Saul Alejandro Mendoza Guzman
Github root directory:

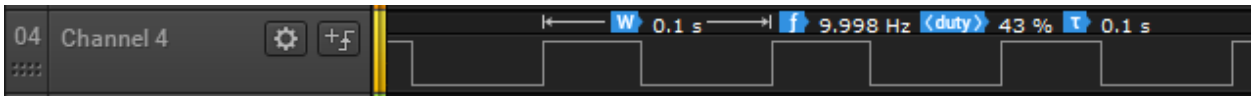
Date Submitted: 9/28/2019

Task 00: Execute provided code

Youtube Link:
https://www.youtube.com/watch?v=x36VptR0c30

Task 01:

Verification:



Youtube Link:
https://www.youtube.com/watch?v=1Mmxl5n4F08
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"


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)
    {
```

```
    }
}

void Timer0IntHandler(void)
{
    uint32_t ui32Period_high,ui32Period_low;
    // 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))
    {
        ui32Period_low = (SysCtlClockGet() / 10) * 0.57;
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period_low -1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        ui32Period_high = (SysCtlClockGet() / 10) * 0.43;
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period_high -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://www.youtube.com/watch?v=2OL3c6F08yE

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

```
uint32_t ui32High;
uint32_t ui32Low;
uint32_t ui32Delay_1s;

int main(void){
    uint32_t ui32Period; // This variable is used to set the timer

    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
// 40MHz System clock ((400MHz/(default 2))/ 5(SYSCTL_SYSDIV_5))

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable port F (PF0...PF4)

    // The following three lines of code 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 PF0 (switch 2)
    GPIOPadConfigSet(GPIO_PORTF_BASE
,GPIO_PIN_0,GPIO_STRENGTH_2MA,GPIO_PIN_TYPE_STD_WPU); // disables pull of resistor of
PF0 with a 2mA output drive strength
    GPIOIntTypeSet(GPIO_PORTF_BASE,GPIO_PIN_0,GPIO_FALLING_EDGE); // sets PF0 as
falling edge
    //GPIOIntRegister(GPIO_PORTF_BASE,PortFIntHandler); // registers an interrupt
handler for a GPIO port, that is, calls the second parameter function
    GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_0); // enables interupts from PF0
(switch 2)
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3); // set
GPIO_PIN_1,GPIO_PIN_2,GPIO_PIN_3 as base (2 clock cycles needed to toggle) output
pins

    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0); // enable timer0
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC); // configure timer0 as periodic
(Full-width periodic timer)
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    ui32Delay_1s = (SysCtlClockGet());

    ui32Period = (SysCtlClockGet() / 10) / 2; // 40MHz/10/2 gives 2MHz timer (500ns)
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1); // set the timer load value of
timer A
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Delay_1s);

    IntEnable(INT_TIMER0A); // Enable Timer0A interrupts
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT); // only enables Timer A timeout
interrupt
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    TimerEnable(TIMER0_BASE, TIMER_A);// enables timer A
    IntEnable(INT_GPIOF);
```

```
    TimerEnable(TIMER1_BASE, TIMER_A);
    while(1){
    }
}

// make sure Timer 0 subtimer A default handler is replaced with "Timer0IntHandler"
in startup_ccs.c
// also make sure you declare "extern void Timer0IntHandler(void);" bellow "extern
void _c_int00(void);" in startup_ccs.c
void Timer0IntHandler(void){
    uint32_t ui32Period_high,ui32Period_low;
    // 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)){
        ui32Period_low = (SysCtlClockGet() / 10) * 0.57;
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period_low -1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
        else{
            ui32Period_high = (SysCtlClockGet() / 10) * 0.43;
            TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period_high -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);

}
```