Ismael Elsharkawi 900173030

Embedded Systems Lab
Lab 3 Report

Experiment 1:

```
#include <stdint.h>
#include "tm4c123gh6pm.h"
void UART0Tx(char const c);
void delayMs(int n);
int main(void)
{
    SYSCTL->RCGCUART |= 1; /* provide clock to UART0 */
    SYSCTL->RCGCGPIO |= 1; /* enable clock to PORTA */

    /* UART0 initialization */
    UART0->CTL = 0; /* disable UART0 */
    UART0->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
    UART0->FBRD = 11; /* fraction part = 0.1666666*64+0.5 = 11.1666666 */
    UART0->CC = 0; /* use system clock */
    UART0->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
    UART0->CTL = 0x301; /* enable UART0(bit0), TXE(bit8), RXE(bit9) */
    /* UART0 TX0 and RX0 use PA1 and PA0. Set them up. */
    GPIOA->DEN = 0x03; /* Make PA0 and PA1 as digital */
    GPIOA->AFSEL = 0x03; /* Use PA0,PA1 alternate function */
    GPIOA->PCTL = 0x11; /* configure PA0 and PA1 for UART */

    delayMs(25); /* wait for output line to stabilize */

    for(;;)
    {
        UART0Tx('Y');
        UART0Tx('e');
        UART0Tx('s');
        UART0Tx(' ');
    }
}
/* UART0 Transmit */
```

```
/* This function waits until the transmit buffer is available then writes
*/
/* the character in it. It does not wait for transmission to complete */
void UART0Tx(char const c)
{
    while((UART0->FR & 0x20)!= 0){} // Wait until Tx buffer is not full
    UART0->DR = c; // Write byte
}

void delayMs(int delay)
{
    SysTick->LOAD = 16000*delay -1;
    SysTick->CTRL = 0x5; /*Enable the timer and choose sysclk */
    while((SysTick->CTRL & 0x10000) == 0) /*wait until the Count flag is
set */
    { }
    SysTick->CTRL = 0; /*Stop the timer (Enable = 0) */
}
```

Experiment 2:

```
#include <stdint.h>
#include "tm4c123gh6pm.h"
char UART0Rx(void);
void delayMs(int n);
int main(void)
{
 char c;
 SYSCTL->RCGCUART |= 1; /* provide clock to UART0 */
 SYSCTL->RCGCGPIO |= 1; /* enable clock to PORTA */
 SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */

 /* UART0 initialization */
 UART0->CTL = 0; /* disable UART0 */
 UART0->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
 UART0->FBRD = 11; /* fraction part= 0.1666666*64+0.5 = 11.1666666 */
 UART0->CC = 0; /* use system clock */
```

```
 UART0->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
 UART0->CTL = 0x301; /* enable UART0, TXE, RXE */
 /* UART0 TX0 and RX0 use PA1 and PA0. Set them up. */
 GPIOA->DEN = 0x03; /* Make PA0 and PA1 as digital */
 GPIOA->AFSEL = 0x03; /* Use PA0,PA1 alternate function */
 GPIOA->PCTL = 0x11; /* configure PA0 and PA1 for UART */

 GPIOF->DIR = 0x0E; /* configure PortF pins 3,2,1 to control LEDs */
 GPIOF->DEN = 0x0E;
 GPIOF->DATA = 0; /* turn LEDs off */
 for(;;){
 c = UART0Rx(); /* get a character from UART */
 GPIOF->DATA = c << 1; /* shift left & write least sig. 3 bits to LEDs */
 }
}
/* UART0 Receive */
/* This function waits until a character is received then returns it. */
char UART0Rx(void)
{
 char c;
 while((UART0->FR & 0x10) != 0){} /* wait until Rx buffer is not empty */
 c = UART0->DR; /* read the received data */
 return c; /* and return it */
}

void delayMs(int delay)
{
    SysTick->LOAD = 16000*delay -1;
    SysTick->CTRL = 0x5; /*Enable the timer and choose sysclk */
    while((SysTick->CTRL & 0x10000) == 0) /*wait until the Count flag is
set */
    { }
    SysTick->CTRL = 0; /*Stop the timer (Enable = 0) */
}
```

