**Date Submitted: 9/28/2019**

**Task 00: Execute provided code**

**Youtube Link:**

<https://youtu.be/BWhUm6MkkHk>

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

**Task 01:**

Verification:



Youtube Link:

<https://youtu.be/eCEWkqNf93I>

**Modified Schematic (if applicable):**

**Modified 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"

uint32\_t ui32PeriodHigh;

uint32\_t ui32PeriodLow;

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

{

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

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

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

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

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

ui32PeriodHigh = (**SysCtlClockGet**() / 10) \* 0.43;

ui32PeriodLow = (**SysCtlClockGet**() / 10) \* 0.57;

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodHigh -1);

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

**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))

{

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodLow -1);

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

}

**else**

{

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodHigh -1);

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

}

}

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

**Task 02:**

Could not use SW2 because of NMI default making it locked. Tried to unlock it but it ended up making the code not work so I used SW1 instead.

Verification:



Youtube Link:

<https://youtu.be/Wteja5E3ERY>

**Modified Schematic (if applicable):**

**Modified 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"

uint32\_t ui32PeriodHigh;

uint32\_t ui32PeriodLow;

uint32\_t ui32Delay\_1s;

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

{

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

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

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

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

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

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

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

**IntEnable**(INT\_GPIOF);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

ui32Delay\_1s = (**SysCtlClockGet**());

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

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

ui32PeriodHigh = (**SysCtlClockGet**() / 10) \* 0.43;

ui32PeriodLow = (**SysCtlClockGet**() / 10) \* 0.57;

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodHigh -1);

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

**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))

{

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodLow -1);

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

}

**else**

{

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodHigh -1);

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

}

}

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

{

**TimerIntClear**(TIMER1\_BASE, TIMER\_A);

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

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

}

**void** **PortFPin4IntHandler**(**void**)

{

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

// Clear the GPIO interrupt

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

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

// Call TIMER 1 Delay

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

**IntMasterEnable**();

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, ui32Delay\_1s);

**IntEnable**(INT\_TIMER1A);

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

**TimerEnable**(TIMER1\_BASE, TIMER\_A);

}