Lab 06 – Worksheet

Name: Ehzem Farhan Sheikh	ID: es08518	Section: T2

Assumptions and logics should be explained separately in tasks after the task results.

Task 1.

Given that the clock frequency of the Tiva C board is 16MHz, determine the clock cycles value needed to generate delay for one second:

15,999,999

Provide your code here with appropriate comments below

```
#include "TM4C123GH6PM.h"
int main()
{
  SYSCTL->RCGCGPIO |= 0x20;          // turn on bus clock for GPIOF
  GPIOF->DIR       |= 0xE;           // set RBG pins as digital output pin
  GPIOF->DEN       |= 0xE;           // Enable PF1-3 pinw as digital pin

        SysTick->LOAD = (16000000)-1;        // one second delay relaod value
        SysTick->CTRL = 0x7 ;  // enable counter, interrupt and select system bus clock
        SysTick->VAL  = 0;     //initialize current value register

        while (1)
        {
                //do nothing here since we are using SysTick Interrupt Routine
        }
}


// This Interrupt Service Routine will execute after every one second
void SysTick_Handler(void)
{
   //Use GPIOF->DATA command and toggle PF1-3 pins below between HIGH and LOW
        GPIOF->DATA ^= 0xE;

        //GPIOF->DATA ^= GPIOF->DATA;
}
```

Task 2:

a Given that the clock frequency of the Tiva C board is 16MHz, determine the clock cycles needed to generate delay for one millisecond:

(i) 16000

b Calculate the clock cycles required to generate a delay of 5 seconds:

(ii) 3,199,999

c How many bits are required to hold the value calculated in (b)? Should we use the 16-bit mode or the 32-bit mode of the clock?

(iii) 32-bit mode of the clock

Task 1.

Provide your code here with appropriate comments below

```
#include<TM4C123.h>

void timer0_delay(int ms);

int main()
{
  SYSCTL->RCGCGPIO |= 0x20;          // turn on bus clock for GPIOF
  GPIOF->DIR       |= 0xE;           // set RBG pins as digital output pin
  GPIOF->DEN       |= 0xE;           // Enable PF1-3 pinw as digital pin

        while (1)
        {
                // implement Traffic Light sequence here
                // ...rest of the logic should go here for traffic system
                // implement Traffic Light sequence here

                //Turning on Red light
                GPIOF->DATA  ^= (1<<1);
                timer0_delay(1000);
                GPIOF->DATA  &= ~(1<<1); //Turning it off

                // Changing LED from Red to Green
                timer0_delay(2000);
                GPIOF->DATA  ^= (1<<3);
                timer0_delay(2000);
                GPIOF->DATA  &= ~(1<<3); // Turning it off

                //Changing LED from Green to yellow by turning on red and green
                timer0_delay(2000);
                GPIOF->DATA  ^= (1<<1);
                GPIOF->DATA  ^= (1<<3);
                timer0_delay(1000);

                // Turning everything off yellow by turning of red and green
                GPIOF->DATA  &= ~(1<<1);
                GPIOF->DATA  &= ~(1<<3);

                timer0_delay(1000);

        }
}

void timer0_delay(int ms) // ms is in milliseconds
{
        SYSCTL->RCGCTIMER |= (1<<0);  //enable clock for Timer 0

        TIMER0->CTL |= 0x0;           //disable timer 0 before initialization
        TIMER0->CFG = 0x0;            //select configuration for timer 0
        TIMER0->TAMR = 0x1;           //configure timer A for one shot mode
        TIMER0->TAILR = 16000*ms - 1; //timer A interval load value register
        TIMER0->ICR = 0x1;            //clear status flag
        TIMER0->CTL |= 0x1;           //enable timer 0 to start counting after initializtion
        while ((TIMER0->RIS & 0x1) == 0); //Wait for the GPTM Raw Interrupt Status flag to
set
}
```

Task 3:

a Write the last 3 digits of your HU ID (0xABC): (i) 0x518

b You will use the following duty cycles for the RGB channel of LED from Fig. 8:
RED : A0% (i) 50%
GREEN : B0% (ii) 10%
BLUE : C0% (iii) 80%

Task 1.

