Student Name: Serak Gebremedhin
Github root directory: https://github.com/Ber-geb/solid-octo-tribble.git

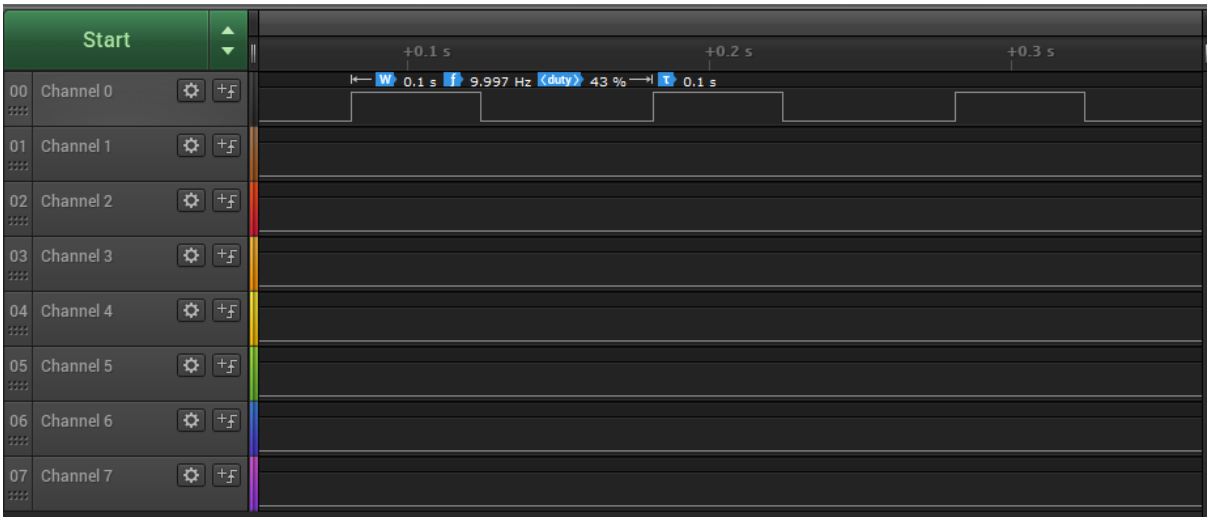
Date Submitted: **09/28/19**

Task 00: Execute provided code

Youtube Link:

https://youtu.be/f39u6m32AMY

Task 01:



Youtube Link:

https://youtu.be/f39u6m32AMY

Modified Code:

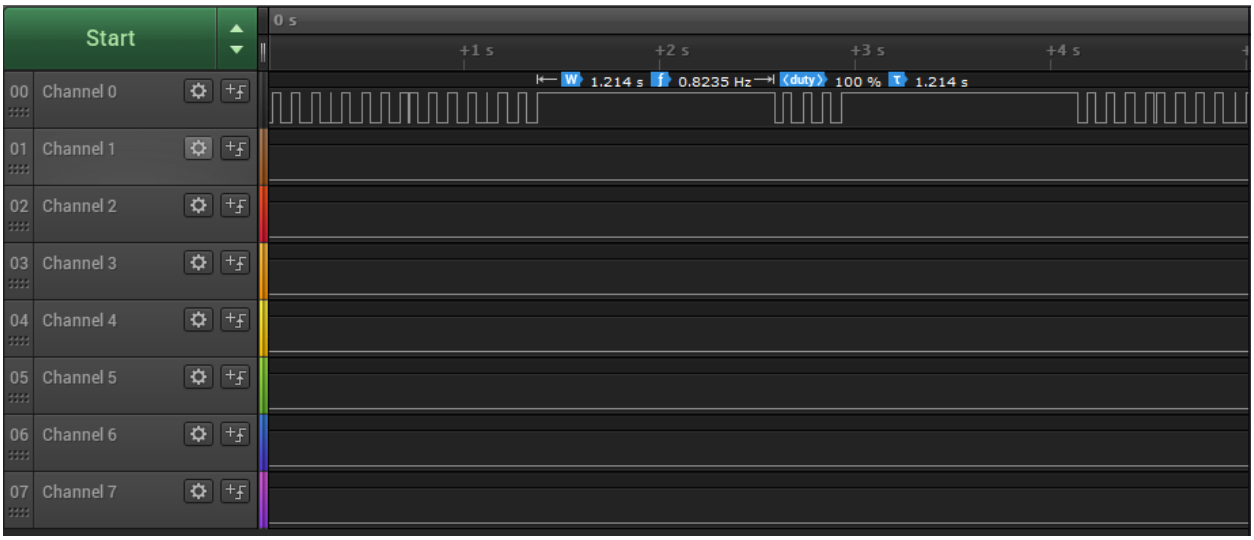
```
// Insert code here

#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"
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.

