Joseph Sharp Halpin
CpE 403 Section 1001
10/9/2018

Task 01:
Youtube Link: https://youtu.be/wzHs0uEpBBw

Code:

main.c

```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"


#define PWM_FREQUENCY 55

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 83;

    //set clock rate
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    //set PWM rate
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

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

    //set output PWM pin
    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);

    //allow SW0 and SW1 to operate
    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);

    //get PWM clock to set the period
    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);
```

```
51
52      //initalize the duty cycle and enable the PWM
53      ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
54      ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
55      ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0);
56
57      while(1)
58      {
59          //check if SW1 is pressed and adjust the servo motor
60          if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
61          {
62              ui8Adjust--;
63              if (ui8Adjust < 1)
64              {
65                  ui8Adjust = 1;
66              }
67              ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
68          }
69          //check if SW0 is pressed and adjust the servo motor
70          if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
71          {
72              ui8Adjust++;
73              if (ui8Adjust > 120)
74              {
75                  ui8Adjust = 120;
76              }
77              ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
78          }
79          //delay for 100000 cycles
80          ROM_SysCtlDelay(100000);
81      }
82
83  }
```

Task 02:
Youtube Link: https://youtu.be/6oyUPCQ3CdA

Code:

main.c

```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"

#define PWM_FREQUENCY 55

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    int i;

    //set clock rate
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    //enable PWM1 and GPIOF
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    //set PWM clock rate
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    //configure the PWM and PF1
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);
    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);

    //configure the PWM
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
    PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN);

    //get the period
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;

    //set the PWM period
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, ui32Load);

    //set the PWM duty cycle
    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui32Load * 0.1);


    //enable PWM1
    PWMGenEnable(PWM1_BASE, PWM_GEN_2);
    PWMGenEnable(PWM1_BASE, PWM_GEN_3);

    PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);

    while(1)
    {
        //go from 10% duty cycle to 90%
        for(i = ui32Load * 0.1; i < ui32Load * 0.9; i++)
        {
            //reset the duty cycle
            PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, i);
            ROM_SysCtlDelay(20000);
        }
    }
}
```

Task 03:

Youtube Link: https://youtu.be/-9-JnY1XDV4

Code:

main.c

```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"


#define PWM_FREQUENCY 55

int main(void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    int i, j, k;

    //set clock rate
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    //enable PWM1 and GPIOF
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    //set PWM clock rate
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    //allow PWM to be edited
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;

    //configure the LED pins
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);
    ROM_GPIOPinConfigure(GPIO_PF2_M1PWM6);
    ROM_GPIOPinConfigure(GPIO_PF3_M1PWM7);
    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);

    //configure the PWM
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
    PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN);

    //get the period
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
```

```
    //get the period
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;

    //set the PWM period
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, ui32Load);

    //set the duty cycle of the PWM
    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui32Load * 0.9);
    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, ui32Load * 0.9);
    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, ui32Load * 0.9);

    //enable the PWM
    PWMGenEnable(PWM1_BASE, PWM_GEN_2);
    PWMGenEnable(PWM1_BASE, PWM_GEN_3);

    //set the PWM to output
    PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT | PWM_OUT_6_BIT | PWM_OUT_7_BIT, true);

    while(1)
    {
        //make the red led go from 90% duty cycle to 10%
        for(i = ui32Load * 0.9; i > ui32Load * 0.1; i--)
        {
            PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, i);
            ROM_SysCtlDelay(10000);
            //make the green led go from 90% duty cycle to 10%
            for(j = ui32Load * 0.9; j > ui32Load * 0.1; j--)
            {
                PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, j);
                ROM_SysCtlDelay(10000);
                //make the blue led go from 90% duty cycle to 10%
                for(k = ui32Load * 0.9; k > ui32Load * 0.1; k--)
                {
                    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, k);
                    ROM_SysCtlDelay(10000);
                }
            }
        }
    }
}
```