Provide your code here with appropriate comments below

```
#include<TM4C123.h>

void timer0_delay(int ms);

void timer1a_pwm(int duty_cycle);

void timer2a_pwm(int duty_cycle);

void timer3a_pwm(int duty_cycle);

int main()
{
	SYSCTL->RCGCTIMER |= 0x0F;    //enable clock for Timer 0-3
	SYSCTL->RCGCGPIO |= (1<<1);      /* enable clock to PORTB  */

	// Initialize Channel Timer 1A PWM to PB4
	GPIOB->DIR |= (1<<4);   /* set PB4 an output pin */
	GPIOB->DEN |= (1<<4);    /* set PB4 a digital pin */
	GPIOB->AFSEL |= (1<<4);  /* enable alternate function on PB4 */
	GPIOB->PCTL &= ~0x000F0000;  /* configure PB4 as T1CCP0 pin */
	GPIOB->PCTL |= 0x00070000;

	// Initialize Channel Timer 2A PWM to PB0
	GPIOB->DIR |= (1<<0);   /* set PB0 an output pin */
	GPIOB->DEN |= (1<<0);    /* set PB0 a digital pin */
	GPIOB->AFSEL |= (1<<0);  /* enable alternate function on PB0 */
	GPIOB->PCTL &= ~0x0000000F;  /* configure PB0 as T2CCP0 pin */
	GPIOB->PCTL |= 0x00000007;



	// Initialize Channel Timer 3A PWM to PB2

	GPIOB->DIR |= (1<<2);   /* set PB2 an output pin */
	GPIOB->DEN |= (1<<2);    /* set PB2 a digital pin */
	GPIOB->AFSEL |= (1<<2);  /* enable alternate function on PB2 */
	GPIOB->PCTL &= ~0x00000F00;  /* configure PB2 as T3CCP0 pin */
	GPIOB->PCTL |= 0x00000700;


	// Use PWM functions to output RGB Color spectrum using dutycycles calculated with
your ID

	// Last three digits of id are 518 (08518)
	timer1a_pwm(50);
	timer0_delay(1000);
	timer2a_pwm(100);
	timer0_delay(1000);
	timer3a_pwm(80);
	timer0_delay(1000);

	while (1)
	{

		// Do nothing OR Write a logic to toggle color spectrum using timer0 delay
	}
}

void timer1a_pwm(int duty_cycle) //duty cycle range: 0 - 100 %
{
	TIMER1->CTL |= 0x0;                                          //disable timer before
initialization
	TIMER1->CFG = 0x4;                                           //select 16-bit
configuration
	TIMER1->TAMR |= (1 << 3);                    //TAAMS set to 0x1 to enable PWM mode
	TIMER1->TAMR &= ~(1UL << 2);  //TACMR reset to 0 for Edge Count Mode
	TIMER1->TAMR |= (2 << 0);                    //TAMR set to 0x2 for periodic timer
mode
	TIMER1->TAILR = 16000 - 1;            // 1ms period | interval load value register
	TIMER1->TAMATCHR = 160 * duty_cycle;  // cut-off edge to control PWM duty cycle
	TIMER1->CTL |= 0x1;                                          //enable timer to start
counting after initializtion
}
```

Task 1.

```
void timer2a_pwm(int duty_cycle) //duty cycle range: 0 - 100 %
{
	//Initialize TIMER2 here
	TIMER2->CTL |= 0x0;                                          //disable timer before
initialization
	TIMER2->CFG = 0x4;                                           //select 16-bit
configuration
	TIMER2->TAMR |= (1 << 3);                    //TAAMS set to 0x1 to enable PWM mode
	TIMER2->TAMR &= ~(1UL << 2);  //TACMR reset to 0 for Edge Count Mode
	TIMER2->TAMR |= (2 << 0);                    //TAMR set to 0x2 for periodic timer
mode
	TIMER2->TAILR = 16000 - 1;           // 1ms period | interval load value register
	TIMER2->TAMATCHR = 160 * duty_cycle;  // cut-off edge to control PWM duty cycle
	TIMER2->CTL |= 0x1;                                          //enable timer to start
counting after initializtion
}

void timer3a_pwm(int duty_cycle) //duty cycle range: 0 - 100 %
{
	//Initialize TIMER3 here
	TIMER3->CTL |= 0x0;                                          //disable timer before
initialization
	TIMER3->CFG = 0x4;                                           //select 16-bit
configuration
	TIMER3->TAMR |= (1 << 3);                    //TAAMS set to 0x1 to enable PWM mode
	TIMER3->TAMR &= ~(1UL << 2);  //TACMR reset to 0 for Edge Count Mode
	TIMER3->TAMR |= (2 << 0);                    //TAMR set to 0x2 for periodic timer
mode
	TIMER3->TAILR = 16000 - 1;           // 1ms period | interval load value register
	TIMER3->TAMATCHR = 160 * duty_cycle;  // cut-off edge to control PWM duty cycle
	TIMER3->CTL |= 0x1;                                          //enable timer to start
counting after initializtion
}

void timer0_delay(int ms) // ms is in milliseconds
{
	SYSCTL->RCGCTIMER |= (1<<0);  //enable clock for Timer 0

	TIMER0->CTL |= 0x0;           //disable timer 0 before initialization
	TIMER0->CFG = 0x0;            //select configuration for timer 0
	TIMER0->TAMR = 0x1;           //configure timer A for one shot mode
	TIMER0->TAILR = 16000*ms - 1; //timer A interval load value register
	TIMER0->ICR = 0x1;            //clear status flag
	TIMER0->CTL |= 0x1;           //enable timer 0 to start counting after initializtion
	while ((TIMER0->RIS & 0x1) == 0); //Wait for the GPTM Raw Interrupt Status flag to
set
}
```

