# Assignment 7 Report

EE615 : Embedded Systems Lab

Atharv Gade (210020004)
Yash Meshram (210020053)
10 October 2024

# Aim:

**1.** To write a program for the Tiva microcontroller to send the 8-bit value "0xF0" when SW1 is pressed and "0xAA" when SW2 is pressed via UART, with a baud rate of 9600 and odd parity.

**2.** Draw the expected UART waveforms for both cases, including indicative timing details.

**3.** Connect an oscilloscope to the TX pin and verify that the captured signals align with the waveforms in part 2.

**4.** Modify the program to listen for incoming UART data using the same baud rate and parity configuration. If "0xAA" is received, the LED should turn GREEN; if "0xF0" is received, the LED should turn BLUE; and if any error is detected, the LED should turn RED. Test this by communicating with a neighbouring group, ensuring that the RX of one board is connected to the TX of the other, and that both board grounds are properly connected. Press and decrease for a long press.

# Theory:

The program is divided into two main parts: one for transmitting data and the other for receiving it. Transmission occurs in the transmit function, where the Data Register is loaded with the data to be sent (this data is passed as a parameter to the UART-Transmit function). GPIO interrupts are enabled, and the two switches are configured to trigger the UART-Transmit function, sending their respective data values.

UART reception is always active, and the UART-Receive function is called within a continuous **while(1)** loop. In the receive function, the receive status is checked, and based on the data present in the UART Data Register, a specific LED is turned ON. For testing, the received data is also sent back via the transmit pin, simulating a loopback.

An additional UART interrupt handler is included, which is triggered whenever a transmission or reception error occurs. In this handler, the RED LED is turned ON.

# Code :

```
#include <stdint.h>
#include <stdbool.h>
#include "tm4c123gh6pm.h"


uint8_t message = 0x00;
uint8_t PORTF_Interrupt = 0x00;

void GPIOPortF_Handler();

int main(void)
{
    SYSCTL_RCGC2_R |= 0x00000020;

    // GPIO_PORTF_REGISTERS
    GPIO_PORTF_LOCK_R = 0x4C4F434B;
    GPIO_PORTF_CR_R = 0x1F;
    GPIO_PORTF_DEN_R = 0x1F;
    GPIO_PORTF_DIR_R = 0x0E;
    GPIO_PORTF_PUR_R = 0x11;

    NVIC_EN0_R = 0x40000000;
    GPIO_PORTF_IS_R = 0x00;
    GPIO_PORTF_IEV_R = 0x00;
    GPIO_PORTF_IM_R = 0x10;

    // enabling to UART module 1
    SYSCTL_RCGCUART_R |= 0x02;
    SYSCTL_RCGCGPIO_R |= 0x22;

    // enabling to PORTB registers
    GPIO_PORTB_LOCK_R = 0x4C4F434B;
    GPIO_PORTB_CR_R = 0x03;
    GPIO_PORTB_DEN_R = 0x03;
    GPIO_PORTB_AFSEL_R = 0x03;
    GPIO_PORTB_PCTL_R = 0x11;
    GPIO_PORTB_DIR_R = 0x02;
    GPIO_PORTB_PUR_R = 0x02;



    // UART module 1 registers
    UART1_IBRD_R = 104;
    UART1_FBRD_R = 11;
    UART1_LCRH_R |= 0x62;
    UART1_CC_R = 0x00;
    UART1_CTL_R |= 0x01;
    uint8_t rx_reg = 0x00;

    while(1){
```

```
        NVIC_EN0_R = 0x40000000; // 30th bit controls PORTF GPIO
interrupts
        GPIO_PORTF_IM_R = 0x11;

        UART1_DR_R = message;
        while (UART1_FR_R & 0x08){
            ;
        }
        rx_reg = UART1_DR_R & 0xFFF;
        if ((rx_reg & 0xFF) == 0xF0){
            GPIO_PORTF_DATA_R &= 0x04;
            GPIO_PORTF_DATA_R ^= 0x04;
            delay(500000);
        }
        else if ((rx_reg & 0xFF) == 0xAA){
            GPIO_PORTF_DATA_R &= 0x08;
            GPIO_PORTF_DATA_R ^= 0x08;
            delay(500000);
        }

        else if (rx_reg & 0xF00){
            GPIO_PORTF_DATA_R &= 0x02;
            GPIO_PORTF_DATA_R ^= 0x02;
        }

        else{
            GPIO_PORTF_DATA_R &= 0x02;
            GPIO_PORTF_DATA_R ^= 0x02;
        }

    }

      return 0;
}

void delay(int us){
    NVIC_ST_RELOAD_R = 16*us;
    NVIC_ST_CTRL_R = 0x00000005;
    while( (NVIC_ST_CTRL_R & 0x00010000) != 0x00010000 ){;}
    NVIC_ST_CTRL_R = 0x00000000;
}



void GPIOPortF_Handler(void){
    PORTF_Interrupt = GPIO_PORTF_RIS_R & 0x11;


    NVIC_EN0_R = 0x00000000;
    GPIO_PORTF_IM_R = 0x00;
    if (PORTF_Interrupt == 0x01){
        GPIO_PORTF_ICR_R = 0x11;
        message = 0xAA;
    }
    if (PORTF_Interrupt == 0x10){
        GPIO_PORTF_ICR_R = 0x11;
        message = 0xF0;
    }

}
```

## Conclusion:

We successfully established communication between two boards using the correct transmission and reception settings. The waveform clearly displayed the expected start and stop bits. Since we were only transmitting one byte or character at a time, we kept the FIFO disabled.