Ron Joshua Recrio
Github root directory: https://github.com/recrio/submissions.2/tree/master/Lab%204

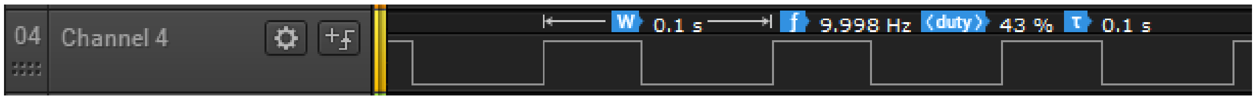
Date Submitted: 9/28/2019

Task 00: Execute provided code

Youtube Link:
https://youtu.be/BWhUm6MkkHk

Task 01:

Verification:



Youtube Link:
https://youtu.be/eCEWkqNf93I

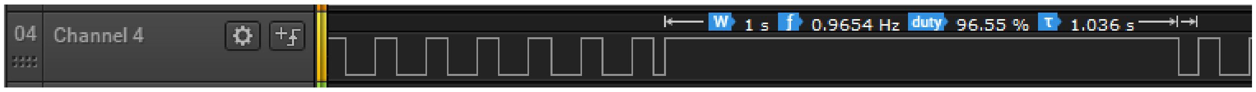
Modified Schematic (if applicable):

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 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);
    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:

Could not use SW2 because of NMI default making it locked. Tried to unlock it but it ended up making the code not work so I used SW1 instead.
Verification:



Youtube Link:
https://youtu.be/Wteja5E3ERY
Modified Schematic (if applicable):

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 ui32PeriodHigh;
uint32_t ui32PeriodLow;
uint32_t ui32Delay_1s;
```

Ron Joshua Recrio
Github root directory: https://github.com/recrio/submissions.2/tree/master/Lab%204

```
int main(void)
{

    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

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


    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    ui32Delay_1s = (SysCtlClockGet());

    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);
    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);
    }
}
```

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

void PortFPin4IntHandler(void)
{
    TimerDisable(TIMER0_BASE, TIMER_A);
    // Clear the GPIO interrupt
    GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_4);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
    // Call TIMER 1 Delay
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    IntMasterEnable();
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Delay_1s);
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    TimerEnable(TIMER1_BASE, TIMER_A);

}
```

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