**DATE SUBMITTED 10/9/18**

**Task 0**

No submission required, ran given code

**Task 1**

No video for this one because forgot to get one in lab and didn’t have the jumper wires at home to resetup the servo

#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)

{

//my variables

int i;

volatile uint32\_t ui32Load;

volatile uint32\_t ui32PWMClock;

volatile uint8\_t ui8Adjust;

ui8Adjust = 83;

//set the clock

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

//enable pwm1 and gpiod

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

//config the PWM

ROM\_GPIOPinTypePWM(GPIO\_PORTD\_BASE, GPIO\_PIN\_0);

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0);

//set s1/s2 on the launchpad, used to move the servo

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);

//set up pwm clock

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);

//set pulse, output state, then enable

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)

{

//controls one direction of the servo

for(i = 30; i < 150; ++i)

{

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_0, i \* ui32Load / 1000);

ROM\_SysCtlDelay(100000);

}

//controls the other direction of the servo

for(i = 150; i > 30; --i)

{

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_0, i \* ui32Load / 1000);

ROM\_SysCtlDelay(100000);

}

ROM\_SysCtlDelay(200000);

}

}

**Youtube Link:** N/A

**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 55

int main(void)

{

//my variables

volatile uint32\_t ui32Load;

volatile uint32\_t ui32PWMClock;

volatile uint8\_t ui8Adjust;

ui8Adjust = 83;

//set the clock

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

//enable pwm1 and gpiod

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

//config the PWM

ROM\_GPIOPinTypePWM(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5);

//set s1/s2 on the launchpad, used to move the servo

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);

//set up pwm clock

ui32PWMClock = SysCtlClockGet() / 64;

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1;

PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_DOWN);

PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_2, ui32Load);

//set pulse, output state, then enable

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)

{

//s1 increase brightness

if(ROM\_GPIOPinRead(GPIO\_PORTF\_BASE,GPIO\_PIN\_4)==0x00)

{

ui8Adjust--;

if (ui8Adjust < 100){ui8Adjust = 100;}

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui8Adjust \* ui32Load / 1000);

}

//s2 decreases brightness

if(ROM\_GPIOPinRead(GPIO\_PORTF\_BASE,GPIO\_PIN\_0)==0x00)

{

ui8Adjust++;

if (ui8Adjust > 900){ui8Adjust = 900;}

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui8Adjust \* ui32Load / 1000);

}

ROM\_SysCtlDelay(100000);

}

}

**Youtube Link:** [**https://youtu.be/2Xn32oFnHBE**](https://youtu.be/2Xn32oFnHBE)

**Task 3**

#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)

{

//my variables

int r, g, b = 100;

volatile uint32\_t ui32Load;

volatile uint32\_t ui32PWMClock;

volatile uint8\_t ui8Adjust;

ui8Adjust = 83;

//set the clock

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

//enable pwm1 and gpiof

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

//config the PWM

ROM\_GPIOPinTypePWM(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

//config the three LEDs

ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5);

ROM\_GPIOPinConfigure(GPIO\_PF2\_M1PWM6);

ROM\_GPIOPinConfigure(GPIO\_PF3\_M1PWM7);

//set up pwm clock

ui32PWMClock = SysCtlClockGet() / 64;

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1;

PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_DOWN);

PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_3, PWM\_GEN\_MODE\_DOWN);

PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_2, ui32Load);

PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_3, ui32Load);

//set pulse, output state, then enable

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui8Adjust \* ui32Load / 1000);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6, ui8Adjust \* ui32Load / 1000);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_7, 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);

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_3);

while(1)

{

for(r; r < 900; r++)

{

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, r \* ui32Load /1000);

ROM\_SysCtlDelay(10000);

for(b; b < 900; b++)

{

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6, b \* ui32Load /1000);

ROM\_SysCtlDelay(10000);

for(g; g < 900; g++)

{

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_7, g \* ui32Load /1000);

ROM\_SysCtlDelay(10000);

}

}

}

for(g; g >= 100; g--)

{

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_7, g \* ui32Load /1000);

ROM\_SysCtlDelay(10000);

}

for(b; b >= 100; b--)

{

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6, b \* ui32Load /1000);

ROM\_SysCtlDelay(10000);

}

for(r; r >= 100; r--)

{

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, r \* ui32Load /1000);

ROM\_SysCtlDelay(10000);

}

}

}

**Youtube Link:** <https://youtu.be/RbAXYszGalw>