Itzel Becerril
Github root directory: https://github.com/HadidBuilds/TivaC_project_labs

Date Submitted: 10/28/19

Task 00: Execute provided code

Youtube Link:
https://www.youtube.com/watch?v=G-Irx6d6MLM

```
#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 to control the servo

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint32_t ui8Adjust;
    ui8Adjust = 83; //center position to create 1.5mS pulse from PWM
    //F=(1/55)*10^3=18.2mS---->18.2/1000=1.82uS*x=1.5mS---->x=83

    //CPU is running at 40MHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);//pwm is clocked by the system clock
through a divider with a range of 2 to 64
    //it will run the PWM clock at 625 kHz, using ROM to reduce code size

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1); //enable PWM1
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD); //enable GPIOD
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable GPIOF

    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0); //Configure port D pin 0
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0); //as a PWM output pin for module 1, pwm
generator 0

    //pins must be pulled up to be used
    //unlock GPIO commit control registerto use PF0
    //PF0 & PF4 are for SW1 and SW2
    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);
//Configures PF0 & PF4 as inputs
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU); //Configures the internal pull-up resistors on both pins
```

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

```
    ui32PWMClock = SysCtlClockGet() / 64; //PWM clock is SYSCLK/64
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1; //divide PWM clock by 55Hz
Frequency to get count to be loaded into load register, sub 1 bc starts at zero
    PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN); //Configure module 1
PWM generator 0 as a down-counter
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);//load the count value

    //make final PWM settings and enable it
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000); //sets
the pulse width
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);//PWM module 1 generator 0 is
enabled as an output
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0); //PWM module 1 is enabled to run

    //runs servo
    while(1)
    {
        //read if PF4(SW1) is pressed
        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);
//loads PWM pulse width register with the new value
        }

        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);
//loads PWM pulse width register with the new value
        }

        ROM_SysCtlDelay(100000); //speed of the loop
    }
}
```

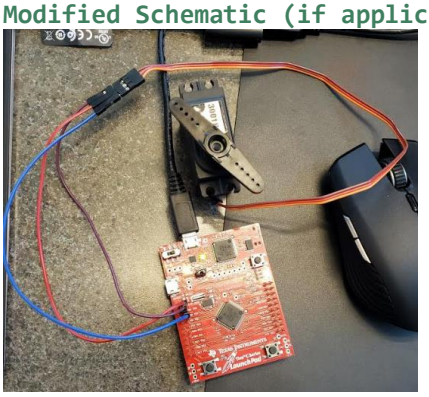
Task 01:

Have servo motor go from 0 to 180 degrees.

Youtube Link:
https://www.youtube.com/watch?v=y-RHj7lMFXA&pbjreload=10

Itzel Becerril
Github root directory: https://github.com/HadidBuilds/TivaC_project_labs

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"

#define PWM_FREQUENCY 50//50Hz base frequency to control the servo

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint32_t ui8Adjust;
    ui8Adjust = 75;

    //CPU is running at 40MHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);//pwm is clocked by the system clock
through a divider with a range of 2 to 64
    //it will run the PWM clock at 625 kHz, using ROM to reduce code size

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1); //enable PWM1
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD); //enable GPIOD
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable GPIOF

    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0); //Configure port D pin 0
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0); //as a PWM output pin for module 1, pwm
generator 0
```

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

```
    //pins must be pulled up to be used
    //unlock GPIO commit control registerto use PF0
    //PF0 & PF4 are for SW1 and SW2
    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);
//Configures PF0 & PF4 as inputs
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU); //Configures the internal pull-up resistors on both pins

    ui32PWMClock = SysCtlClockGet() / 64; //PWM clock is SYSCLK/64
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1; //divide PWM clock by 55Hz
Frequency to get count to be loaded into load register, sub 1 bc starts at zero
    PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN); //Configure module 1
PWM generator 0 as a down-counter
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);//load the count value