```
    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) * 0.43; //toggle GPIO at 10Hz with 43% 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)
{
    uint32_t ui32Period;
    // 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))
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        ui32Period = (SysCtlClockGet() / 10) * 0.57;
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
    }
    else
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        ui32Period = (SysCtlClockGet() / 10) * 0.43;
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
    }
}
```

Student Name: Serak Gebremedhin
Github root directory: https://github.com/Ber-geb/solid-octo-tribble.git

Task 02:



Youtube Link:

https://youtu.be/f39u6m32AMY

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"

void timer1A_delaySec(int ttime)
{
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4); //turn on led

    int i;

    SYSCTL_RCGCTIMER_R |= 2; //enable clock to timer block 1
    TIMER1_CTL_R = 0;        //disable timer before initialization
    TIMER1_CFG_R = 0x04;     //16-bit option
    TIMER1_TAMR_R = 0x02;    //periodic mode and down counter
    TIMER1_TAILR_R = 64000 - 1;   //TimerA interval load value reg
    TIMER1_TAPR_R = 250 - 1;    //TimerA Prescaler 16MHz/250=64000Hz
    TIMER1_ICR_R = 0x1;      //clear the TimerA timeout flag
    TIMER1_CTL_R |= 0x01;    //enable Timer A after initialization
    for(i = 0; i < ttime; i++){
        while ((TIMER1_RIS_R & 0x1) == 0); //wait for TimerA timeout flag
        TIMER1_ICR_R = 0x1;               //clear the timer A timeout flag
```

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

```
    }
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0); //turn off
led
}

void configureTimer1A()
{
    uint32_t ui32Period;
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1); //Enable Timer 1 Clock
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC); //configure timer operation as
periodic
    //Configure timer frequency
    //Frequency is given by MasterClock / CustomValue
    //ui32Period = (SysCtlClockGet() / 1) * 0.5; //toggle GPIO at 1Hz with 50% duty
cycle
    //TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period -1);
    TimerLoadSet(TIMER1_BASE, TIMER_A, 120000000);

    IntEnable(INT_TIMER1A); //Enable timer 1a interrupt
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT); //timer 1a interrupt when
timeout
    IntMasterEnable(); //Enable Interrupts
    TimerEnable(TIMER1_BASE, TIMER_A); //Start Timer 1a

}

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) * 0.43; //toggle GPIO at 10Hz with 43% duty
cycle
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

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

    TimerEnable(TIMER0_BASE, TIMER_A);

    //Switch interrupt
    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);

    //configure timer1a
    configureTimer1A();


    while(1)
    {
    }
}
void Timer0IntHandler(void)
{
    uint32_t ui32Period;
    // 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))
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        ui32Period = (SysCtlClockGet() / 10) * 0.57;
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
    }
    else
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        ui32Period = (SysCtlClockGet() / 10) * 0.43;
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
    }
}

void PortFPin4IntHandler(void){
    //Clear the GPIO interrupt
    GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_4);
    //Read the current state of the GPIO pin and
    //write back the opposite state
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
    //Call TIMER 1 Delay
    timer1A_delaySec(3);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);

}


void Timer1AIntHandler(void){
    //Required to launch next interrupt

    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    //TimerIntClear(TIMER1_BASE, TIMER_A);

    // Read the current state of the GPIO pin and
    // write back the opposite state
    if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
    {
```

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