Adrian Ruiz

CPE 403-1001

Lab 4

Task 00:

Code:

```
/*
 * Adrian Ruiz CpE 403  Lab4
 */

#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)
{
    //used to generate a certain duty cycle
    uint32_t ui32Period;

    //Set clock to run at 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);


    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    //Set Pin 1,2,3 of PortF as outputs
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //Enable Timer 0 and Configure Timer 0 as a 32 bit Full-Width Periodic Timer
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    //Calculate desired frequency and duty cycle wanted.
    ui32Period = (SysCtlClockGet() / 10) / 2; //Get the period
    //Load value (set Period) count up to ui32Period - 1
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    //Specify which interrupt to be enable in this case TIMER0A
    IntEnable(INT_TIMER0A);
    // Enable event to generate interrupt
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    //Allow Processor to Respond to Interrupts
    IntMasterEnable();
```

```
    while(1)
    {
    }
}

void Timer0IntHandler(void)
{
    // 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))
    {
        //Write Low to pins 1,2,3 of Port F
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        //Write High to Pin 2 of PORTF
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
    }
}
```

Task 01:

Snip of Altered code:

```
//set the period to 2Hz
ui32Period = (SysCtlClockGet() / 2) / 2; //Get the period
```

To generate a 2Hz clock, I divided the System clock by 2 and by 2 again to get a 2Hz 50% duty cycle.

```
//Set clock to run at 5.3MHz in order to get a Delay of about .375 seconds
SysCtlClockSet(SYSCTL_SYSDIV_38|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
```

To get a .375 delay need for the duty cycle, I changed the clock to run at 5.3 Mhz in order to get StyCtlDelay(2000000) to equal about .375.

```
//Generate a 75% duty Cycle
SysCtlDelay(2000000);
```

By calling this inside the Timer interrupt when the clock cycle is high, the duty cycle gets shifted to about 75%.

Code:

```
 * Adrian Ruiz CpE 403  Lab4
 */

#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)
{
    //used to generate a certain duty cycle
    uint32_t ui32Period;

    //Set clock to run at 5.3MHz in order to get a Delay of about .375 seconds
    SysCtlClockSet(SYSCTL_SYSDIV_38|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);


    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    //Set Pin 1,2,3 of PortF as outputs
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //Enable Timer 0 and Configure Timer 0 as a 32 bit Full-Width Periodic Timer
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    //set the period to 2Hz
    ui32Period = (SysCtlClockGet() / 2) / 2; //Get the period
    //Load value (set Period) count up to ui32Period - 1
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    //Specify which interrupt to be enable in this case TIMER0A
    IntEnable(INT_TIMER0A);
    // Enable event to generate interrupt
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    //Allow Processor to Respond to Interrupts
    IntMasterEnable();
```

```
    //Enable operations of Timer0
    TimerEnable(TIMER0_BASE, TIMER_A);

    while(1)
    {
    }
}

void Timer0IntHandler(void)
{
    // 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))
    {
        //Write Low to pins 1,2,3 of Port F
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        //Write High to Pin 2 of PORTF
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);

        //Generate a 75% duty Cycle
        SysCtlDelay(2000000);
    }
}
```

Task 02:

Altered Code:

```
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) = 0x1;
```

This unlocks the pin 0 from portF, SW2, in order to be used as an input.

```
//GPIO_PIN_4, SW2, configured to be an input
    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE,GPIO_PIN_0);

    //Enable switch interrupt
    IntEnable(INT_GPIOF);

    //How the interrupt will be called and enabling event for interrupt
    GPIOIntTypeSet(GPIO_PORTF_BASE,GPIO_PIN_0, GPIO_FALLING_EDGE);
    GPIOIntEnable(GPIO_PORTF_BASE,GPIO_INT_PIN_0);
```

