Ivan Soto
Github root directory: https://github.com/sotoi2/Class3.0.4

Date Submitted:

Task 00: Execute provided code

Youtube Link:

```
#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"

// 55Hz to control the servo
#define PWM_FREQUENCY 55

int main(void)
{
    // program the PWM, 83 is the center to create a 1.5ms pulse to the PWM
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 83;


    // run the clk at 40MHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    //pwm module clocked by the sys clk through a divider, (625 khz)
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);


    // enable the pwm1 and gpiod modules (for output on pd0)
    // and gpiof module (for the launchpad buttons on pf0 and pf4)
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);



    // PORT D PIN 0 CONFIGURED as a pwm outputpin for module 1, pwm generator 0
    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);


    // Port F pin 0 and pin 4 are connected to the S2 and S1 switches on the
LaunchPad.
    // In order for the state of the pins to be read in our code, the pins must be
pulled up.
```

```
    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);



    // divide the pwm clock by the desired frequency to determine the count loaded
into the load register
    // config module 1 pwm generator 0
    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);



    // FINAL PWN SETTINGS AND ENABLE IT
    //first line setsthe pulse width
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 100);

    // pwm module 1, gen 0 needs to be enabled as an output and enabled
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0);



    // Read pf4 pin to see if sw1 is pressed
    //
    while(1)
    {

        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
        {
            ui8Adjust--;
            if (ui8Adjust < 56)
            {
                ui8Adjust = 56;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
        }

        //read the pf0 pin to see if sw2 is pressed
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
        {
            ui8Adjust++;
            if (ui8Adjust > 111)
            {
                ui8Adjust = 111;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
        }
        // determines the speed
        ROM_SysCtlDelay(100000);
    }

}
```

Task 01:

Youtube Link:

Modified Schematic (if applicable):

Modified Code:

```
// Insert code here
#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"

// 55Hz to control the servo
#define PWM_FREQUENCY 55

int main(void)
{
    // program the PWM, 83 is the center to create a 1.5ms pulse to the PWM
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 83;


    // run the clk at 40MHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    //pwm module clocked by the sys clk through a divider, (625 khz)
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
    // enable the pwm1 and gpiod modules (for output on pd0)
    // and gpiof module (for the launchpad buttons on pf0 and pf4)
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);



    // PORT D PIN 0 CONFIGURED as a pwm outputpin for module 1, pwm generator 0
    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);


    // Port F pin 0 and pin 4 are connected to the S2 and S1 switches on the
LaunchPad.
    // In order for the state of the pins to be read in our code, the pins must be
pulled up.
    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);



    // divide the pwm clock by the desired frequency to determine the count loaded
into the load register
    // config module 1 pwm generator 0
    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);



    // FINAL PWN SETTINGS AND ENABLE IT
    //first line setsthe pulse width
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 100);

    // pwm module 1, gen 0 needs to be enabled as an output and enabled
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0);



    // Read pf4 pin to see if sw1 is pressed
    //
    while(1)
    {

        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
        {
            ui8Adjust--;
            if (ui8Adjust < 20)
            {
                ui8Adjust = 20;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
        }

        //read the pf0 pin to see if sw2 is pressed
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
        {
            ui8Adjust++;
            if (ui8Adjust > 150)
            {
                ui8Adjust = 150;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
        }



        // determines the speed
        ROM_SysCtlDelay(100000);
    }

}
```

Task 02:

Youtube Link:

Modified Schematic (if applicable):

Modified Code:

// Insert code here

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

Ivan Soto
Github root directory: https://github.com/sotoi2/Class3.0.4

Task 03:

Youtube Link:

Modified Schematic (if applicable):

Modified Code:

```
// Insert code here
```

--

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