    //make final PWM settings and enable it
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000); //sets
the pulse width
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);//PWM module 1 generator 0 is
enabled as an output
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0); //PWM module 1 is enabled to run

    //runs servo
    while(1)
    {
        //read if PF4(SW1) is pressed
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4) == 0x00)
        {
            ui8Adjust--;
            if(ui8Adjust < 50)
            {
                ui8Adjust = 50;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 2000);
//loads PWM pulse width register with the new value
        }

        if(ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_0) == 0x00)
        {
            ui8Adjust++;
            if(ui8Adjust > 100)
            {
                ui8Adjust = 100;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 500);
//loads PWM pulse width register with the new value
        }

        ROM_SysCtlDelay(100000); //speed of the loop
    }
}
```

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

Itzel Becerril
Github root directory: https://github.com/HadidBuilds/TivaC_project_labs

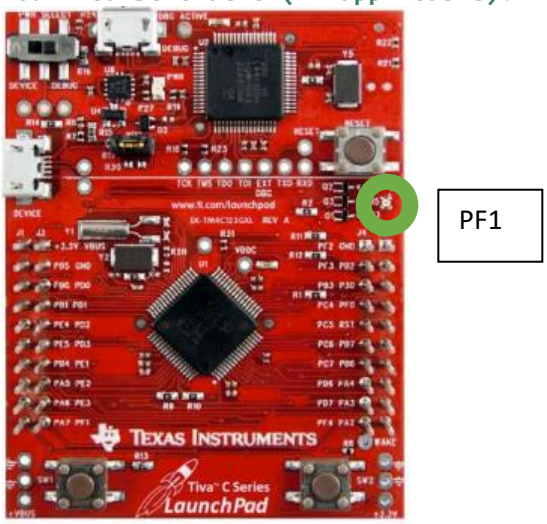
---

Task 02:

Turn brightness of PF1 from 10% to 90% using PWM DC.

Youtube Link:
https://www.youtube.com/watch?v=Sy9WD68nMqc

Modified Schematic (if applicable):



Modified Code:

```
#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 100//100Hz base frequency to control the servo

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint32_t ui8Adjust;
    ui8Adjust = 1;

    //CPU is running at 40MHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);//pwm is clocked by the system clock
through a divider with a range of 2 to 64
    //it will run the PWM clock at 625 kHz, using ROM to reduce code size
```

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

```
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1); //enable PWM1
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD); //enable GPIOD
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable GPIOF

    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1); //Configure port F pin 1
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5); //as a PWM output pin for module 1, pwm
generator 5

    //pins must be pulled up to be used
    //unlock GPIO commit control registerto use PF0
    //PF0 & PF4 are for SW1 and SW2
    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);
//Configures PF0 & PF4 as inputs
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU); //Configures the internal pull-up resistors on both pins

    ui32PWMClock = SysCtlClockGet() / 64; //PWM clock is SYSCLK/64
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1; //divide PWM clock by 55Hz
Frequency to get count to be loaded into load register, sub 1 bc starts at zero
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN); //Configure module 1
PWM generator 0 as a down-counter
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);//load the count value

    //make final PWM settings and enable it
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 100); //sets
the pulse width
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);//PWM module 1 generator 0 is
enabled as an output
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2); //PWM module 1 is enabled to run

    //runs servo
    while(1)
    {
        //read if PF4(SW1) is pressed
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4) == 0x00)
        {
            ui8Adjust--;
            if(ui8Adjust < 10)
            {
                ui8Adjust = 10;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 100);
//loads PWM pulse width register with the new value
        }

        if(ROM_GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_0) == 0x00)
        {
            ui8Adjust++;
            if(ui8Adjust > 90)
            {
                ui8Adjust = 90;
```

```
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 100);
//loads PWM pulse width register with the new value
        }

        ROM_SysCtlDelay(100000); //speed of the loop
    }
}
```

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