**Date Submitted: 10/28/19**

**Task 00: Execute provided code**

**Youtube Link:**

<https://www.youtube.com/watch?v=G-Irx6d6MLM>

**#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 //55Hz base frequency to control the servo

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

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint32\_t ui8Adjust;

ui8Adjust = 83; //center position to create 1.5mS pulse from PWM

//F=(1/55)\*10^3=18.2mS---->18.2/1000=1.82uS\*x=1.5mS---->x=83

//CPU is running at 40MHz

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);//pwm is clocked by the system clock through a divider with a range of 2 to 64

//it will run the PWM clock at 625 kHz, using ROM to reduce code size

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1); //enable PWM1

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD); //enable GPIOD

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //enable GPIOF

ROM\_GPIOPinTypePWM(GPIO\_PORTD\_BASE, GPIO\_PIN\_0); //Configure port D pin 0

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0); //as a PWM output pin for module 1, pwm generator 0

//pins must be pulled up to be used

//unlock GPIO commit control registerto use PF0

//PF0 & PF4 are for SW1 and 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); //Configures PF0 & PF4 as inputs

ROM\_GPIOPadConfigSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU); //Configures the internal pull-up resistors on both pins

ui32PWMClock = **SysCtlClockGet**() / 64; //PWM clock is SYSCLK/64

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1; //divide PWM clock by 55Hz Frequency to get count to be loaded into load register, sub 1 bc starts at zero

**PWMGenConfigure**(PWM1\_BASE, PWM\_GEN\_0, PWM\_GEN\_MODE\_DOWN); //Configure module 1 PWM generator 0 as a down-counter

**PWMGenPeriodSet**(PWM1\_BASE, PWM\_GEN\_0, ui32Load);//load the count value

//make final PWM settings and enable it

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

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_0\_BIT, true);//PWM module 1 generator 0 is enabled as an output

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_0); //PWM module 1 is enabled to run

//runs servo

**while**(1)

{

//read if PF4(SW1) is pressed

**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); //loads PWM pulse width register with the new value

}

**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); //loads PWM pulse width register with the new value

}

ROM\_SysCtlDelay(100000); //speed of the loop

}

}

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

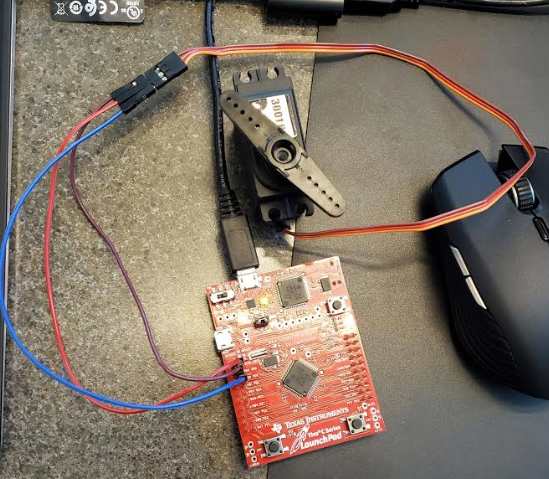
**Task 01:**

**Have servo motor go from 0 to 180 degrees.**

Youtube Link:

<https://www.youtube.com/watch?v=y-RHj7lMFXA&pbjreload=10>

**Modified Schematic (if applicable):**



**Modified Code:**

**// Insert code here**

**#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 50//50Hz base frequency to control the servo

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

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint32\_t ui8Adjust;

ui8Adjust = 75;

//CPU is running at 40MHz

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);//pwm is clocked by the system clock through a divider with a range of 2 to 64

//it will run the PWM clock at 625 kHz, using ROM to reduce code size

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1); //enable PWM1

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD); //enable GPIOD

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //enable GPIOF

ROM\_GPIOPinTypePWM(GPIO\_PORTD\_BASE, GPIO\_PIN\_0); //Configure port D pin 0

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0); //as a PWM output pin for module 1, pwm generator 0

//pins must be pulled up to be used

//unlock GPIO commit control registerto use PF0

//PF0 & PF4 are for SW1 and 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); //Configures PF0 & PF4 as inputs

