**Timers and Interrupts Lab**

**Links to videos:**

**Task 1:** http://screencast.com/t/PuxUt1ug8

**Task 2:** http://screencast.com/t/y7YjZ2L3JjJo

**Task 3:** http://screencast.com/t/1f7CCWK6RW

**Task 1: Adding comments to original code**

//video: http://screencast.com/t/PuxUt1ug8

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/tm4c123gh6pm.h"

**#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(**void**)

{

uint32\_t ui32Period; //will store number to initialize counter

//set the clock to run at 40MHz

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

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //Enable port F

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1| GPIO\_PIN\_2|GPIO\_PIN\_3); //Enable pins F1,F2,F3 as outputs

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0); //Enable timer 0

TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); //Configure timer 0 to periodic mode

//get system clock, divide by desired blinking frequency, and divide by 2 because interrupt will happen twice per cycle

ui32Period = (SysCtlClockGet() / 10) / 2; //will blink at 10 Hz

TimerLoadSet(TIMER0\_BASE, TIMER\_A, ui32Period - 1); //load timer with value in ui32Period

IntEnable(INT\_TIMER0A); //Enable interrupts on timer 0

TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //set timer 0 to interrupt at timeout

IntMasterEnable(); //enable master interrupt

TimerEnable(TIMER0\_BASE, TIMER\_A); //start the timer

**while**(1) //infinite loop

{

}

}

**void** Timer0IntHandler(**void**) //interrupt handler for timer 0

{

//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))

{

//turn on pin

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

}

**else**

{

//turn off pin

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);

}

}

**Task 2: Changing toggle of GPIO to 50 Hz at 50% duty cycle.**

//video: http://screencast.com/t/y7YjZ2L3JjJo

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/tm4c123gh6pm.h"

**#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(**void**)

{

uint32\_t ui32Period; // will store number to initialize counter

//set the clock to run at 40MHz

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

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //Enable port F

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1| GPIO\_PIN\_2|GPIO\_PIN\_3); //Enable pins F1,F2,F3 as outputs

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0); //Enable timer 0

TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); //Configure timer 0 to periodic mode

//get system clock, divide by desired blinking frequency, and divide by 2 because interrupt will happen twice per cycle

ui32Period = (SysCtlClockGet() / 50) / 2; //will blink at 50 Hz

TimerLoadSet(TIMER0\_BASE, TIMER\_A, ui32Period - 1); //load timer with value in ui32Period

IntEnable(INT\_TIMER0A); //Enable interrupts on timer 0

TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //set timer 0 to interrupt at timeout

IntMasterEnable(); //enable master interrupt

TimerEnable(TIMER0\_BASE, TIMER\_A); //start the timer

**while**(1) //infinite loop

{

}

}

**void** Timer0IntHandler(**void**) //interrupt handler for timer 0

{

//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))

{

//turn on pin

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

}

**else**

{

//turn off pin

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);

}

}

**Task 3: Include GPIO interrupt to Task 02 from switch SW2 to turn ON and OFF the LED for 0.5 sec. The toggle should of the GPIO is suspended when executing the interrupt. Normal operation of Task 02 should begin after the switch event.**

//video: http://screencast.com/t/1f7CCWK6RW

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/tm4c123gh6pm.h"

**#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"

**void** IntGPIOF0(**void**);

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

{

uint32\_t ui32Period; //will store number to initialize counter

//set the clock to run at 40MHz

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

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //Enable port F

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1| GPIO\_PIN\_2|GPIO\_PIN\_3); //Enable pins F1,F2,F3 as outputs

GPIOPinTypeGPIOInput(GPIO\_PORTF\_BASE, GPIO\_PIN\_0); //Enable PF0 as input

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0); //Enable timer 0

TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); //Configure timer 0 to periodic mode

//get system clock, divide by desired blinking frequency, and divide by 2 because interrupt will happen twice per cycle

ui32Period = (SysCtlClockGet() / 50) / 2; //will blink at 50 Hz

TimerLoadSet(TIMER0\_BASE, TIMER\_A, ui32Period - 1); //load timer with value in ui32Period

//Unlock Pin F0 to use an interrupt on SW2

SYSCTL\_RCGC2\_R |= 0x00000020; // 1) activate clock for Port F

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // 2) unlock GPIO Port F

GPIO\_PORTF\_CR\_R = 0x1F; // allow changes to PF4-0

// only PF0 needs to be unlocked, other bits can't be locked

GPIO\_PORTF\_AMSEL\_R = 0x00; // 3) disable analog on PF

GPIO\_PORTF\_PCTL\_R = 0x00000000; // 4) PCTL GPIO on PF4-0

GPIO\_PORTF\_DIR\_R = 0x0E; // 5) PF4,PF0 in, PF3-1 out

GPIO\_PORTF\_AFSEL\_R = 0x00; // 6) disable alt funct on PF7-0

GPIO\_PORTF\_PUR\_R = 0x11; // enable pull-up on PF0 and PF4

GPIO\_PORTF\_DEN\_R = 0x1F; // 7) enable digital I/O on PF4-0

GPIOIntRegister(GPIO\_PORTF\_BASE, IntGPIOF0); //register the interrupt handler for PF0

GPIOIntTypeSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_FALLING\_EDGE); //SW2 goes low when pressed

GPIOIntEnable(GPIO\_PORTF\_BASE, GPIO\_PIN\_0); //enable interrupts on PF0

IntEnable(INT\_TIMER0A); //Enable interrupts on timer 0

TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //set timer 0 to interrupt at timeout

IntMasterEnable(); //enable master interrupt

TimerEnable(TIMER0\_BASE, TIMER\_A); //start the timer

**while**(1) //infinite loop

{

}

}

**void** Timer0IntHandler(**void**) //interrupt handler for timer 0

{

//Clear the timer interrupt

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

//Read the current state of the GPIO pin and

//write back the opposite statee

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

{

//turn on pin

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

}

**else**

{

//turn off pin

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);

}

}

**void** IntGPIOF0(**void**) //interrupt handler for GPIO pin F0

{

GPIOIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_0); //clear interrupt flag on pin F0

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4); //turn ON blue LED

SysCtlDelay(20000000); //keep LED on for 0.5 seconds

}