Experiment 3:

```
/* Read data from UART0 and display it at the tri-color LEDs. The LEDs are
connected to Port F 3-1. Press any A-z, a-z, 0-9 key at the terminal
emulator
and see ASCII value in binary is displayed on LEDs of PORTF. */
#include "tm4c123gh6pm.h"
int main(void)
{
 SYSCTL->RCGCUART |= 1; /* provide clock to UART0 */
 SYSCTL->RCGCGPIO |= 1; /* enable clock to PORTA */
 SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */

 /* UART0 initialization */
 UART0->CTL = 0; /* disable UART0 */
 UART0->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
 UART0->FBRD = 11; /* fraction part= 0.1666666*64+0.5 = 11.1666666 */
 UART0->CC = 0; /* use system clock */
 UART0->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
 UART0->IM |= 0x0010; /* enable RX interrupt */
 UART0->CTL = 0x301; /* enable UART0, TXE, RXE */
 /* UART0 TX0 and RX0 use PA1 and PA0. Set them up. */
 GPIOA->DEN = 0x03; /* Make PA0 and PA1 as digital */
 GPIOA->AFSEL = 0x03; /* Use PA0,PA1 alternate function */
 GPIOA->PCTL = 0x11; /* configure PA0 and PA1 for UART */

 GPIOF->DIR = 0x0E; /* configure Port F to control the LEDs */
 GPIOF->DEN = 0x0E;
 GPIOF->DATA = 0; /* turn LEDs off */
 /* enable interrupt in NVIC and set priority to 3 */
 NVIC->IP[5] = 3 << 5; /* set interrupt no 5 priority to 3 */
 NVIC->ISER[0] |= 0x00000020; /* enable IRQ5 for UART0 */
 __enable_irq(); /* global enable IRQs */
 for(;;){}
}
void UART0_Handler(void)
{
```

```
 volatile int readback;
 char c;
 if (UART0->MIS & 0x0010) /* if a receive interrupt has occurred */
 {
 c = UART0->DR; /* read the received data */
 GPIOF->DATA = c << 1; /* shift left and write it to LEDs */
 UART0->ICR = 0x0010; /* clear Rx interrupt flag */
 readback = UART0->ICR; /* a read to force clearing of interrupt flag */
 }
 else
 {
 /* should not get here. But if it does, */
 UART0->ICR = UART0->MIS; /* clear all interrupt flags */
 readback = UART0->ICR; /* a read to force clearing of interrupt flag */
 }
}
// Remember to disable IRQs in SystemInit function in generated startup C
src.
```

Experiment 4:

```
/* Read data from UART0 and display it at the tri-color LEDs. The LEDs are
connected to Port F 3-1. Press any A-z, a-z, 0-9 key at the terminal
emulator
and see ASCII value in binary is displayed on LEDs of PORTF. */
#include "tm4c123gh6pm.h"
int main(void)
{
 SYSCTL->RCGCUART |= 1; /* provide clock to UART0 */
 SYSCTL->RCGCGPIO |= 1; /* enable clock to PORTA */
 SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */

 /* UART0 initialization */
 UART0->CTL = 0; /* disable UART0 */
 UART0->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
 UART0->FBRD = 11; /* fraction part= 0.1666666*64+0.5 = 11.1666666 */
```

```
 UART0->CC = 0; /* use system clock */
 UART0->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
 UART0->IM |= 0x0010; /* enable RX interrupt */
 UART0->CTL = 0x301; /* enable UART0, TXE, RXE */
 /* UART0 TX0 and RX0 use PA1 and PA0. Set them up. */
 GPIOA->DEN = 0x03; /* Make PA0 and PA1 as digital */
 GPIOA->AFSEL = 0x03; /* Use PA0,PA1 alternate function */
 GPIOA->PCTL = 0x11; /* configure PA0 and PA1 for UART */

 GPIOF->DIR = 0x0E; /* configure Port F to control the LEDs */
 GPIOF->DEN = 0x0E;
 GPIOF->DATA = 0; /* turn LEDs off */
 /* enable interrupt in NVIC and set priority to 3 */
 NVIC->IP[5] = 3 << 5; /* set interrupt no 5 priority to 3 */
 NVIC->ISER[0] |= 0x00000020; /* enable IRQ5 for UART0 */
 __enable_irq(); /* global enable IRQs */
 for(;;){}
}
void UART0_Handler(void)
{
 volatile int readback;
 char c;
 if (UART0->MIS & 0x0010) /* if a receive interrupt has occurred */
 {
 c = UART0->DR; /* read the received data */
 GPIOF->DATA = c << 1; /* shift left and write it to LEDs */
 UART0->ICR = 0x0010; /* clear Rx interrupt flag */
 readback = UART0->ICR; /* a read to force clearing of interrupt flag */
 }
 else
 {
 /* should not get here. But if it does, */
 UART0->ICR = UART0->MIS; /* clear all interrupt flags */
 readback = UART0->ICR; /* a read to force clearing of interrupt flag */
 }
}
```

```
// Remember to disable IRQs in SystemInit function in generated startup C
src.
```

