**Date Submitted:**

**https://youtu.be/win3PqiOCKk**

**Task 00: Execute provided code**

**#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; //variable to compute timer delays

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN); // Set clock to 40MHz

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // enable GPIO peripherals

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3); // configure pins as outputs for LEDs

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

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configure Timer 0 in periodic mode

ui32Period = (**SysCtlClockGet**() / 10) / 2; // sets the delay

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32Period -1); // load into Timer's Interval Load register

**IntEnable**(INT\_TIMER0A); // enables specific vector associated with Timer 0A

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

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

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

**while**(1)

{

}

}

**void** **Timer0IntHandler**(**void**) // Add an interrupt handler, clears the interrupt source, and toggles GPIO pin based on state.

{

// 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))

{

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

}

**else**

{

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

}

}

**Youtube Link: https://youtu.be/win3PqiOCKk**

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

**Task 01:**

Youtube Link: https://youtu.be/XA-lexjGoll

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

**#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"

uint32\_t ui32PeriodOn; //variable to compute timer delays

uint32\_t ui32PeriodOff;

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

{

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN); // Set clock to 40MHz

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // enable GPIO peripherals

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3); // configure pins as outputs for LEDs

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

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configure Timer 0 in periodic mode

ui32PeriodOn = (**SysCtlClockGet**() / 10) / 2.3;// sets the delay

ui32PeriodOff = (**SysCtlClockGet**() / 10) / 1.75;

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOn -1); // load into Timer's Interval Load register

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOff -1);

**IntEnable**(INT\_TIMER0A); // enables specific vector associated with Timer 0A

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

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

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

**while**(1)

{

}

}

**void** **Timer0IntHandler**(**void**) // Add an interrupt handler, clears the interrupt source, and toggles GPIO pin based on state.

{

// 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))

{

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOff);

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

}

**else**

{

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOn);

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

}

}

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

**Task 02:**

Youtube Link: https://youtu.be/VqCIDMWYXs0

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

// Insert code here

**#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"

uint32\_t ui32PeriodOn; //variable to compute timer delays

uint32\_t ui32PeriodOff;

uint32\_t ui32timer1On;

uint32\_t ui32timer1Off;

**void** **Timer1IntHandler**(**void**) //

{

**TimerIntClear**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // clear the int

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

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, **SysCtlClockGet**()); // loads first timer back

// GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0); // turns off led'

**GPIOIntClear**(GPIO\_PORTF\_BASE, GPIO\_INT\_PIN\_4);

}

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

{

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN); // Set clock to 40MHz

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // enable GPIO peripherals

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3); // configure pins as outputs for LEDs

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

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configure Timer 0 in periodic mode

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1); // enable clock to timer1

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); // configure timer 1 in periodic mode

// functions for enabling the switch button, telling it to go to the handler

**GPIOPinTypeGPIOInput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4);

**GPIOPadConfigSet**(GPIO\_PORTF\_BASE ,GPIO\_PIN\_4,GPIO\_STRENGTH\_2MA,GPIO\_PIN\_TYPE\_STD\_WPU);

**GPIOIntTypeSet**(GPIO\_PORTF\_BASE,GPIO\_PIN\_4,GPIO\_FALLING\_EDGE);

**GPIOIntRegister**(GPIO\_PORTF\_BASE,Timer1IntHandler);

**GPIOIntEnable**(GPIO\_PORTF\_BASE, GPIO\_INT\_PIN\_4);

ui32PeriodOn = (**SysCtlClockGet**() / 10) / 2.3;// sets the delay

ui32PeriodOff = (**SysCtlClockGet**() / 10) / 1.75;

ui32timer1On = (**SysCtlClockGet**() / 1) / 2;

ui32timer1Off = (**SysCtlClockGet**() / 1) / 2 ;

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOn -1); // load into Timer's Interval Load register

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOff -1);

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, ui32timer1On -1);

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32timer1Off- 1);

**IntEnable**(INT\_TIMER0A); // enables specific vector associated with Timer 0A

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

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

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

**while**(1)

{

}

}

**void** **Timer0IntHandler**(**void**) // Add an interrupt handler, clears the interrupt source, and toggles GPIO pin based on state.

{

// 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))

{

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOff);

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

}

**else**

{

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOn);

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

}

}

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

**Task 03:**

Youtube Link:

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

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