Jett Guerrero
Github root directory: https://github.com/guerrj1/Advanced-Embedded-Systems

Date Submitted: **12/5/2019**

Task 00: Execute provided code

Youtube Link: https://youtu.be/jp9MZAHWqNQ

Task 01:

Youtube Link: https://youtu.be/Fyuvg3eUZXk

Modified Code:

```
//Lab6 Task 1
#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

int main(void)
{
	volatile uint32_t ui32Load;
	volatile uint32_t ui32PWMClock;
	volatile uint8_t ui8Adjust;
	ui8Adjust = 43;  //starting point

	//clock initialization
	ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MH
Z);
	ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

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

	//GPIO config
	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);
	//pwm config
```

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

```
	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)
	{

		//putton is pressed perform 180 sweep
		if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
		{
			if(ui8Adjust < 155)
			{
				ui8Adjust = 155;
			}
			else
			{
				ui8Adjust = 43;
			}
			ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load /
1000);
		}

		ROM_SysCtlDelay(100000);
	}
}
```

Task 02:

Youtube Link: https://youtu.be/ctTw6ZgHjY4

Modified Code:

```
//Lab6 Task 2
#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

int main(void)
{
	volatile uint32_t ui32Load;
```

```
	volatile uint32_t ui32PWMClock;
	volatile uint8_t ui8Adjust;
	ui8Adjust = 1;

	//clock initialization
	ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MH
Z);
	ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

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

	//GPIO config
	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);

	//pwm config
	ui32PWMClock = SysCtlClockGet() / 64;
	ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
	ROM_PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
	PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);

	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);

	while(1)
	{
		//when pressing button
		//pwm 10%
		if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
		{
			if(ui8Adjust < 10)
			{
				ui8Adjust = 10;
			}
			ui8Adjust--;
			ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load /
1000);
		}

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

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

```
                ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load /
1000);
            }

            ROM_SysCtlDelay(100000);
        }
}
```

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