**Date Submitted:**

**Task 00: Execute provided code**

**Youtube Link:** [**https://youtu.be/WWb1MMGSWlo**](https://youtu.be/WWb1MMGSWlo)

**#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

**void** **main**(**void**)

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint8\_t ui8Adjust;

ui8Adjust = 83; // Adjusts the position of the servo

// Sets up the system clock and PWM clock

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

// Enables PWM1, GPIOD, GPIOF

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

// Configure PD0 as a PWM

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

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0);

// Unlock PF0 for SW

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;

// Configure pull-up for PF0 and PF4

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);

// Configure 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);

// Adjust width and enable PWM

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 01:**

Youtube Link: [**https://youtu.be/zA6AKIJ3z7A**](https://youtu.be/zA6AKIJ3z7A)

**Modified Code:**

**Changed the code in the while loop. Replaced the *if* structure from task 00 with motor position and delays. The first motor position is 0 degrees followed by a 3-second delay. Then, the second motor position is 165 degrees followed by a 3-second delay. Although instructed to do 180 degrees, the PowerHD 3001 Servo can only go to 165 degrees max as specified by the datasheet.**

**#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

**void** **main**(**void**)

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint8\_t ui8Adjust;

ui8Adjust = 83; // Adjusts the position of the servo

// Sets up the system clock and PWM clock

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

// Enables PWM1, GPIOD, GPIOF

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

// Configure PD0 as a PWM

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

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0);

// Unlock PF0 for SW

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;

// Configure pull-up for PF0 and PF4

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);

// Configure 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);

// Adjust width and enable PWM

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)

{

// 0 degrees

ui8Adjust = 44;

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

ROM\_SysCtlDelay(40000000); // 3-second delay

// 165 degrees [Maximum angle specified by PowerHD 30001HB datasheet]

ui8Adjust = 121;

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

ROM\_SysCtlDelay(40000000); // 3-second delay

}

}

**------------------------------------------------------------------------------------**

**Task 02:**

Youtube Link: [**https://youtu.be/K-zCS-PIG7g**](https://youtu.be/K-zCS-PIG7g)

**Modified Code:**

**Using task 00 as a template, the code was modified to enable PF1’s PWM. Instead of PD0’s M1PWM0 and generator 0, PF1 will be using M1PWM5 and generator 2. For the PWM output control logic, instead of port 0, port 5 is used to output the PWM of PF1. Modify all PWM functions to fit these specifications. Finally, refine the delay function and increment/decrement ui16Adjust in the *if* structure for the LED to increase/decrease brightness with a 10-90% duty-cycle.**

**#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

**void** **main**(**void**)

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint16\_t ui16Adjust;

ui16Adjust = 1136; // Adjusts the brightness of the LED

// Sets up the system clock and PWM clock

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

// Enables PWM1, GPIOD, GPIOF

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

// Configure PF1 as a PWM

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

ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5);

// Unlock PF0 for SW

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;

// Configure pull-up for PF0 and PF4

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);

// Configure 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);

// Adjust width and enable PWM

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui16Adjust);

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_5\_BIT, **true**);

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

**while**(1)

{

// Decrease DC

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

{

ui16Adjust = ui16Adjust - 5;

// Min DC = 10%

**if** (ui16Adjust < 1136)

{

ui16Adjust = 1136;

}

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui16Adjust);

}

// Increase DC

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

{

ui16Adjust = ui16Adjust + 5;

// Max DC = 90%

**if** (ui16Adjust > 10225)

{

ui16Adjust = 10225;

}

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui16Adjust);

}

ROM\_SysCtlDelay(10000);

}

}

**------------------------------------------------------------------------------------**

**Task 03:**

Youtube Link:

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

**------------------------------------------------------------------------------------**