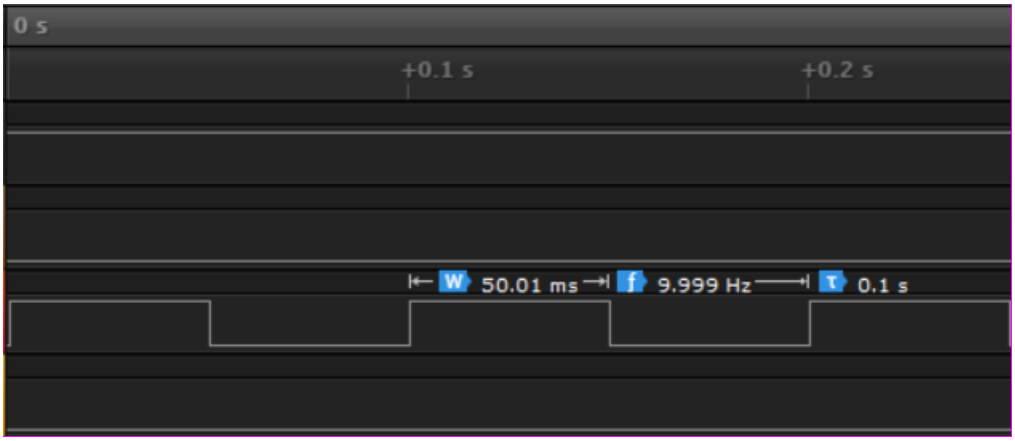
Ameera Essaqi
Github root directory: https://github.com/AmeeraE/microcontrollers/tree/master/TIVAC

Date Submitted: 10/20/19

Task 00: Execute provided code

Youtube Link: https://youtu.be/opduzr5G9yU



Task 01:

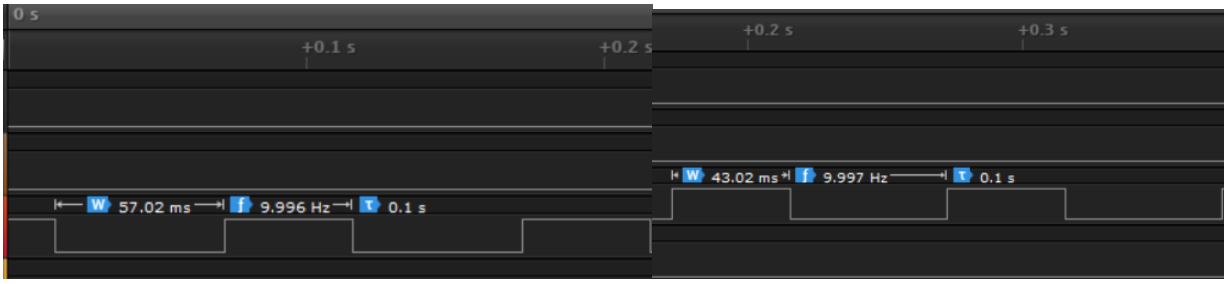
Youtube Link: https://youtu.be/baHZctpp0s8

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;

    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;
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

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

    TimerEnable(TIMER0_BASE, TIMER_A);

    while(1)
    {
```

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

```
    }
}

void Timer0IntHandler(void)
{

    uint32_t ui32PeriodOn;
    uint32_t ui32PeriodOff;

    // 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))
    {
        ui32PeriodOff = (SysCtlClockGet() / 10) *.57;
         TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodOff -1);

        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        ui32PeriodOn = (SysCtlClockGet() / 10) *.43;
         TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodOn -1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
    }
}
```



Task 02:

Youtube Link: https://youtu.be/3mACettFDCA

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"
#include "inc/hw_gpio.h"

int main(void)
{
    uint32_t ui32Period;

    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
```

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

```
    HWREG(GPIO_PORTF_BASE+GPIO_O_LOCK)=GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE+GPIO_O_CR) |= GPIO_PIN_0;

    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0);
    GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_0);
    IntEnable(INT_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) *.57;
    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 PortFIntHandler(){

 int status=0;  //checks if button is pressed

  status = GPIOIntStatus(GPIO_PORTF_BASE,true);
  GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_0);

  if(status & GPIO_INT_PIN_0)
  {
    //Then there was a Button pin interrupt
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2,4);
    Timer1ADelay(5); //Delays for 1 second
  }
    /*
      This delay is for deboucing but since it's in a interrupt it
      should be used a better method that is faster
    */
    SysCtlDelay(100000);

  }

void Timer1ADelay(int timer) //Delays timer for .5 seconds
{
    int i;

    SYSCTL_RCGCTIMER_R |= 2; //Enable Timer1.
    TIMER1_CTL_R = 0; //Disable Timer1A during setup.
    TIMER1_CFG_R = 0x04; //Configure to 16-bit mode.
    TIMER1_TAMR_R = 0x02; //Configure for periodic mode, default down.
    TIMER1_TAILR_R = 64000-1; //Period=64000. Reload Value.
    TIMER1_TAPR_R = 625-1; //Bus Clock Resolution.
    TIMER1_ICR_R = 0x1; //Clear Timer1A timeout flag.
    TIMER1_CTL_R |= 0x1; //Enable Timer1A.

    for(i = 0; i < timer; i++)
    {
        while((TIMER1_RIS_R &0x01) == 0); //Timeout wait.
        TIMER1_ICR_R = 0x1; //Flag clear.
    }
}
```

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

```
void Timer0IntHandler(void)
{
    uint32_t ui32PeriodOn;
    uint32_t ui32PeriodOff;

    // 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))
    {
        ui32PeriodOff = (SysCtlClockGet() / 10) *.57;
         TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodOff -1);

        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        ui32PeriodOn = (SysCtlClockGet() / 10) *.43;
         TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodOn -1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
    }
}
```

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