**Date Submitted:**

**Task 00: Execute provided code**

**Youtube Link:** <https://www.youtube.com/watch?v=v26aU9PYP_4>

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

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=TlIYjeFe_jQ>

**Modified Schematic (if applicable):**

**Modified Code:**

**#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 // PWM frequency at 55Hz

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

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint8\_t ui8Adjust;

ui8Adjust = 35; // value represents 1ms of PWM ON time (servo at 0 or 180 degrees)

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64); // PWM clock is set to 625kHz after div by 64

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1); // enable PWM 1 peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD); // enable GPIO PORTD to use as output for servo motor

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // enable GPIO PORTF for SW1 and SW2

ROM\_GPIOPinTypePWM(GPIO\_PORTD\_BASE, GPIO\_PIN\_0); // set PORTD as a PWM output pin

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0); // use PWM motion control module 1

//unlock SW2

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); // set SW1 and SW2 as inputs

ROM\_GPIOPadConfigSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU); // use switches as pull-up

//setup of PWM 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);

//setup of PWM ON width size

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_0, ui8Adjust \* ui32Load / 1000); // PWM at 1.5ms (center position)

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_0\_BIT, **true**); // set generator 0 as output

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_0); // enable pwm and generator 0

**while**(1)

{

//continuous sweep from 0 to 180 degrees

**while** (ui8Adjust < 135)

{

ui8Adjust++;

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

ROM\_SysCtlDelay(100000); // speed of servo motor

}

ROM\_SysCtlDelay(2000000); // small delay in between sweep

**while** (ui8Adjust > 35) {

ui8Adjust--;

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

ROM\_SysCtlDelay(100000); // speed of servo motor

}

ROM\_SysCtlDelay(2000000); // small delay in between sweep

}

}

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

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=lfwiizhAhv8>

**Modified Schematic (if applicable):**

**Modified Code:**

**#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 // PWM frequency at 55Hz

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

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint32\_t ui32Adjust;

ui32Adjust = 900; // value represents PWM ON time (duty cycle of LED - 90%)

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64); // PWM clock is set to 625kHz after div by 64

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1); // enable PWM 1 peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD); // enable GPIO PORTD for PWM

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // enable GPIO PORTF for LED pin

ROM\_GPIOPinTypePWM(GPIO\_PORTD\_BASE, GPIO\_PIN\_0); // set PORTD as a PWM output pin

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0); // use PWM motion control module 1

ROM\_GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1); // set PORTF as output

//setup of PWM 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);

//setup of PWM ON width size

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_0, ui32Adjust \* ui32Load / 1000);

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_0\_BIT, **true**); // set generator 0 as output

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_0); // enable pwm and generator 0

**while**(1)

{

// if PWM is low, turn off LED

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

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0);

}

// else turn on LED during the ON time of the PWM

**else** {

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2);

}

}

}

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

**Task 03:**

Youtube Link: <https://www.youtube.com/watch?v=MrM6ME3crvs>

**Modified Schematic (if applicable):**

**Modified Code:**

**#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/adc.h"

**#include** "driverlib/pin\_map.h"

**#include** "inc/hw\_gpio.h"

**#include** "driverlib/rom.h"

**#define** PWM\_FREQUENCY 55 // PWM frequency at 55Hz

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

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint32\_t ui32Adjust;

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64); // PWM clock is set to 625kHz after div by 64

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1); // enable PWM 1 peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD); // enable GPIO PORTD to use as output for DC motor

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE); // enable GPIO PORTE using analog input 8 (PE5)

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0); //enable ADC0 peripheral

ROM\_GPIOPinTypePWM(GPIO\_PORTD\_BASE, GPIO\_PIN\_0); // set PORTD as a PWM output pin

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0); // use PWM motion control module 1

ROM\_GPIOPinTypeADC(GPIO\_PORTE\_BASE, GPIO\_PIN\_5); // use PE5 (AIN8 - channel 8) for potentiometer

//setup of PWM 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);

//setup of PWM

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_0\_BIT, **true**); // set generator 0 as output

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_0); // enable pwm and generator 0

//initialize variables ADC for potentiometer

uint32\_t ui32ADC0Value[4]; // array to store samples of ADC with 4 steps

**volatile** uint32\_t ui32ADCAvg; // store avg value

//configure ADC

ROM\_ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0); // using ADC sample sequencer 1 (SS1), set as the highest priority, and processor will trigger ADC

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_CH8); // ADC sample step 0

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_CH8); // ADC sample step 1

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 2, ADC\_CTL\_CH8); // ADC sample step 2

ROM\_ADCSequenceStepConfigure(ADC0\_BASE,1, 3, ADC\_CTL\_CH8|ADC\_CTL\_IE|ADC\_CTL\_END); //ADC sample step 3, set ADC interrupt flag, end sampling

ROM\_ADCSequenceEnable(ADC0\_BASE, 1); // enable ADC0

**while**(1)

{

ROM\_ADCIntClear(ADC0\_BASE, 1); // clear ADC interrupt

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1); // processor begins to trigger ADC

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 1, **false**)) // wait for ADC conversion..

{

}

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value); // get ADC value from samples

ui32ADCAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;

ui32Adjust = ui32ADCAvg; // store ADC avg value into the ui32Adjust variable

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_0, ui32Adjust \* ui32Load / 1000); // set the width of the PWM using the ui32Adjust value - DC motor speed set

}

}

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