Ricky Perez
Github root directory: https://github.com/RickyPerez79/da_submissions_2019

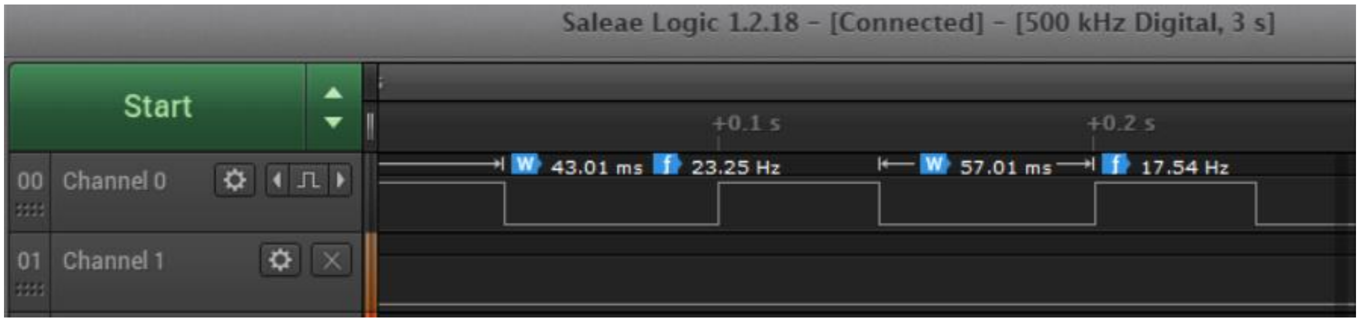
Date Submitted: **10/05/2019**

Task 00: Execute provided code

Youtube Link: https://youtu.be/9e7T5ejmnWg

Task 01:

Youtube Link: https://youtu.be/H8hdNO2ljCA

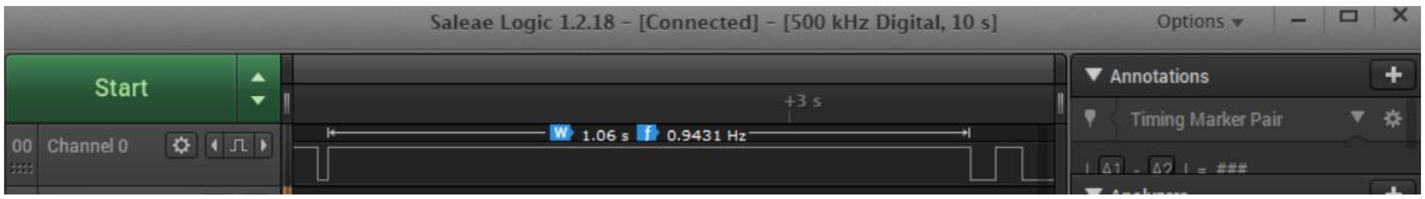


Modified Code:

```
#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"

uint32_t ui32PeriodLow;
uint32_t ui32PeriodHigh;
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) *.43 ; // Period of 0.0430 @10Hz
    ui32PeriodLow = (SysCtlClockGet() / 10) *.57; // Period of 0.0570 @10Hz

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

}
-----------------------------------------------------------------------------------
```

Task 02:

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



```
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"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "driverlib/pin_map.h"
```

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

```
#include "driverlib/sysctl.c"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.c"
#include "driverlib/gpio.h"

// function prototypes
void configTimer0();
void configTimer1A();

// Global Variables
uint32_t ui32Period;
uint32_t ui32PeriodHigh;
uint32_t ui32PeriodLow;
uint32_t sec_delay;


int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable Port F

    // unlock the GPIOLOCK register for PF0 using direct Register Programming
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;

    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0); // set switch 2
    GPIOPadConfigSet(GPIO_PORTF_BASE
,GPIO_PIN_0,GPIO_STRENGTH_2MA,GPIO_PIN_TYPE_STD_WPU); // disables pull up resistor of
switch 2
    GPIOIntTypeSet(GPIO_PORTF_BASE,GPIO_PIN_0,GPIO_FALLING_EDGE);

    GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_0); // enables interupts from switch
2
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    sec_delay = 1.25*(SysCtlClockGet());
    ui32PeriodHigh = (SysCtlClockGet() / 10) *.43 ; // Period of 0.0430 @10Hz
    ui32PeriodLow = (SysCtlClockGet() / 10) *.57; // Period of 0.0570 @10Hz

    configTimer0();
    configTimer1A();
    IntMasterEnable();

    IntEnable(INT_GPIOF);

    while(1)
    {
    }
}


void configTimer0()
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0); // enable timer0
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
    ui32Period = (SysCtlClockGet() / 10) / 2;
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
    IntEnable(INT_TIMER0A); // Enable Timer0A interrupts
    TimerEnable(TIMER0_BASE, TIMER_A);// enables timer A
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
}

void configTimer1A()
{

    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
    TimerLoadSet(TIMER1_BASE, TIMER_A, sec_delay);  // counts up to sec_delay
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntEnable(INT_TIMER1A);
    TimerEnable(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
    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);
    }
}

void Timer1IntHandler(void)
{
    TimerIntClear(TIMER1_BASE, TIMER_A);
    TimerEnable(TIMER0_BASE, TIMER_A);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
}

void PortFIntHandler(void)
{
    TimerDisable(TIMER0_BASE, TIMER_A);
    GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_0);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);

}

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

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