Jett Guerrero
Github root directory: https://github.com/guerrj1/Advanced-Embedded-Systems

Date Submitted: 10/6/2019

Task 00: Execute provided code

Youtube Link: https://youtu.be/HesUo0tELoQ

Modified Code:

```
//Task 0
#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;

	//Clock initialization
	SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

	//GPIO settings
	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);

	//Timer 0 initializations
	ui32Period = (SysCtlClockGet() / 10) / 2;  //50% duty cycle
	TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

	IntEnable(INT_TIMER0A);
	TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
	IntMasterEnable();

	TimerEnable(TIMER0_BASE, TIMER_A);

	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))
```

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

```
        {
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0); //off
        }
        else
        {
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);  //on
        }
}
```

Task 01:

Youtube Link: https://youtu.be/-73rJcj9zlE

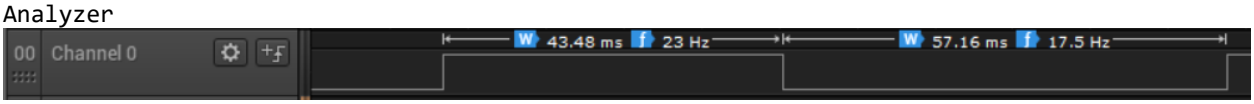
Modified Schematic (if applicable):

Modified Code:

```
//task 1
#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"

//Global Variables for duty cycle
uint32_t ui32PeriodHigh;
uint32_t ui32PeriodLow;

int main(void)
{
        //clock initialization
        SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

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

        //Timer 0 initialization
        SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
        TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

        ui32PeriodHigh = (SysCtlClockGet() / 10) / 2.3;  //43% duty cycle
        ui32PeriodLow = (SysCtlClockGet() /10) / 1.75;
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh -1);
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodLow -1);

        IntEnable(INT_TIMER0A);
        TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
```

```
	IntMasterEnable();

	TimerEnable(TIMER0_BASE, TIMER_A);

	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))
	{
		TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodLow); //57 duty cycle off
		GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
	}
	else
	{
		TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh);  //43 duty cycle on
		GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
	}
}
```

Analyzer



Task 02:

Youtube Link: https://youtu.be/v3q_sHAQZcU

Modified Schematic (if applicable):

Modified Code:

```
//Task 2
#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"

//Global Variables
uint32_t ui32PeriodHigh;
uint32_t ui32PeriodLow;
```

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

```
//Prototype
void PORTFPin4IntHandler();

int main(void)
{
	//uint32_t ui32Period;

	SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

	//GPIO Settings
	SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
	GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4);
	GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);
	GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
	GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_4);
	GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_INT_PIN_4, GPIO_RISING_EDGE);
	IntEnable(INT_GPIOF);

	//Timer 0 Initializations
	SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
	TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

	ui32PeriodHigh = (SysCtlClockGet() / 10) / 2.3; //43% duty cycle
	ui32PeriodLow = (SysCtlClockGet() /10) / 1.75;  //57% off
	TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh -1);
	TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodLow -1);

	IntEnable(INT_TIMER0A);
	TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
	IntMasterEnable();

	TimerEnable(TIMER0_BASE, TIMER_A);

	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))
	{
		TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodLow);
		GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
	}
	else
	{
		TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh);
		GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
	}
```

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

```
}

void Timer1IntHandler(void)
{
	TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
	TimerLoadSet(TIMER0_BASE, TIMER_A, SysCtlClockGet());
}

void PORTFPin4IntHandler(void)
{
	//Clear the GPIO interrupt
	GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_4);
	//Read the current state of the GPIO pin
	//write back the opposite state - OFF
	GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);

	//call timer 1 delay
	SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
	IntMasterEnable();
	TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
	TimerLoadSet(TIMER1_BASE, TIMER_A, 0);
	IntEnable(INT_TIMER1A);
	TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
	TimerEnable(TIMER1_BASE, TIMER_A);

	//write back the opposite state - ON
	GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_2, GPIO_PIN_2);
}
```

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