Geovanni Portillo
Github root directory: https://github.com/portig1/submissions_E2

Date Submitted: **September 27, 2019**

Task 00: Execute provided code

Youtube Link: https://youtu.be/JkNfsvaS3hg

Task 01:

Youtube Link: https://youtu.be/vN7L4JZnA88
Modified Schematic (if applicable):

Modified Code:

```
  uint32_t ui32PeriodHigh;
    uint32_t ui32PeriodLow;

int main(void)
{


      SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAI
N);

      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;     // = 40MHz -> To get 43%
duty cycle multiply by 0.43
      ui32PeriodLow = (SysCtlClockGet() / 10) * 0.57;     // = 40MHz -> To get 43%
duty cycle multiply by 0.43
      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)
{
```

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

```
        // 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);
                TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodLow -1);
        }
        else
        {
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
                TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh -1);
        }
}
```

Task 02:

Youtube Link: https://youtu.be/PjFIY-6eInA

Modified Schematic (if applicable):

Modified Code:

```
#include "inc/hw_gpio.h"


#define LED_PERIPH SYSCTL_PERIPH_GPIOF
#define LED_BASE GPIO_PORTF_BASE
#define RED_LED GPIO_PIN_1
#define BLUE_LED GPIO_PIN_2
#define GREEN_LED GPIO_PIN_3

#define Button_PERIPH SYSCTL_PERIPH_GPIOF
#define ButtonBase GPIO_PORTF_BASE
#define Button GPIO_PIN_0
#define ButtonInt GPIO_INT_PIN_0

    uint32_t ui32PeriodHigh;
    uint32_t ui32PeriodLow;
void timer1A_delayMs(int ttime);

int main(void)
{

        SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAI
N);

        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
        SysCtlDelay(3);
```

```
        //Switch Config

        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, Button);
        GPIOPadConfigSet(GPIO_PORTF_BASE,Button,GPIO_STRENGTH_2MA,GPIO_PIN_TYPE_STD_WP
U);

        //LED Config
        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

        //GPIO INT Config
        GPIOIntEnable(GPIO_PORTF_BASE, Button);
        GPIOIntTypeSet(GPIO_PORTF_BASE, Button, GPIO_FALLING_EDGE);
        IntEnable(INT_GPIOF);

        //Timer Config
        SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
        TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    ui32PeriodHigh = (SysCtlClockGet() / 10) * 0.43;    // = 40MHz -> To get 43% duty
cycle multiply by 0.43
    ui32PeriodLow = (SysCtlClockGet() / 10) * 0.57;    // = 40MHz -> To get 43% duty
cycle multiply by 0.43
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh -1);

        //Timer INT config
        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))
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodLow -1);
    }
    else
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
```

```
        TimerLoadSet(TIMER0_BASE, TIMER_A, ui32PeriodHigh -1);
    }
}

void PortFIntHandler(void)
{

      //Clear the GPIO interrupt
      GPIOIntClear(GPIO_PORTF_BASE, ButtonInt);
      //Read the current state of the GPIO pin and write back the opposite state
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
      GPIOPinWrite(GPIO_PORTF_BASE, GREEN_LED, GREEN_LED);
      timer1A_delayMs(1000);
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);

      SysCtlDelay(2000000); //delay of .05ms to resolve debouncing. 2M was chosen as
40MHz * .05 = 2M
}

void timer1A_delayMs(int ttime)
{
    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 = 40000 - 1;             //TimerA interval load value reg
    TIMER1_ICR_R = 0x1;                     //clear the Timer A 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 TimerA timeout flag
    }
}
```

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