Task 4:

Provide your code here with appropriate comments below

```
/* This example code Measures the distance using HC-SR04 Ultrasonic range sensor*/
/* It displays the measured distance value on computer using UART communication moduel of
TM4C123 */
/* Timer0A is used to measure distance by measuring pulse duration of Echo output signal */
/* Timer1A is used to make percise microsecond delay function */

/*header files for TM4C123 device and sprintf library */
#include "TM4C123GH6PM.h"
#include <stdio.h>

/*Function prototype for Timer0A and UART module initialization */

uint32_t Measure_distance(void);
void Timer0ACapture_init(void);
void Delay_MicroSecond(int time);
void UART5_init(void);
void UART5_Transmitter(unsigned char data);
void printstring(char *str);
void Delay(unsigned long counter);

/* global variables to store and display distance in cm */
```

Task 1.

```
uint32_t time; /*stores pulse on time */
uint32_t distance; /* stores measured distance value */
char mesg[20];  /* string format of distance value */

/* main code to take distance measurement and send data to UART terminal */
int main(void)
{
Timer0ACapture_init();  /*initialize Timer0A in edge edge time */
UART5_init(); /* initialize UART5 module to transmit data to computer */
       while(1)
       {
time = Measure_distance(); /* take pulse duration measurement */
distance = (time * 10625)/10000000; /* convert pulse duration into distance */
sprintf(mesg, "\r\nDistance = %d cm", distance); /*convert float type distance data into
string */
printstring(mesg); /*transmit data to computer */
Delay(2000);

       }

}
/* This function captures consecutive rising and falling edges of a periodic signal */
/* from Timer Block 0 Timer A and returns the time difference (the period of the signal).
*/
uint32_t Measure_distance(void)
{
    int lastEdge, thisEdge;

          /* Given 10us trigger pulse */
          GPIOA->DATA &= ~(1<<4); /* make trigger  pin high */
          Delay_MicroSecond(10); /*10 seconds delay */
          GPIOA->DATA |= (1<<4); /* make trigger  pin high */
          Delay_MicroSecond(10); /*10 seconds delay */
          GPIOA->DATA &= ~(1<<4); /* make trigger  pin low */

        while(1)
        {
         // clear timer0A capture flag
    TIMER0->ICR = 4;
    while((TIMER0->RIS & 4) == 0) ;    /* wait till captured */
         if(GPIOB->DATA & (1<<6)) /*check if rising edge occurs */
               {
    lastEdge = TIMER0->TAR;     /* save the timestamp */
               /* detect falling edge */
    TIMER0->ICR = 4;            /* clear timer0A capture flag */
    while((TIMER0->RIS & 4) == 0) ;    /* wait till captured */
    thisEdge = TIMER0->TAR;     /* save the timestamp */
               return (thisEdge - lastEdge); /* return the time difference */
               }
        }
}

/* Timer0A initialization function */
/* Initialize Timer0A in input-edge time mode with up-count mode */
void Timer0ACapture_init(void)
{
    SYSCTL->RCGCTIMER |= 1;     /* enable clock to Timer Block 0; please refer to page 339
of the datasheet*/
    SYSCTL->RCGCGPIO |= 1;      /* enable clock to PORTB; please refer to page 349 of the
datasheet*/

    GPIOB->DIR &= (1<<6);        /* make PB6 an input pin; please refer to page 663 of the
datasheet*/
    GPIOB->DEN |= (1<<6);         /* make PB6 as digital pin; please refer to page 682 on
the datasheet */
    GPIOB->AFSEL |= (1 << 6);       /* use PB6 alternate function */
    GPIOB->PCTL &= 0x00000F00;  /* configure PB6 for T0CCP0 */
    GPIOB->PCTL |= 0x07000000;

          /* PA4 as a digital output signal to provide trigger signal */
          SYSCTL->RCGCGPIO |= 0x1;      /* enable clock to PORTA */
          GPIOA->DIR &= (1<<4);         /* set PA4 as a digial output pin */
          GPIOA->DEN |= (1<<4);         /* make PA4 as digital pin */

/*
```

