**Task 0:**

Execute the supplied code, no submission required.

**#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;

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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);

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)

{

**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);

}

**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);

}

ROM\_SysCtlDelay(100000);

}

}

**Task 1:**

Change the PWM duty cycle to make the servo motor to do a loop of a complete sweep

from 0 to 180 deg.

**#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;

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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);

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);

// set servo to 0 degrees first

ui8Adjust = 1;

**while**(1)

{

// increment PWM adjust by 1

ui8Adjust++;

// if servo goes past 180 degrees, reset to 0

**if**(ui8Adjust > 118)

ui8Adjust = 1;

// apply duty to servo

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_0, ui8Adjust \* ui32Load / 1000);

// allow the PWM to slowly take effect

ROM\_SysCtlDelay(450000);

}

}

**Task 2:**

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

**#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 ui32Period;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint32\_t DUTY\_R = 0;

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5);

// set period to 1000 in order to allow gradual change of color

ui32Period = 1000;

ROM\_PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

ROM\_PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_2, ui32Period);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, DUTY\_R);

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_2);

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_5\_BIT|PWM\_OUT\_6\_BIT|PWM\_OUT\_7\_BIT, true);

**while**(1)

{

// keep looping while R LED is <90%

**while** (DUTY\_R < 900)

{

// increment duty by 0.1

DUTY\_R += 1;

// if R LED reaches 90% duty, reset to 10%

**if**(DUTY\_R > 900)

DUTY\_R = 100;

// LED change delay

**SysCtlDelay**(50000);

// apply duty to R LED

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, DUTY\_R);

}

// reset R LED to 10%

DUTY\_R = 100;

}

}

**Task 3:**

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”.

**#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 ui32Period;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint32\_t DUTY\_R;

**volatile** uint32\_t DUTY\_G;

**volatile** uint32\_t DUTY\_B;

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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);

ROM\_PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

ROM\_PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_3, PWM\_GEN\_MODE\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

ui32Period = 100;

ROM\_PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_2, ui32Period);

ROM\_PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_3, ui32Period);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, DUTY\_R);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6, DUTY\_B);

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

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_2);

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

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_5\_BIT|PWM\_OUT\_6\_BIT|PWM\_OUT\_7\_BIT, true);

**while**(1)

{

// loop through R LED starting at 90% and ending at 10%

**for**(DUTY\_R = 90; DUTY\_R >= 10; DUTY\_R = DUTY\_R - 1)

{

**SysCtlDelay**(1000000);

// loop through B LED starting at 90% and ending at 10%

**for**(DUTY\_B = 90; DUTY\_B >= 10; DUTY\_B = DUTY\_B - 1)

{

**SysCtlDelay**(1000000);

// loop through G LED starting at 90% and ending at 10%

**for**(DUTY\_G = 90; DUTY\_G >= 10; DUTY\_G = DUTY\_G - 1)

{

**SysCtlDelay**(1000000);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, DUTY\_R);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6, DUTY\_B);

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

}

}

}

// reset all RGB LEDs to 90% duty

DUTY\_R = 90;

DUTY\_G = 90;

DUTY\_B = 90;

}

}