Experiment 5:

```
/* Receive characters from phone using UART1 and send it to PC using UART0
*/
#include "TM4C123GH6PM.h"
void UART0Tx(char c);
int main(void)
{
    SYSCTL->RCGCUART |= 3; /* provide clock to UART0 and UART1*/
    SYSCTL->RCGCGPIO |= 3; /* enable clock to PORTA and PORTB */
    SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */

    /* UART0 initialization */
    UART0->CTL = 0; /* disable UART0 */
    UART0->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
    UART0->FBRD = 11; /* fraction part= 0.1666666*64+0.5 = 11.1666666 */
    UART0->CC = 0; /* use system clock */
    UART0->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
    UART0->IM |= 0x0010; /* enable RX interrupt */
    UART0->CTL = 0x301; /* enable UART0, TXE, RXE */
    /* UART1 initialization */
    UART1->CTL = 0; /* disable UART1 */
    UART1->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
    UART1->FBRD = 11; /* fraction part= 0.1666666*64+0.5 = 11.1666666 */
    UART1->CC = 0; /* use system clock */
    UART1->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
    UART1->IM |= 0x0010; /* enable RX interrupt */
    UART1->CTL = 0x301; /* enable UART1, TXE, RXE */
    /* UART0 TX0 and RX0 use PA1 and PA0. Set them up. */
    GPIOA->DEN = 0x03; /* Make PA0 and PA1 as digital */
    GPIOA->AFSEL = 0x03; /* Use PA0,PA1 alternate function */
    GPIOA->PCTL = 0x11; /* configure PA0 and PA1 for UART */
```

```
    GPIOF->DIR = 0x0E; /* configure Port F to control the LEDs */
    GPIOF->DEN = 0x0E;
    GPIOF->DATA = 0; /* turn LEDs off */
    /* UART1 TX0 and RX0 use PB1 and PB0. Set them up. */
    GPIOB->DEN = 0x03; /* Make PB0 and PB1 as digital */
    GPIOB->AFSEL = 0x03; /* Use PB0,PB1 alternate function */
    GPIOB->PCTL = 0x11; /* configure PB0 and PB1 for UART */

    /* enable interrupt in NVIC and set priority to 3 */
    NVIC->IP[6] = 3 << 5; /* set interrupt no 6 priority to 3 */
    NVIC->ISER[0] |= 0x00000040; /* enable IRQ6 for UART1 */
     __enable_irq(); /* global enable IRQs */
/* provide clock to UART0 */

/* enable clock to PORTA */
/* provide clock to UART1 */
/* enable clock to PORTB */
/* enable clock to PORTF */
/* UART0 initialization */
/* UART0 TX0 and RX0 use PA1 and PA0. Set them up. */
/* UART1 initialization, enabling RX interrupt */
/* UART1 TX0 and RX0 use PB1 and PB0. Set them up. */
/* configure Port F pins 3,2,1 to control the LEDs */
/* enable UART1 interrupt in NVIC and set priority to 3 */
/* global enable IRQs */
 while (1) {}
}
void UART1_Handler(void)
{
 volatile int readback;
 char c;
 if (UART1->MIS & 0x0010) /* if a receive interrupt has occurred */
 {
     c = UART1->DR; /* read the received data */
     if(c=='r') GPIOF->DATA = 2;
     if(c=='g') GPIOF->DATA = 8;
     if(c=='b') GPIOF->DATA = 4;
```

```
      //GPIOF->DATA = c << 1; /* shift left and write it to LEDs */

      UART1->ICR = 0x0010; /* clear Rx interrupt flag */
      readback = UART1->ICR; /* a read to force clearing of interrupt flag
*/
      UART0Tx(c);
  }
  else
  {
      /* should not get here. But if it does, */
      UART1->ICR = UART0->MIS; /* clear all interrupt flags */
      readback = UART1->ICR; /* a read to force clearing of interrupt flag
*/
      }
}
void UART0Tx(char c)
{

  //while((UART0->FR & 0x10) != 0){} /* wait until Rx buffer is not empty
*/
  UART0->DR = c; /* read the received data */

}
```

