**Lab 4**

**Task 1**

//This program will give insight on the use of interrupts and timers on the TIVA C board

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/tm4c123gh6pm.h" //interrupt definitions and register assignments

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

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

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

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

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

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

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

{

uint32\_t ui32Period;

//Sets clock to run at 40 MHz

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN);

//Sets the clock to 40MHz => 400MHz (PLL) / (5\*2)

//Configures the GPIO

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

//Configures Timer0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0);

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC);

//Generates a period for the GPIO to run at 10MHz => (SystemClockSpeed/DesiredSpeed)/2

ui32Period = (**SysCtlClockGet**() / 10) / 2;

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32Period -1); //subtract period by 1 since it starts at 0

//Enables the interrupt for TIMER0

**IntEnable**(INT\_TIMER0A);

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);

**IntMasterEnable**();

//Enables TIMER

**TimerEnable**(TIMER0\_BASE, TIMER\_A);

**while** (1)

{}

}

**void** **Timer0IntHandler**(**void**)

//This is the interrupt handler that will be called when the Timer reaches the value specified

{

// Clear the timer interrupt

**TimerIntClear**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);

// Read the current state of the GPIO pin and

// write back the opposite state

**if**(**GPIOPinRead**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2))

//Checks if there is a value written to PORTF.2, write 0 if there is

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);

}

**else**

//else write 4 to PORTF to set PORTF.2 to 1

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4);

}

}

**Task 2**

//This program will give insight on the use of interrupts and timers on the TIVA C board

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/tm4c123gh6pm.h" //interrupt definitions and register assignments

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

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

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

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

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

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

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

{

uint32\_t ui32Period;

//Sets clock to run at 40 MHz

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN);

//Sets the clock to 40MHz => 400MHz (PLL) / (5\*2)

//Configures the GPIO

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

//Configures Timer0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0);

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC);

//Generates a period for the GPIO to run at 50MHz => (SystemClockSpeed/DesiredSpeed)/(duty cycle)

ui32Period = (**SysCtlClockGet**() / 50) / 5; //This is set to create a 50Hz pulse with 20% duty cycle

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32Period -1); //subtract period by 1 since it starts at 0

//Enables the interrupt for TIMER0

**IntEnable**(INT\_TIMER0A);

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);

**IntMasterEnable**();

//Enables TIMER

**TimerEnable**(TIMER0\_BASE, TIMER\_A);

**while** (1)

{}

}

**void** **Timer0IntHandler**(**void**)

//This is the interrupt handler that will be called when the Timer reaches the value specified

{

// Clear the timer interrupt

**TimerIntClear**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);

// Read the current state of the GPIO pin and

// write back the opposite state

**if**(**GPIOPinRead**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2))

//Checks if there is a value written to PORTF.2, write 0 if there is

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);

}

**else**

//else write 4 to PORTF to set PORTF.2 to 1

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4);

}

}