**Date Submitted: 10/4/2018**

**Task 00: Execute provided code**

Youtube Link: <https://youtu.be/lH26UfPrlPM>

**------------------------------------------------------------------------------------**

**Task 01:**

Youtube Link: <https://youtu.be/eYt_HcNnOEg>

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

**#include** <stdint.h> //include necessary libraries

**#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; //variable to compute timer delays

//the line below sets a 40MHz system clock

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

//the two lines below enable the GPIO peripheral and configure the pins connected to the LEDs as outputs

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

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

//the two lines below enables the clock to the peripheral

//and configures Timer0 as a 32-bit timer in periodic mode

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

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

/\*divide SysCtlClockGet() by desired frequency. 2Hz

Divide that by 4 to get a 25% duty cycle, multiply by 3 for 75%

\*/

ui32Period = ((**SysCtlClockGet**() / 2) / 4)\*3;

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

**IntEnable**(INT\_TIMER0A);//enables the specific vector associated with Timer0A

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);//enables a specific event within the timer to generate an interrupt

**IntMasterEnable**();//master interrupt enable API for all interrupts

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

**while**(1) //stay in this loop until the ISR is triggered

{

}

}

**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))

{

//the line below turns all pins off

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

}

**else**

{

//the line below turns the blue LED on

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

}

}

**------------------------------------------------------------------------------------**

**Task 02:**

Youtube Link:

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**#include** <stdint.h> //include necessary libraries

**#include** <stdbool.h>

**#include** <string.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/ssi.h"

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

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

**#include** "stdlib.h"

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

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

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

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

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

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

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

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

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

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

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

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

{

uint32\_t ui32Period; //variable to compute timer delays

HWREG(GPIO\_PORTF\_BASE+GPIO\_O\_LOCK) = GPIO\_LOCK\_KEY; //this unlocks SW2 so we can use it

HWREG(GPIO\_PORTF\_BASE+GPIO\_O\_CR) |= GPIO\_PIN\_0; //this is the pin for the button

//the line below sets a 40MHz system clock

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

//the two lines below enable the GPIO peripheral and configure the pins connected to the LEDs as outputs

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

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

//the two lines below enables the clock to the peripheral

//and configures Timer0 as a 32-bit timer in periodic mode

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

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

/\*divide SysCtlClockGet() by desired frequency. 2Hz

Divide that by 4 to get a 25% duty cycle, multiply by 3 for 75%

\*/

ui32Period = ((**SysCtlClockGet**() / 2) / 4)\*3;

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

IntEnable(INT\_TIMER0A);//enables the specific vector associated with Timer0A

TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);//enables a specific event within the timer to generate an interrupt

IntMasterEnable();//master interrupt enable API for all interrupts

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

**while**(1) //stay in this loop until the ISR is triggered

{

}

}

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

{

uint32\_t currentState = 0;

uint32\_t status = 0;

status = **GPIOIntStatus**(BUttonBase,true);

**GPIOIntClear**(ButtonBase,status);

**if**(status & ButtonInt == ButtonInt)

{

TIMER0\_CTL\_R = 0; //turn the timer off

TimerIntClear(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //clear the interrupt caused by timer

**GPIOPinWrite**(GPIO\_PORTF\_BASE,GPIO\_PIN\_1|GPIO\_PIN\_2| GPIO\_PIN\_3, 0); //turn off all LEDS

}

// Read the current state of the GPIO pin and

// write back the opposite state

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

{

//the line below turns all pins off

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

}

**else**

{

//the line below turns the blue LED on

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

}

**void** **timer1A\_delaySec**(**int** ttime)

{

**int** i;

SYSCTL\_RCGCTIMER\_R |= 2; //enable clock to timer block 1

TIMER1\_CTL\_R = 0; //disable timer before initialization

TIMER1\_CFG\_R = 0x04 //16-bit option

TIMER1\_TAMR\_R = 0x02; //periodic mode and down counter

TIMER1\_TAILR\_R = 64000 -1; //TImerA interval load value reg

TIMER1\_TAPR\_R = 250 -1; //TimerA prescaler 16MHz/250 = 64000Hz

TIMER1\_ICR\_R = 0x1; //clear the TimerA timeout flag

TIMER1\_CTL\_R |= 0X01; //enable Timer A after initialization

**for**(**int** i = 0;, i<ttime;i++)

{

**while**((TIMER1\_RIS\_R & 0x1) == 0)

; //wait for TimerA timeout flag

TIMER1\_ICR\_R = 0x1; //clear the TimerA timeout flag

}

}

**------------------------------------------------------------------------------------**