Itzel Becerril
Github root directory: https://github.com/HadidBuilds/TivaC_project_labs

Date Submitted: 09/28/19

Task 00:

Youtube Link:
https://www.youtube.com/watch?v=JRWcje1kyH4

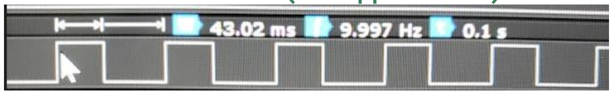
Execute provided code
```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"


int main(void)
{
	uint32_t ui32Period; //unsigned 32 bit integer

	//Clock runs at 40MHz
	SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAI
N);

	SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
	GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

	SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
	TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

	ui32Period = (SysCtlClockGet() / 10) / 2;//10Hz and 50% DC
	TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

	//Enabling Interrupt
	IntEnable(INT_TIMER0A);
	TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
	IntMasterEnable();

	//Will start the timer and interrupts will begin triggering on the timeouts
	TimerEnable(TIMER0_BASE, TIMER_A);

	//The toggling of the GPIO will happen in the interrupt service routine
	while(1)
	{
	}
}

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

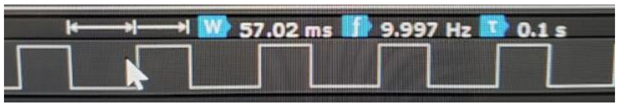
```
        // Read the current state of the GPIO pin and
        // write back the opposite state
        if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
        {
            //turn off LEDs
              GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
        else
        {
            //lights blue LED
              GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        }
}
```

Task 01:

Youtube Link:
https://www.youtube.com/watch?v=u0CFcncRoSI

Modified Schematic (if applicable):





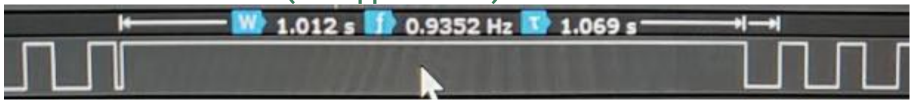
Modified Code:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"


int main(void)
{
    uint32_t ui32Period; //unsigned 32 bit integer

    //Clock runs at 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    ui32Period = (SysCtlClockGet() / 10)/2;//10Hz and 50% DC
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    //Enabling Interrupt
    IntEnable(INT_TIMER0A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    //Will start the timer and interrupts will begin triggering on the timeouts
    TimerEnable(TIMER0_BASE, TIMER_A);

    //The toggling of the GPIO will happen in the interrupt service routine
    while(1)
    {
    }
}

void Timer0IntHandler(void)
{
    uint32_t ui32Period1, ui32Period2; //unsigned 32 bit integer

    // 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
        ui32Period1 = (SysCtlClockGet() / 10) * .57;//10Hz and 57% DC
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period1 -1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        //lights blue LED
        ui32Period2 = (SysCtlClockGet() / 10) * .43;//10Hz and 43% DC
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period2 -1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
    }
}
```

Task 02:

Youtube Link:
https://www.youtube.com/watch?v=3O5Fmm4Ijf8

Modified Schematic (if applicable):



Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Modified Code:

```
#define SW2 GPIO_PIN_0

#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_gpio.h"
#include "inc/hw_memmap.h"
#include "inc/hw_sysctl.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"

uint32_t secDelay;

int main(void)
{
    uint32_t ui32Period; //unsigned 32 bit integer

    //Clock runs at 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //Unlocking SW2
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK ) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) = 0x1;

    //Switch Interrupt
    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, SW2); //enabling switch as an inout
    GPIOPadConfigSet(GPIO_PORTF_BASE, SW2, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
//turn weak pull-ups on
    GPIOIntTypeSet(GPIO_PORTF_BASE,SW2,GPIO_RISING_EDGE); //sets interrupt to rising
edge on GPIO
    GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_0); //enables a specific event within
the GPIO to generate an interrupt

    //Timer 0 enabling and config
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    //Timer 1 enabling and config
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    //Timer0 value
    ui32Period = (SysCtlClockGet() / 10)/2;
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    //Timer1 value
    secDelay = SysCtlClockGet();
    ui32Period = (SysCtlClockGet() / 10)/2;
    TimerLoadSet(TIMER1_BASE, TIMER_A, secDelay);

    //Enabling Interrupt
    IntEnable(INT_TIMER0A);
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    //Will start the timer and interrupts will begin triggering on the timeouts
    TimerEnable(TIMER0_BASE, TIMER_A);
    TimerEnable(TIMER1_BASE, TIMER_A);

    //Port F Interrupt for SW1
    IntEnable(INT_GPIOF); //enables the specific vector associated with GPIO

    //Will allow interrupts to occur without other occurrences in code
    while(1)
    {
    }
}

void Timer0IntHandler(void)
{
    uint32_t ui32Period1, ui32Period2; //unsigned 32 bit integer

    // 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
        ui32Period1 = (SysCtlClockGet() / 10) * .57;//10Hz and 57% DC
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period1 -1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        //lights blue LED
        ui32Period2 = (SysCtlClockGet() / 10) * .43;//10Hz and 43% DC
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period2 -1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
    }
}

void Timer1IntHandler(void)
{
    TimerIntClear(TIMER1_BASE,TIMER_A); //Required to launch next interrupt
    TimerEnable(TIMER0_BASE, TIMER_A);
}

void IntPortFHandler(void)
{
    TimerDisable(TIMER0_BASE, TIMER_A); //stop timer0
    GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_0); //Clear interrupt
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4); //turn on Blue LED
}


-----------------------------------------------------------------------------------
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Itzel Becerril
Github root directory: https://github.com/HadidBuilds/TivaC_project_labs

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.