GEBZE TECHNICAL UNIVERSITY

ELECTRONIC ENGINEERING

ELEC335

MICROPROCESSORS LAB

LAB 5 Report

Prepared by
171024086 Berat KIZILARMUT

1. Introduction

In this lab, we will learn about setting up UART protocol and transmitting and receiving datas from it. We will experiment with Pulse Width Modulation and display it on an external LED. Then we will connect a keypad to use the keypad to modulate the led to our own desire.

2. Problem 1

In this problem we will initialize the board with UART protocol and will create input and output functions. Starting from the UART initialization function, we start by clocking A pins. After that enable USART2 by setting bit 17 on APBENR1, APB peripheral clock enable register. USART pins on this board are available on PA2/PA3, PA9/PA10 or PC10/PC11. Chose PA2/PA3 pair to use. We have to set these pins on GPIOA MODER as Alternate function. Then chose AF1 on alternate function mux because these are the modes for USART2_TX for PA2 and USART2_RX for PA3, retrieved from Section 4, Table 13 on STM32G031 datasheet. After setting the pins, we set Transmitter Enable (bit 3), Receiver Enable (bit 2) and USART Enable (bit 0) as 1 on Control Register, CR1. Prior to setting UE pin as one, we have to set the USART baud rate register as we wish but we cannot set it after we enable it. After these set up phase of the UART is completed.

Now looking at the data transmission and receive functions, starting from the transmission function, uart_tx; we write the input data to transmit data register, USART_TDR. While loop waits till the transmission is complete, then sets Transmission Complete bit as 1 on USART interrupt and status register. Now to the receiving of data function, uart_rx. On this function we read the data from USART receive data register and return it.

```c
/* main.c Problem 1, UART Protocol
    Berat Kizilarmut */
#include "stm32g0xx.h"
static volatile uint32_t tick = 0;

void SysTick_Handler(void) /* Overriding Systick Handler */
{
    if(tick > 0)
    {
        --tick;
    }
}

void BSP_UART_init(uint32_t baud){ /*Initializing UART */
    RCC->IOPENR |= (1U << 0); /* Enable A */
    RCC->APBENR1 |= (1U << 17); /*Enable USART2 */
    /* Setup PA2 as Alternate function mode */
    GPIOA->MODER &= ~(3U << 2*2);
    GPIOA->MODER |= (2U << 2*2);
    /* Choose AF1 from the Mux */
    GPIOA->AFR[0] &= ~(0xFU << 4*2);
    GPIOA->AFR[0] |= (1U << 4*2);
    /* Setup PA3 as Alternate function mode */
    GPIOA->MODER &= ~(3U << 2*3);
    GPIOA->MODER |= (2U << 2*3);
    /* Choose AF1 from the Mux */
    GPIOA->AFR[0] &= ~(0xFU << 4*3);
    GPIOA->AFR[0] |= (1U << 4*3);
```

```
    /* Setup UART2 */
    /* Reset UART2 CR1 */
    USART2->CR1 = 0;
    USART2->CR1 |= (1U << 3); /* TE */
    USART2->CR1 |= (1U << 2); /* RE */
    USART2->BRR = (uint16_t)(SystemCoreClock / baud);
    USART2->CR1 |= (1U << 0); /* UE */

}

void uart_tx(uint8_t c){ /* Setting Transmission of Data */
    USART2->TDR = (uint16_t)c; /* Transmit Data Register */
    while (!(USART2->ISR & (1U << 6)));
}

uint8_t uart_rx(void){ /* Setting Receiving of Data */
    uint8_t data = (uint8_t)USART2->RDR;
    return data;
}

int main(void) {
    SysTick_Config(SystemCoreClock / 1000); /* Setting systick timing */
    BSP_UART_init(9600); /* Calling the UART initializer */

    while(1) {
        uart_tx(uart_rx()); /* Continuous data stream */
     } /* Endless loop */
    return 0;
}
```

3. Problem 2

Firstly we will create a Pulse Width Modulation function and we will need a pin capable of timer functionality. Looking at that datahsheet pinout, I've chosen PA6 because it has timer functionality on its alternate function 1. PA6 has Timer 3, Channel 1. Created a timer initialization function that sets the timer3.

```
/* main.c Problem 2, Pulse Width Modulation
    Berat Kizilarmut */
#include "stm32g0xx.h"

void init_timer3(void) /* Setting TIM3 */
{
    RCC->APBENR1 |= (1U << 1); /* Enable TIM3 Clock */
    TIM3->CCMR1 = 1;
    TIM3->CR1 = 0; /* Clearing the control register */
    TIM3->CR1 |= (1U << 7);  /* Auto Reload Preload Enable */
    TIM3->CNT = 0; /* Zero the counter */
    TIM3->PSC = 7999;
    TIM3->ARR = 1;
    TIM3->DIER |= (1U << 0); /* Updating interrupt enabler */
    TIM3->CR1 |= (1U << 0); /* Enable TIM3 */
    NVIC_SetPriority(TIM3_BRK_UP_TRG_COM_IRQn, 3); /* Setting lowest priority */
    NVIC_EnableIRQ(TIM3_BRK_UP_TRG_COM_IRQn); /* Enabling interrupt */
}

int main(void) {
    /* Enable GPIOA clock */
    RCC->IOPENR |= (1U);
    /* Setup PA6 as Alternate function mode */
    GPIOA->MODER &= ~(3U << 2*6);
    GPIOA->MODER |= (2U << 2*6);
```

```
    /* Choose AF1 from the Mux */
    GPIOA->AFR[0] &= ~(0xFU << 4*6);
    GPIOA->AFR[0] |= (1U << 4*6);
    init_timer3;
    while(1) { } /* Endless loop */
    return 0;
}
```

4. References

1. STM32G031x4/x6/x8 DS12992 Rev2, ST Microelectronics