Phillip Sortomme
Github root directory:

Date Submitted: **10/10/2018**

Task 01:

Youtube Link: https://youtu.be/jyU1u2Lddfg

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"

#define PWM_FREQUENCY 55    // 55Hz base frequency

int main(void)
{
   volatile uint32_t ui32Load;
   volatile uint32_t ui32PWMClock;
   volatile uint8_t ui8Adjust;
   // servo center position:
   // to find pulse width of 1.5ms use equation (PWM period) / 1000.
   // with frequency 55Hz, period = 18.2ms so 18.2 / 1000 = 18.2us,
   // 1.5m / 18.2u = 82.4, so the center position value used is 83
   ui8Adjust = 83;

   ROM_SysCtlClockSet(
         SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN
               | SYSCTL_XTAL_16MHZ);
   ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

   ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
   ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
   ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

   ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
   ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);

   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;
   ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4 | GPIO_PIN_0,
                  GPIO_DIR_MODE_IN);
   ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4 | GPIO_PIN_0,
                   GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

   ui32PWMClock = SysCtlClockGet() / 64;
   ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
   PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
   PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);

   ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
   ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
   ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0);

   while (1)
   {
      // min adjust value equation
      // pulsewidth = adjust*load / 1000
      // load = PWMClk / PWMfrequency
      //
      // minPWMtime = pulsewidth/PWMClk
      // combining the two equations together
      // minPWMtime = adjust*load/(1000*PWMClk)
      // rearranging this we get:
      // adjust = minPWMtime*1000*PWMClk/load
      // so adjust = 0.5ms * 1000 * 625000 / 11363 = 27.5 so we use 28
      if (ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4) == 0x00)
      {
         ui8Adjust--;
```

```
            if (ui8Adjust < 28)
            {
                ui8Adjust = 28;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0,
                                ui8Adjust * ui32Load / 1000);
        }
        // max adjust value is the same as the min value
        // but now we use maxPWMtime so:
        // adjust = 2.5ms * 1000 * 625000 / 11363 = 137.5 so 138 is used
        if (ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_0) == 0x00)
        {
            ui8Adjust++;
            if (ui8Adjust > 138)
            {
                ui8Adjust = 138;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0,
                                ui8Adjust * ui32Load / 1000);
        }
        ROM_SysCtlDelay(100000);
    }
}
```

Task 02:

Youtube Link: https://youtu.be/OhpQPrOrFv4

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"
#include "inc/tm4c123gh6pm.h"

#define PWM_FREQUENCY 55    // 55Hz base frequency

int main(void)
{
    volatile uint32_t ui32Load;  //don't think i need this value
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Dutycycle;
    // starting with a duty cycle of 50%
    ui8Dutycycle = 50;
    ROM_SysCtlClockSet(
            SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN
                    | SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
//    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);

    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;
    ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4 | GPIO_PIN_0,
                      GPIO_DIR_MODE_IN);
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4 | GPIO_PIN_0,
                        GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);


    // formula to calculate the number of cycles for the GenPeriod:
    // Load = 1/PWMfrequency * PWMClk - 1.
    // Load is the parameter, PWMfrequency is the desired frequency, PWMClk is the PWM clock frequency based off the system clock
    // so for this task, Load = 1/55 * 625kHz-1 = 11363
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    ROM_PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
    ROM_PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);


    // formula for number of cycles in duty cycle:
    // N = %DC * Load
    // N is the parameter, %DC is the percent duty cycle, Load
    // starting with 50% DC: N = 50/100*11364 = 5682
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Dutycycle * ui32Load / 100);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);

    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2);

    while (1)
    {
        // Pulsewidth adjustments are made using the same formula above.
        // the smallest duty cycle will be 10% and the largest will be 90%
        if (ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4) == 0x00)
        {
            ui8Dutycycle--;
            if (ui8Dutycycle < 10)
            {
                ui8Dutycycle = 10;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5,
                         ui8Dutycycle * ui32Load / 100);
        }

        if (ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_0) == 0x00)
        {
            ui8Dutycycle++;
            if (ui8Dutycycle > 90)
            {
                ui8Dutycycle = 90;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5,
                         ui8Dutycycle * ui32Load / 100);
        }
        ROM_SysCtlDelay(100000);
    }
}
```