ROM\_GPIOPadConfigSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU); //Configures the internal pull-up resistors on both pins

ui32PWMClock = **SysCtlClockGet**() / 64; //PWM clock is SYSCLK/64

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1; //divide PWM clock by 55Hz Frequency to get count to be loaded into load register, sub 1 bc starts at zero

**PWMGenConfigure**(PWM1\_BASE, PWM\_GEN\_0, PWM\_GEN\_MODE\_DOWN); //Configure module 1 PWM generator 0 as a down-counter

**PWMGenPeriodSet**(PWM1\_BASE, PWM\_GEN\_0, ui32Load);//load the count value

//make final PWM settings and enable it

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

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_0\_BIT, true);//PWM module 1 generator 0 is enabled as an output

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_0); //PWM module 1 is enabled to run

//runs servo

**while**(1)

{

//read if PF4(SW1) is pressed

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

{

ui8Adjust--;

**if**(ui8Adjust < 50)

{

ui8Adjust = 50;

}

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_0, ui8Adjust \* ui32Load / 2000); //loads PWM pulse width register with the new value

}

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

{

ui8Adjust++;

**if**(ui8Adjust > 100)

{

ui8Adjust = 100;

}

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_0, ui8Adjust \* ui32Load / 500); //loads PWM pulse width register with the new value

}

ROM\_SysCtlDelay(100000); //speed of the loop

}

}

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

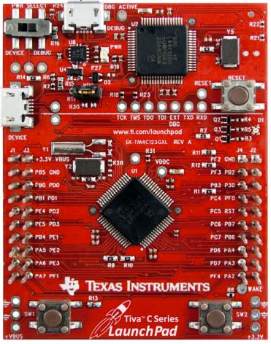
**Task 02:**

Turn brightness of PF1 from 10% to 90% using PWM DC.

Youtube Link:

<https://www.youtube.com/watch?v=Sy9WD68nMqc>

**Modified Schematic (if applicable):**



PF1

**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 100//100Hz base frequency to control the servo

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

{

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint32\_t ui8Adjust;

ui8Adjust = 1;

//CPU is running at 40MHz

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);//pwm is clocked by the system clock through a divider with a range of 2 to 64

//it will run the PWM clock at 625 kHz, using ROM to reduce code size

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1); //enable PWM1

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD); //enable GPIOD

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //enable GPIOF

ROM\_GPIOPinTypePWM(GPIO\_PORTF\_BASE, GPIO\_PIN\_1); //Configure port F pin 1

ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5); //as a PWM output pin for module 1, pwm generator 5

//pins must be pulled up to be used

//unlock GPIO commit control registerto use PF0

//PF0 & PF4 are for SW1 and 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); //Configures PF0 & PF4 as inputs

ROM\_GPIOPadConfigSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU); //Configures the internal pull-up resistors on both pins

ui32PWMClock = **SysCtlClockGet**() / 64; //PWM clock is SYSCLK/64

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1; //divide PWM clock by 55Hz Frequency to get count to be loaded into load register, sub 1 bc starts at zero

**PWMGenConfigure**(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_DOWN); //Configure module 1 PWM generator 0 as a down-counter

**PWMGenPeriodSet**(PWM1\_BASE, PWM\_GEN\_2, ui32Load);//load the count value

//make final PWM settings and enable it

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui8Adjust \* ui32Load / 100); //sets the pulse width

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_5\_BIT, true);//PWM module 1 generator 0 is enabled as an output

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_2); //PWM module 1 is enabled to run

//runs servo

**while**(1)

{

//read if PF4(SW1) is pressed

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

{

ui8Adjust--;

**if**(ui8Adjust < 10)

{

ui8Adjust = 10;

}

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui8Adjust \* ui32Load / 100); //loads PWM pulse width register with the new value

}

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

{

ui8Adjust++;

**if**(ui8Adjust > 90)

{

ui8Adjust = 90;

}

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui8Adjust \* ui32Load / 100); //loads PWM pulse width register with the new value

}

ROM\_SysCtlDelay(100000); //speed of the loop

}

}

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