Abenezer Namaga
CPE 403 - 1001
TIVAC LAB 06

Github root directory: https://github.com/Ayertena/TIVAC

Date Due: October 9th, 2018

Task 00: Execute the supplied code, no submission required.

Task 01: Change the PWM duty cycle to make the servo motor to do a loop of a complete sweep from 0 to 180 deg.

Youtube Link: https://www.youtube.com/watch?v=WfSe7PtSy_c

```
#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"

// 55-Hz
#define PWM_FREQUENCY 55

int main(void)
{
    // For the PWM variables, ui8Adjust will determine position of
servo motor
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 83;

    // 40-MHz clk
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|
SYSCTL_XTAL_16MHZ);
```

```
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    //Enable PWM1
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD); // Set PWM pin to
Port D
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); // Set Button
pins to Port F

    // Set Pin D0 as PWM output
    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);

    // Unlock GPIO control register
    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;

    // Set PF0 and PF4 pins as inputs
    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 internal system clock by 64 for PWM clock
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1; // Get load count
    PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN); //
Configure PWM generator
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load); // Load ui32Load
as count

    // Set pulse width and enables PWM outputs
    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); // Enable PWM0

    while(1)
    {
        // Check if SW1 was pressed
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
        {
```

```
            // Increase ui8Adjust until minimum limit is reached
            ui8Adjust--;
            if (ui8Adjust < 28) //56-28
            {
                ui8Adjust = 28;
            }
            // Set pulse width
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust *
ui32Load / 1000);
        }
        // Check if SW2 was pressed
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
        {
            // Increase ui8Adjust until max limit is reached
            ui8Adjust++;
            if (ui8Adjust > 139) // 111+28
            {
                ui8Adjust = 139;
            }
            // Set pulse width
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust *
ui32Load / 1000);
        }
        // System delay for servo motor
        ROM_SysCtlDelay(100000);
    }
}
```

Task 02: Change PWM duty cycle from 10% to 90% to control the brightness of the LED at PF1.

Youtube Link: https://www.youtube.com/watch?v=txwGxnDTK-w

```
#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"

// 55-Hz
#define PWM_FREQUENCY 55

int main(void)
{
    // PWM variables, ui8Adjust will determine position of servo motor
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint16_t ui8Adjust;
    ui8Adjust = 83;

    // 40-MHz clock
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|
SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    // Enables PWM1
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); // Button pins
set to Port F

   // Pin F1 is PWM output
```

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

    // Unlock GPIO control register
    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;
    // PF0 and PF4 pins as inputs
    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 internal system clock by 64 for PWM clock
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1; // Get load count
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN); //
Configure PWM generator
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load); // Load ui32Load
as count

    // Sets pulse width and enables PWM outputs
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load /
1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2); // Enables PWM0

    while(1)
    {
        // Checks if SW1 was pressed
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
        {
            // Increases ui8Adjust  until min limit is reached (10%)
            ui8Adjust--;
            if (ui8Adjust < 100)
            {
                ui8Adjust = 100;
            }
            // Sets pulse width
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust *
ui32Load / 1000);
        }
        // Checks if SW2 was pressed
```

```
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
        {
            // Increases ui8Adjust until max limit is reached (90%)
            ui8Adjust++;
            if (ui8Adjust > 900)
            {
                ui8Adjust = 900;
            }
            // Sets pulse width
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust *
ui32Load / 1000);
        }
        // System delay for LED
        ROM_SysCtlDelay(100000);
    }
}
```

Task 03: Change PWM duty cycle from 90% to 10% to control the brightness of the all three LED at PF1, PF2, and PF3 using three nested “for loops”.

Youtube Link: https://www.youtube.com/watch?v=fGqV_40NoMk

```
#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"

// 55-Hz
#define PWM_FREQUENCY 55

int main(void)
{
    // For the PWM variables, ui8Adjust will determine position of
servo motor
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 83;

    // 40-MHz clk
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|
SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

   // Enable PWM1
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); // Set Button
pins to Port F
```

```
    // Set Pin F1 as a PWM output
    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);

    // Divide internal system clock by 64 for PWM clock
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1; //get load count
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN); //PWM
generator configure
    PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load); //load ui32Load
as count
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, ui32Load);

    //Sets pulse width and enables PWM outputs
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5_BIT, ui8Adjust *
ui32Load / 1000);
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6_BIT, ui8Adjust *
ui32Load / 1000);
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7_BIT, ui8Adjust *
ui32Load / 1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT|PWM_OUT_6_BIT|
PWM_OUT_7_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2); //enables PWM0
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_3);

    while(1)
    {
        //90% PWM
        uint16_t a = 900; //RED LED
        uint16_t b = 900; //BLUE LED
        uint16_t c = 900; //GREEN LED

        //Three nested for loops, change to change to 10%
        for(;a>100;--a) //for loops ends with LED being red
        {
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, a * ui32Load /
1000);
            ROM_SysCtlDelay(10000);
            for(;b>100;--b) //LED with be purple without green
```

```
            {
                ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, b *
ui32Load / 1000);
                ROM_SysCtlDelay(10000);
                for(;c>100;--c) //for loop begins with LED being white
                {
                    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, c *
ui32Load / 1000);
                    ROM_SysCtlDelay(10000);
                }
            }
        }
    }
}
```