Task 03:

Youtube Link: https://youtu.be/XrFQZuIVIC0

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"
#include "inc/tm4c123gh6pm.h"

#define PWM_FREQUENCY 55    // 55Hz base frequency

int main(void)
{
    volatile uint32_t ui32Load;  //don't think i need this value
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Dutycycle[3];
    int i, j, k;
    // starting with a duty cycle of 10%
    for(i = 0; i < 3; i++){
        ui8Dutycycle[i] = 10;
    }
    ROM_SysCtlClockSet(
            SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN
                    | SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
// set LEDs to PWM configuration
    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);
    ROM_GPIOPinConfigure(GPIO_PF2_M1PWM6);
    ROM_GPIOPinConfigure(GPIO_PF3_M1PWM7);


    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;
    ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4 | GPIO_PIN_0,
                    GPIO_DIR_MODE_IN);
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4 | GPIO_PIN_0,
                     GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);


    // formula to calculate the number of cycles for the GenPeriod:
    // Load = 1/PWMfrequency * PWMClk - 1.
    // Load is the parameter, PWMfrequency is the desired frequency, PWMClk is the PWM clock frequency based off the system clock
    // so for this task, Load = 1/55 * 625kHz-1 = 11363
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;

    // LED R is controlled by PWM1 Generator 2 while LED G,B are controlled by PWM1 Generator 3
    // configure both generators the same
    ROM_PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
    ROM_PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);

    ROM_PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN);
    ROM_PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, ui32Load);

    // formula for number of cycles in duty cycle:
    // N = %DC * Load
    // N is the parameter, %DC is the percent duty cycle, Load
    // starting with 50% DC: N = 50/100*11364 = 5682
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Dutycycle[0] * ui32Load / 100);
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, ui8Dutycycle[0] * ui32Load / 100);
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, ui8Dutycycle[0] * ui32Load / 100);

    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT | PWM_OUT_6_BIT | PWM_OUT_7_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_3);

    while (1)
    {
        // Pulsewidth adjustments are made using the same formula above.
        // the smallest duty cycle will be 10% and the largest will be 90%
        if (ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4) == 0x00)
        {
            for(i = ui8Dutycycle[0]; i > 10; i--){
                for(j = ui8Dutycycle[1]; j > 10; j--){
                    for(k = ui8Dutycycle[2]; k > 10; k--){
                        ui8Dutycycle[2] = k;
                        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7,
                                ui8Dutycycle[2] * ui32Load / 100);
                        ROM_SysCtlDelay(100000);
                    }
                    ui8Dutycycle[1] = j;
                    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6,
                            ui8Dutycycle[1] * ui32Load / 100);
                    ROM_SysCtlDelay(100000);
                }
                ui8Dutycycle[0] = i;
                ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5,
                        ui8Dutycycle[0] * ui32Load / 100);
                ROM_SysCtlDelay(100000);
            }
        }

        if (ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_0) == 0x00)
        {
            for(i = ui8Dutycycle[0]; i < 90; i++){
                for(j = ui8Dutycycle[1]; j < 90; j++){
                    for(k = ui8Dutycycle[2]; k < 90; k++){
                        ui8Dutycycle[2] = k;
                        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7,
                                ui8Dutycycle[2] * ui32Load / 100);
                        ROM_SysCtlDelay(100000);
                    }
                    ui8Dutycycle[1] = j;
                    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6,
                            ui8Dutycycle[1] * ui32Load / 100);

                    ROM_SysCtlDelay(100000);
                }
                ui8Dutycycle[0] = i;
                ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5,
                        ui8Dutycycle[0] * ui32Load / 100);
                ROM_SysCtlDelay(100000);
            }
        }
    }
}
```