Question 1:

https://youtu.be/CDEp76nrFWw

```
/* Receive characters from phone using UART1 and send it to PC using UART0 */
#include "TM4C123GH6PM.h"
char volatile buffer[1024];
int volatile i = 0;
int volatile flag =0;
void UART0Tx(char c);
int main(void)
{
        SYSCTL->RCGCUART |= 3; /* provide clock to UART0 and UART1*/
        SYSCTL->RCGCGPIO |= 3; /* enable clock to PORTA and PORTB */
        SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */
```

```
        /* UART0 initialization */
        UART0->CTL = 0; /* disable UART0 */
        UART0->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
        UART0->FBRD = 11; /* fraction part= 0.1666666*64+0.5 = 11.1666666 */
        UART0->CC = 0; /* use system clock */
        UART0->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
        UART0->IM |= 0x0010; /* enable RX interrupt */
        UART0->CTL = 0x301; /* enable UART0, TXE, RXE */
        /* UART1 initialization */
        UART1->CTL = 0; /* disable UART1 */
        UART1->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
        UART1->FBRD = 11; /* fraction part= 0.1666666*64+0.5 = 11.1666666 */
        UART1->CC = 0; /* use system clock */
        UART1->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
        UART1->IM |= 0x0010; /* enable RX interrupt */
        UART1->CTL = 0x301; /* enable UART1, TXE, RXE */
        /* UART0 TX0 and RX0 use PA1 and PA0. Set them up. */
        GPIOA->DEN = 0x03; /* Make PA0 and PA1 as digital */
        GPIOA->AFSEL = 0x03; /* Use PA0,PA1 alternate function */
        GPIOA->PCTL = 0x11; /* configure PA0 and PA1 for UART */

        GPIOF->DIR = 0x0E; /* configure Port F to control the LEDs */
        GPIOF->DEN = 0x0E;
        GPIOF->DATA = 0; /* turn LEDs off */
        /* UART1 TX0 and RX0 use PB1 and PB0. Set them up. */
        GPIOB->DEN = 0x03; /* Make PB0 and PB1 as digital */
        GPIOB->AFSEL = 0x03; /* Use PB0,PB1 alternate function */
        GPIOB->PCTL = 0x11; /* configure PB0 and PB1 for UART */

        /* enable interrupt in NVIC and set priority to 3 */
        NVIC->IP[6] = 3 << 5; /* set interrupt no 6 priority to 3 */
        NVIC->ISER[0] |= 0x00000040; /* enable IRQ6 for UART1 */
        __enable_irq(); /* global enable IRQs */

while (1) {
        while(flag==0){}
        __disable_irq();
         if(buffer[0] =='A' && buffer[1] == 'T' && buffer[2] == '+'){
                        GPIOF->DATA=0;
                        if(buffer[3]=='R'){
```

```
                        if(buffer[5]=='0') GPIOF->DATA  &= ~2;
                        else GPIOF->DATA |= 2;
                }
                if(buffer[3]=='G'){
                        if(buffer[5]=='0') GPIOF->DATA  &= ~8;
                        else GPIOF->DATA |= 8;
                }
                if(buffer[3]=='B'){
                        if(buffer[5]=='0') GPIOF->DATA  &= ~4;
                        else GPIOF->DATA |= 4;
                }
                if(buffer[3]=='W'){
                        if(buffer[5]=='0') GPIOF->DATA  &= ~14;
                        else GPIOF->DATA |= 14;
                }
                if(buffer[3]=='O' && buffer[4] == 'F' && buffer[5] == 'F'){
                        GPIOF->DATA =0;
                }

        }
        i=0;
        flag =0;
        __enable_irq();


}
}
void UART1_Handler(void)
{
 volatile int readback;
 char c;

 if (UART1->MIS & 0x0010) /* if a receive interrupt has occurred */
 {

        c = UART1->DR; /* read the received data */
        if(c=='\n' || i>=6){
                        i=0;
                        flag =1;
        }else{
                buffer[i] = c;
```

```
                i++;
        }
        UART1->ICR = 0x0010; /* clear Rx interrupt flag */
        readback = UART1->ICR; /* a read to force clearing of interrupt flag */
        UART0Tx(c);
}
 else
{
        /* should not get here. But if it does, */
        UART1->ICR = UART0->MIS; /* clear all interrupt flags */
        readback = UART1->ICR; /* a read to force clearing of interrupt flag */
        }
}
void UART0Tx(char c)
{
 UART0->DR = c; /* read the received data */
}
```

