TivaC Lab 4 – Timers and Interrupts

CPE 403

**Checklist for Lab 3**

* A text/word document of the initial code with comments
* In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also include the comments.
* Provide a permanent link to all main and dependent source code files only (name them as LabXX-TYY, XX-Lab# and YY-task#)Screenshots of debugging process along with pictures of actual circuit
* Video link of demonstration.

**Code for Experiment**

**Task 1:**

**#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**() {

uint32\_t ui32Period;

// Use 40 MHz Clock

**SysCtlClockSet**(

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

// Enable Port F

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

// Set Port F as output

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE,

GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3);

// Timer configuration

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0); // enable TIMER0

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configure TIMER0 as 32 bit timer

// Calculate and set delay

ui32Period = (**SysCtlClockGet**() / 10) / 2; // 50% DC

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32Period - 1);

// Enable interrupt

**IntEnable**(INT\_TIMER0A); // enable vector associated with TIMER0A

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); // Enable event to generate interrupt

**IntMasterEnable**(); // Master int enable for all interrupts

// Enable the timer

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

**while** (1)

;

}

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

// Clear the timer interrupt

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

// Read and write states of Pins

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

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

} **else** {

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

}

}

**Task 2:**

**#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**() {

uint32\_t ui32Period;

// Use 40 MHz Clock

**SysCtlClockSet**(

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

// Enable Port F

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

// Set Port F as output

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE,

GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3);

// Timer configuration

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0); // enable TIMER0

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configure TIMER0 as 32 bit timer

// Calculate and set delay

ui32Period = (**SysCtlClockGet**() / 50) / 2; // 50% DC

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32Period - 1);

// Enable interrupt

**IntEnable**(INT\_TIMER0A); // enable vector associated with TIMER0A

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); // Enable event to generate interrupt

**IntMasterEnable**(); // Master int enable for all interrupts

// Enable the timer

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

**while** (1)

;

}

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

// Clear the timer interrupt

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

// Read and write states of Pins

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

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

} **else** {

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

}

}

**Task 3:**

**#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"

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

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/rom\_map.h"

**void** **IntSwitch2Handler**();

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

uint32\_t ui32Period;

// Use 40 MHz Clock

**SysCtlClockSet**(

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

// Enable Port F

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

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE,

GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3); // LED Pins

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_LOCK) = GPIO\_LOCK\_KEY;

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_CR) = 0x1;

**GPIOPinTypeGPIOInput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0); // SW2 for input

// Timer configuration

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0); // enable clock to TIMER0

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configure TIMER0 as 32 bit timer

// Calculate and set delay

ui32Period = (**SysCtlClockGet**() / 10) / 2; // set the period to 10Hz, 50% DC

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32Period - 1);

// Enable interrupt

**IntEnable**(INT\_TIMER0A); // enable vector for TIMER0A

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); // Enable timer A, timeout mode

**IntMasterEnable**(); // Enable all interrupts

// Enable the timer

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

**IntEnable**(INT\_GPIOF);

**GPIOIntTypeSet**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_FALLING\_EDGE);

**GPIOIntEnable**(GPIO\_PORTF\_BASE, GPIO\_INT\_PIN\_0); // enable on port F0

**IntMasterEnable**();

**while** (1)

;

}

**void** **IntSwitch2Handler**(){

**GPIOIntClear**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0);

// Read and write states

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

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

**SysCtlDelay**(7000000);

}

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

// Clear the timer interrupt

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

// Read and write states

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

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

} **else** {

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

}

}

**Video Link to Demo**

[**Task**](https://www.youtube.com/watch?v=6Nrp7cJtleM) **2 and 3:** [**https://www.youtube.com/watch?v=QyYhrPlrXe4**](%20https:/www.youtube.com/watch?v=QyYhrPlrXe4%20)