Highlighting: Task 01(no highlighting), Task 02, Task 03)

Task01: Submit a comprehensive commented file of the original code.

```
/*Shabrya Lott
 * Tiva_c Lab04
 * Usage:This is a simple program enables toggles LED using a timer and interrupt
 * Input: NONE
 * Output: Blue LED's toggling as specified freq and duty cycle
 */
#include <stdint.h>                     //Variable definitions for the C99 standard
#include <stdbool.h>                    //Boolean definitions for the C99 standard
#include "inc/tm4c123gh6pm.h"    //def. for the interrupt and register assignments on the Tiva C Series device on the launchPad board
#include "inc/hw_memmap.h"              //Macros defining the memory map of the Tiva C Series
#include "inc/hw_types.h"               //Defines common types and macros
#include "driverlib/sysctl.h"           //Defines macros for System Control API of Driverlib
#include "driverlib/interrupt.h"//defines & macros for NVIC Controller(Interrupt)API of driverlib.
#include "driverlib/gpio.h"             //Defines macros for GPIO API of Driverlib
#include "driverlib/timer.h"            //Defines and macros for Timer API of driverLib.

int main(void)
{
        uint32_t ui32Period;

        //System clock to 40Mhz (PLL= 400Mhz / 10 = 40Mhz)
        SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

        //Port configuration (LEDS)
        //Enable GPIOF port
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
        //set LEDS connected to pins as outputs
        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

        //Timer configurations
        //Enable TIMER0
        SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
        //Set Timer0 to Periodic mode
        TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
        //PWM period at 10 Hz at 50% duty cycle
        ui32Period = (SysCtlClockGet() / 10) / 2;
        //Load timer period
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
        //Enable interrupts on Timer0
        IntEnable(INT_TIMER0A);
        //Set Timer0 to interrupt at timeout
        TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
        //Enable master interrupt
        IntMasterEnable();
        //Enable Timer
        TimerEnable(TIMER0_BASE, TIMER_A);
        while(1) //infinite loop
        {
        }
}
```

```
void Timer0IntHandler(void)
{
	// Clear the timer interrupt
	TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);

	// Read the current state of the GPIO pin and
	// write back the opposite state
	if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
	{
		//turn off LEDS
		GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);

	}
	else
	{
		//Turn on Blue LED
		GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);

	}
}
```

Task02: Change the toggle of the GPIO at 50 Hz and at 20% duty cycle and verify

```
uint32_t ui32Period;
int main(void)
{	.
	.
	.
	//PWM period at 50Hz
	ui32Period = (SysCtlClockGet() / 50);
	//Load timer period with 80% of period
	TimerLoadSet(TIMER0_BASE, TIMER_A, (ui32Period -1)*0.80);
	.
	.
	.
}
void Timer0IntHandler(void)
{	.
	.
	.
	if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
	{
		//turn off LEDS
		GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
		//Load timer period with 80% of period
		TimerLoadSet(TIMER0_BASE, TIMER_A, (ui32Period -1)*0.80);
	}
	else
	{
		//Turn on Blue LED
		GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
		//Load timer period with  DC = 20%
		TimerLoadSet(TIMER0_BASE, TIMER_A, (ui32Period -1)*0.20);
	}
}
```

Task 03: Include a GPIO Interrupt to Task 02 from switch SW2 to turn ON and the LED for 0.5 sec. Use a Timer1 to calculate the 0.5 sec delay. The toggle of the GPIO is suspended when executing the interrupt.

```
void GPIOF0IntHandler(void);
int main(void)
{       .
        .
        .

        //Unlock Pin F0 to use an interrupt on SW2
        SYSCTL_RCGC2_R |= 0x00000020;     // activate clock for Port F
        GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock GPIO Port F
        GPIO_PORTF_CR_R = 0x1F;           // allow changes to PF4-0
        // only PF0 needs to be unlocked, other bits can't be locked
        GPIO_PORTF_AMSEL_R = 0x00;        // disable analog on PF
        GPIO_PORTF_PCTL_R = 0x00000000;   // PCTL GPIO on PF4-0
        GPIO_PORTF_DIR_R = 0x0E;          // PF4,PF0 in, PF3-1 out
        GPIO_PORTF_AFSEL_R = 0x00;        // disable alt funct on PF7-0
        GPIO_PORTF_PUR_R = 0x11;          // enable pull-up on PF0 and PF4
        GPIO_PORTF_DEN_R = 0x1F;          // enable digital I/O on PF4-0

        .
        .
        .
}
void Timer0IntHandler(void)
{       .
        .
        .
}
void GPIOF0IntHandler(void) //interrupt handler for GPIO pin F0
{
        uint32_t delay;
        //clear interrupt flag on pin F0
        GPIOIntClear(GPIO_PORTF_BASE, GPIO_PIN_0);
        //Turn on Blue LED
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4);

        //set up TIMER1

        //Enable TIMER1 peripheral
        SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
        //Set TIMER1 to periodic mode
        TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
        delay = (SysCtlClockGet()/8);
        //load TIMER1 period
        TimerLoadSet(TIMER1_BASE, TIMER_A, (delay-1));
        //Enable timer
        TimerEnable(TIMER1_BASE, TIMER_A);
        //wait for 0.5 sec
        while (TimerValueGet(TIMER1_BASE, TIMER_A) < (delay-8));
         //Turn off Blue LED
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0 );

}
```