Question 2:

https://youtu.be/fHcBbbDDG2k

```
/* Receive characters from phone using UART1 and send it to PC using UART0 */
#include "TM4C123GH6PM.h"
#include <stdio.h>
char  buffer[8];
char n2[] = {'T','(' ,'M', 's', ')', ':'};
char nl[] = {'\r', '\n'};
int volatile i = 0;
int volatile counter = 0;
int volatile flag =0;
void UART0Tx(char c);
int main(void)
{
        __disable_irq();
        SYSCTL->RCGCUART |= 3; /* provide clock to UART0 and UART1*/
        SYSCTL->RCGCGPIO |= 3; /* enable clock to PORTA and PORTB */
        SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */

        /* UART0 initialization */
```

```
        UART0->CTL = 0; /* disable UART0 */
        UART0->IBRD = 104; /* 16MHz/(16*9600 baud rate) = 104.1666666666 */
        UART0->FBRD = 11; /* fraction part= 0.1666666*64+0.5 = 11.1666666 */
        UART0->CC = 0; /* use system clock */
        UART0->LCRH = 0x60; /* 8-bit, no parity, 1-stop bit, no FIFO */
        UART0->IM |= 0x0010; /* enable RX interrupt */
        UART0->CTL = 0x301; /* enable UART0, TXE, RXE */

        /* UART0 TX0 and RX0 use PA1 and PA0. Set them up. */
        GPIOA->DEN = 0x03; /* Make PA0 and PA1 as digital */
        GPIOA->AFSEL = 0x03; /* Use PA0,PA1 alternate function */
        GPIOA->PCTL = 0x11; /* configure PA0 and PA1 for UART */
                        SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */
                        /* PORTF0 has special function, need to unlock to modify */
                        GPIOF->LOCK = 0x4C4F434B; /* unlock commit register */
                        GPIOF->CR = 0x01; /* make PORTF0 configurable */
                        GPIOF->LOCK = 0; /* lock commit register */
                        /* configure PORTF for switch input and LED output */
                        GPIOF->DIR &= ~0x11; /* make PORTF4,0 input for switch */
                        GPIOF->DIR |= 0x0E; /* make PORTF3, 2, 1 output for LEDs */
                        GPIOF->DEN |= 0x1F; /* make PORTF4-0 digital pins */
                        GPIOF->PUR |= 0x11; /* enable pull up for PORTF4,0 */

                        /* configure PORTF4, 0 for falling edge trigger interrupt */
                        GPIOF->IS &= ~0x11; /* make bit 4, 0 edge sensitive */
                        GPIOF->IBE |= 0x11; /* trigger is controlled by IEV */
                        GPIOF->ICR |= 0x11; /* clear any prior interrupt */
                        GPIOF->IM |= 0x11; /* unmask interrupt for PF4,PF0 */
                        /* enable interrupt in NVIC and set priority to 3 */
                        NVIC->IP[30] = 3 << 5; /* set interrupt priority to 3 */
                        NVIC->ISER[0] |= 0x40000000; /* enable IRQ30 (D30 of ISER[0]) */
                        //SysTick->CTRL = 7;
                        //SysTick->LOAD = 16000;
        __enable_irq(); /* global enable IRQs */

 while(1){}
}

void UART0Tx(char c)
{
        int time_consumption =0;
```

```
        while((UART0->FR & 0x100) != 0){} /* wait until Rx buffer is not empty */
        UART0->DR= c;
        while(time_consumption<10000){ time_consumption++;}
}

void GPIOF_Handler(void)
{
        int n;
        int i;

        /*if(flag ==0){
                flag =1;
                counter =0;
                SysTick->CTRL = 7;
                SysTick->LOAD = 16000;
        }else{
                flag =0;
                SysTick->CTRL=0;
                UART0Tx(counter);
                counter=0;
        }*/
if (GPIOF->MIS & 0x0001) /* if a receive interrupt has occurred */
        {
         if((GPIOF->DATA & 0x1)==0){
                 //counter =0;
                 SysTick->LOAD = 16000-1;
                 SysTick->CTRL = 7;


         }else{
                 if(counter>0){
                         SysTick->CTRL=0;
                                 for (i =0; i< 6; i++) UART0Tx(n2[i]);
                          n = sprintf(buffer, "%d", counter);

                                 for(i = 0; i < n; i++){
                                         UART0Tx(buffer[i]);
                                 }
                                 UART0Tx(nl[0]);
                                 UART0Tx(nl[1]);
```

```
                        counter=0;
                        }
        }
                GPIOF->RIS |= 0x0001;
}
}
void SysTick_Handler(void)
{
        counter+=1;
}
```