Abenezer Namaga

CPE 403 – 1001

TIVAC LAB 06

Github root directory: https://github.com/Ayertena/TIVAC

Date Due: October 9th, 2018

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

----------------------------------------------------------------------

**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.

Youtube Link: https://www.youtube.com/watch?v=WfSe7PtSy\_c

#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"

// 55-Hz

#define PWM\_FREQUENCY 55

int main(void)

{

// For the PWM variables, ui8Adjust will determine position of servo motor

volatile uint32\_t ui32Load;

volatile uint32\_t ui32PWMClock;

volatile uint8\_t ui8Adjust;

ui8Adjust = 83;

// 40-MHz clk

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

//Enable PWM1

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD); // Set PWM pin to Port D

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // Set Button pins to Port F

// Set Pin D0 as PWM output

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

ROM\_GPIOPinConfigure(GPIO\_PD0\_M1PWM0);

// Unlock GPIO control register

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;

// Set PF0 and PF4 pins as inputs

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);

// Divide internal system clock by 64 for PWM clock

ui32PWMClock = SysCtlClockGet() / 64;

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1; // Get load count

PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_0, PWM\_GEN\_MODE\_DOWN); // Configure PWM generator

PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_0, ui32Load); // Load ui32Load as count

// Set pulse width and enables PWM outputs

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); // Enable PWM0

while(1)

{

// Check if SW1 was pressed

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

{

// Increase ui8Adjust until minimum limit is reached

ui8Adjust--;

if (ui8Adjust < 28) //56-28

{

ui8Adjust = 28;

}

// Set pulse width

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

}

// Check if SW2 was pressed

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

{

// Increase ui8Adjust until max limit is reached

ui8Adjust++;

if (ui8Adjust > 139) // 111+28

{

ui8Adjust = 139;

}

// Set pulse width

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

}

// System delay for servo motor

ROM\_SysCtlDelay(100000);

}

}

----------------------------------------------------------------------

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

Youtube Link: https://www.youtube.com/watch?v=txwGxnDTK-w

#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"

// 55-Hz

#define PWM\_FREQUENCY 55

int main(void)

{

// PWM variables, ui8Adjust will determine position of servo motor

volatile uint32\_t ui32Load;

volatile uint32\_t ui32PWMClock;

volatile uint16\_t ui8Adjust;

ui8Adjust = 83;

// 40-MHz clock

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

// Enables PWM1

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // Button pins set to Port F

// Pin F1 is PWM output

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

ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5);

// Unlock GPIO control register

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;

// PF0 and PF4 pins as inputs

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);

// Divide internal system clock by 64 for PWM clock

ui32PWMClock = SysCtlClockGet() / 64;

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1; // Get load count

PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_DOWN); // Configure PWM generator

PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_2, ui32Load); // Load ui32Load as count

// Sets pulse width and enables PWM outputs

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); // Enables PWM0

while(1)

{

// Checks if SW1 was pressed

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

{

// Increases ui8Adjust until min limit is reached (10%)

ui8Adjust--;

if (ui8Adjust < 100)

{

ui8Adjust = 100;

}

// Sets pulse width

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui8Adjust \* ui32Load / 1000);

}

// Checks if SW2 was pressed

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

{

// Increases ui8Adjust until max limit is reached (90%)

ui8Adjust++;

if (ui8Adjust > 900)

{

ui8Adjust = 900;

}

// Sets pulse width

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui8Adjust \* ui32Load / 1000);

}

// System delay for LED

ROM\_SysCtlDelay(100000);

}

}

————————————————————————————————————————————————————————————————————

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”.

Youtube Link: <https://www.youtube.com/watch?v=fGqV_40NoMk>

#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"

// 55-Hz

#define PWM\_FREQUENCY 55

int main(void)

{

// For the PWM variables, ui8Adjust will determine position of servo motor

volatile uint32\_t ui32Load;

volatile uint32\_t ui32PWMClock;

volatile uint8\_t ui8Adjust;

ui8Adjust = 83;

// 40-MHz clk

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

ROM\_SysCtlPWMClockSet(SYSCTL\_PWMDIV\_64);

// Enable PWM1

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM1);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // Set Button pins to Port F

// Set Pin F1 as a PWM output

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);

// Divide internal system clock by 64 for PWM clock

ui32PWMClock = SysCtlClockGet() / 64;

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1; //get load count

PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_DOWN); //PWM generator configure

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

PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_2, ui32Load); //load ui32Load as count

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

//Sets pulse width and enables PWM outputs

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5\_BIT, ui8Adjust \* ui32Load / 1000);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6\_BIT, ui8Adjust \* ui32Load / 1000);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_7\_BIT, 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); //enables PWM0

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

while(1)

{

//90% PWM

uint16\_t a = 900; //RED LED

uint16\_t b = 900; //BLUE LED

uint16\_t c = 900; //GREEN LED

//Three nested for loops, change to change to 10%

for(;a>100;--a) //for loops ends with LED being red

{

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

ROM\_SysCtlDelay(10000);

for(;b>100;--b) //LED with be purple without green

{

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

ROM\_SysCtlDelay(10000);

for(;c>100;--c) //for loop begins with LED being white

{

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

ROM\_SysCtlDelay(10000);

}

}

}

}

}