Enrique Saldana

<https://github.com/enri10>

**Date Submitted: 11/01/18**

**Task 00: Execute the supplied code, no submission required.**

**LAB06 Task00 :** [**https://youtu.be/JUPFVbQ5wbY**](https://youtu.be/JUPFVbQ5wbY)

**Task 01: Change the PWM duty cycle to make the servo motor to do a loop of a complete sweep from 0 to 180 deg.**

**LAB06 Task01 :** [**https://youtu.be/SjWgACQkBX0**](https://youtu.be/SjWgACQkBX0)

**#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"

//50Hz base frequency to control servo(20ms period)

**#define** PWM\_FREQUENCY 50

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

{

//Variables used to program the PWM

//20 is the 0 degree position

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint8\_t ui8Adjust;

ui8Adjust = 20;

//Run CPU at 40MHz

//Run PWM clock at 40MHz/64 = 625kHz

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

//Enable PWMI, GPIOD,and GPIOF modules

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

//Configure PDO as a PWM output pin for module 1

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

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0);

//Configure module 1 PWM generator 0 as a down-counter and load the count value

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-width

//PWM module 1, generator 0 is enable as an output

//Enabled to run

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)

{

//Sweep of 0 to 180 in increments of 0.1 ms

//When servo reaches 180, it returns to 0 degree

**if** (ui8Adjust > 115)

{

ui8Adjust = 20;

}

**else**

{

ui8Adjust = ui8Adjust +5;

}

// //Load the PWM pulse width register with the new value

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

//Speed of the loop

ROM\_SysCtlDelay(3000000);

}

}

**Task 02: Change PWM duty cycle from 10% to 90% to control the brightness of the LED at PF1.**

**LAB06 Task02 :** [**https://youtu.be/axFRau8zu58**](https://youtu.be/axFRau8zu58)

**Used Generator 2 in order to use PF1 as a PWM output**

**The values of 10 and 90 will give me duty cycles of 10% and 90% respectively**

**#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"

//55Hz base frequency to control servo

**#define** PWM\_FREQUENCY 55

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

{

//Variables used to program the PWM

//Start LED at around 50% duty cycle

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint8\_t ui8Adjust;

ui8Adjust = 50;

//Run CPU at 40MHz

//Run PWM clock at 40MHz/64 = 625kHz

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

//Enable PWMI,and GPIOF modules

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

//Configure PF1(Red LED) as output

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

//Configure PF1 as a PWM output pin for module 1

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

ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5);

//Unlock the GPIO commit control register

//Configure PF0 and PF4 as inputs

//Configure the internal pull-up resistors on both pins

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);

//Configure module 1 PWM generator 2 as a down-counter and load the count value

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-width

//PWM module 1, generator 2 is enabled as an output

//Enabled to run

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)

{

//Read PF4(SW1)

//ui8Adjust is decremented until it reaches 10% duty cycle

//Load the PWM pulse width register with the new value

**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 / 1000);

}

//Read PF0(SW2)

//Pulse width is incremented until it reaches 90% duty cycle

//Load the 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 / 1000);

}

//Speed of the loop

ROM\_SysCtlDelay(50000);

}

}

**Task 03: 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”.**

**LAB06 Task03 :** [**https://youtu.be/tnsuxeBblYQ**](https://youtu.be/tnsuxeBblYQ)

**Used generator 3 in order to use PF2 and PF3 as PWM outputs**

**#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"

//55Hz base frequency to control servo

**#define** PWM\_FREQUENCY 55

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

{

//Variables used to program the PWM

//Start LED at around 90% duty cycle

**volatile** uint32\_t ui32Load;

**volatile** uint32\_t ui32PWMClock;

**volatile** uint8\_t ui8Adjust;

ui8Adjust = 90;

//Variables to individually control each LED

**volatile** uint8\_t red;

**volatile** uint8\_t green;

**volatile** uint8\_t blue;

//Run CPU at 40MHz

//Run PWM clock at 40MHz/64 = 625kHz

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

//Enable PWMI,and GPIOF modules

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

//Configure LEDs as outputs

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

//Configure PF1,PF2,and PF3 as a PWM output pins for module 1

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);

//Unlock the GPIO commit control register

//Configure PF0 and PF4 as inputs

//Configure the internal pull-up resistors on both pins

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);

//Configure module 1 PWM generator 2 as a down-counter and load the count value

//Configure module 1 PWM generator 3 as a down-counter and load the count value

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);

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

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

//Set pulse-width for the 3 LED's

//PWM module 1, generator 2 and 3 are enabled as outputs

//Enabled to run

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)

{

//Set RGB values to 90% to output a bright white light

red = 90;

green =90;

blue = 90;

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

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

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

//Delay to have bright white light visible for a couple seconds

ROM\_SysCtlDelay(50000000);

//Nested loop that will cycle RGB LEDS from 90% to 10%

//Delays are there to slow down the change from 90% to 10%

**for**(red=90;red>10; red--)

{

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

ROM\_SysCtlDelay(200);

**for**(green=90; green>10; green--)

{

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

ROM\_SysCtlDelay(200);

**for**(blue=90; blue>10; blue--)

{

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

ROM\_SysCtlDelay(200);

}

}

}

//Delay to have the dull white light stay for a couple seconds

ROM\_SysCtlDelay(50000000);

}

}