This snippet sets SW2 to be an input and enables the GPIO Interrupt. SW2 becomes a the trigger for the interrupt.

```
void GPIO_PF_IntHandler(void)
{
```

```
    IntMasterDisable();

    //Clear the GPIOPin0
    GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_0);

    //Enable Timer1 and configure it as a 32 periodic timer
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    TimerLoadSet(TIMER1_BASE, TIMER_A, SysCtlClockGet() -0.5);
    TimerEnable(TIMER1_BASE, TIMER_A);

    //Write Low to all LEDs and turn on a single LED
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,0);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 8);

    //creat a 1.5 sec delay
    int i = 0;
    for(i = 0; i< 3; ++i)
    {
        TimerIntEnable(TIMER1_BASE,TIMER_TIMA_TIMEOUT);
        while(1)
        {

if(TimerIntStatus(TIMER1_BASE,true)&TIMER_TIMA_TIMEOUT==TIMER_TIMA_TIMEOUT)
            {
                TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
                break;
            }
        }
    }

    TimerDisable(TIMER1_BASE, TIMER_A);

    IntMasterEnable();
}
```

This is the GPIO Interrupt Handler. When the button is pressed, it disables the blinking LED and makes the Green LED blink for about 1.5 seconds.

Code:

```
/*
 * Adrian Ruiz CpE 403  Lab4
 * I left the blinking part in. The driver libraries wouldn't
 * link properly if i tried taking it out
 */

#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"


int main(void)
{
    //used to generate a certain duty cycle
    uint32_t ui32Period;

    //Set clock to run at 40MHz

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MA
IN);


    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    //Set Pin 1,2,3 of PortF as outputs
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);


    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) = 0x1;

    //GPIO_PIN_4, SW2, configured to be an input
    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE,GPIO_PIN_0);

    //Enable switch interrupt
    IntEnable(INT_GPIOF);

    //How the interrupt will be called and enabling event for interrupt
    GPIOIntTypeSet(GPIO_PORTF_BASE,GPIO_PIN_0, GPIO_FALLING_EDGE);
    GPIOIntEnable(GPIO_PORTF_BASE,GPIO_INT_PIN_0);


    //Enable Timer 0 and Configure Timer 0 as a 32 bit Full-Width Periodic
Timer
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    //set the period to 2Hz
    ui32Period = (SysCtlClockGet() / 2) / 2; //Get the period
    //Load value (set Period) count up to ui32Period - 1
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    //Specify which interrupt to be enable in this case TIMER0A
    IntEnable(INT_TIMER0A);
    // Enable event to generate interrupt
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    //Allow Processor to Respond to Interrupts
    IntMasterEnable();

    //Enable operations of Timer0
    TimerEnable(TIMER0_BASE, TIMER_A);

    while(1)
    {
    }
```

```
}

void Timer0IntHandler(void)
{
    // 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))
    {
        //Write Low to pins 1,2,3 of Port F
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        //Write High to Pin 2 of PORTF
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);

    }
}

void GPIO_PF_IntHandler(void)
{
    IntMasterDisable();

    //Clear the GPIOPin0
    GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_0);

    //Enable Timer1 and configure it as a 32 periodic timer
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    TimerLoadSet(TIMER1_BASE, TIMER_A, SysCtlClockGet() -0.5);
    TimerEnable(TIMER1_BASE, TIMER_A);

    //Write Low to all LEDs and turn on a single LED
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,0);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 8);

    //creat a 1.5 sec delay
    int i = 0;
    for(i = 0; i< 3; ++i)
    {
        TimerIntEnable(TIMER1_BASE,TIMER_TIMA_TIMEOUT);
        while(1)
        {

if(TimerIntStatus(TIMER1_BASE,true)&TIMER_TIMA_TIMEOUT==TIMER_TIMA_TIMEOUT)
            {
                TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
                break;
            }
        }
    }

    TimerDisable(TIMER1_BASE, TIMER_A);
```

```
    IntMasterEnable();
}
```