Task 1.

```
The next block of code will be used
Look at page 724 for input edge time mode description
*/

    TIMER0->CTL &= 0x0;          /* disable timer0A during setup;  */
    TIMER0->CFG = 0x4;            /* 16-bit timer mode */
    TIMER0->TAMR = 0x17;        /* up-count, edge-time, capture mode */
    TIMER0->CTL |= (0<<2);        /* capture the rising edge */
                TIMER0->CTL |= (0<<3);                        // In order to capture the
rising edge the 3 and 2 bits should be 0 of the GPTMCTL register
    TIMER0->CTL |= (1<<0);           /* enable timer0A */
}



/* Create one microsecond second delay using Timer block 1 and sub timer A */

void Delay_MicroSecond(int time)
{
    int i;
    SYSCTL->RCGCTIMER |= 2;     /* enable clock to Timer Block 1 */
    TIMER1->CTL = 0;            /* disable Timer before initialization */
    TIMER1->CFG = 0x04;         /* 16-bit option */
    TIMER1->TAMR = 0x02;        /* periodic mode and down-counter */
    TIMER1->TAILR = 16 - 1;  /* TimerA interval load value reg */
    TIMER1->ICR = 0x1;          /* clear the TimerA timeout flag */
    TIMER1->CTL |= 0x01;        /* enable Timer A after initialization */

    for(i = 0; i < time; i++)
    {
        while ((TIMER1->RIS & 0x1) == 0) ;      /* wait for TimerA timeout flag */
        TIMER1->ICR = 0x1;      /* clear the TimerA timeout flag */
    }
}
void UART5_init(void)
{
          SYSCTL->RCGCUART |= 0x20;  /* enable clock to UART5 */
    SYSCTL->RCGCGPIO |= 0x10;  /* enable clock to PORTE for PE4/Rx and RE5/Tx */
    /* UART0 initialization */
    UART5->CTL = 0;         /* UART5 module disbable */
    UART5->IBRD = 104;      /* for 9600 baud rate, integer = 104 */
    UART5->FBRD = 11;       /* for 9600 baud rate, fractional = 11*/
    UART5->CC = 0;          /*select system clock*/
    UART5->LCRH = 0x60;     /* data lenght 8-bit, not parity bit, no FIFO */
    UART5->CTL = 0x301;     /* Enable UART5 module, Rx and Tx */

    /* UART5 TX5 and RX5 use PE4 and PE5. Configure them digital and enable alternate
function */
    GPIOE->DEN = 0x30;      /* set PE4 and PE5 as digital */
    GPIOE->AFSEL = 0x30;    /* Use PE4,PE5 alternate function */
    GPIOE->AMSEL = 0;    /* Turn off analg function*/
    GPIOE->PCTL = 0x00110000;     /* configure PE4 and PE5 for UART */
}
void UART5_Transmitter(unsigned char data)
{
    while((UART5->FR & (1<<5)) != 0); /* wait until Tx buffer not full */
    UART5->DR = data;                  /* before giving it another byte */
}
void printstring(char *str)
{
  while(*str)
        {
                UART5_Transmitter(*(str++));
        }
}
void Delay(unsigned long counter)
{
        unsigned long i = 0;

        for(i=0; i< counter*1000; i++);
}
/* This function is called by the startup assembly code to perform system specific
initialization tasks. */
```