Jiajian Chen
Github root directory: https://github.com/IMSB007/2019Fall

Date Submitted: 9/28/2019

Task 01:

Youtube Link: https://www.youtube.com/watch?v=kQBP2-Zp74Y

Modified Schematic (if applicable):

Modified Code:

```
// Insert code here
#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"

uint32_t ui32PeriodHigh;
uint32_t ui32PeriodLow;

int main(void)
{


        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);

        ui32PeriodHigh = (SysCtlClockGet() / 10) * 0.43;
        ui32PeriodLow = (SysCtlClockGet() / 10) * 0.57;

        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh -1);

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

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

```
        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-1);
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
        else
        {
                TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh-1);
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        }
}
```

Task 02:

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

Modified Schematic (if applicable):

Modified Code:

```
// Insert code here
#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"
```

```
uint32_t ui32PeriodHigh;
uint32_t ui32PeriodLow;
int time = 1;

int main(void)
{


        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);

        ui32PeriodHigh = (SysCtlClockGet() / 10) * 0.43;
        ui32PeriodLow = (SysCtlClockGet() / 10) * 0.57;

        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh -1);

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

        TimerEnable(TIMER0_BASE, TIMER_A);
        configureTimer1A();
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
        GPIOPinTypeGPIOInput(GPIO_PORTF_BASE,GPIO_PIN_4);
        GPIOPadConfigSet(GPIO_PORTF_BASE,GPIO_PIN_4,GPIO_STRENGTH_2MA,GPIO_PIN_TYPE_ST
D_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);



        while(1)
        {
        }
}

void PortFPin4IntHandler(void)
{
        GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_4);
        GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_2,GPIO_PIN_2);
```

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

```
        timer1A_delaySec(1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
}

void timer1A_delaySec(int ttime)
{
        int i;

        SYSCTL_RCGCTIMER_R |= 2;
        TIMER1_CTL_R=0;
        TIMER1_CFG_R=0x04;
        TIMER1_TAMR_R =0x02;
        TIMER1_TAILR_R=64000-1;
        TIMER1_TAPR_R=250-1;
        TIMER1_ICR_R=0x1;
        TIMER1_CTL_R |= 0x01;
        for(i=0;i<ttime;i++)
        {
                while((TIMER1_RIS_R & 0x1) == 0)
                {}
                TIMER1_ICR_R=0X1;
        }
}

void configureTimer1A()
{
        SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
        IntMasterEnable();
        TimerConfigure(TIMER1_BASE,TIMER_CFG_PERIODIC);

        TimerLoadSet(TIMER1_BASE, TIMER_A, 120000000);
        IntEnable(INT_TIMER1A);
        TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
        TimerEnable(TIMER1_BASE, TIMER_A);
}

void Timer1AHandler(void)
{
        TimerIntClear(TIMER1_BASE,TIMER_A);
}

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
```

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

```
        if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
        {
                TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodLow-1);
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
        else
        {
                TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh-1);
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        }
}
```

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