Clinton Bess

Dr. Venkatesan Muthukumar

# TivaC Lab 4

*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 2 and 3:** https://www.youtube.com/watch?v=QyYhrPlrXe4