Online]. Available: https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-and-stm32f105xx-stm32f107xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf. Accessed: Oct. 14, 2024