**Date Submitted: 9/30/2019**

**Task 01:**

Youtube Link: <https://youtu.be/25beMAUtVZw>

**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;

uint32\_t ui32PeriodOff;

**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);

//set up button

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

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

ui32PeriodOn = (**SysCtlClockGet**() / 10)\*.43; //variable to determine on time

ui32PeriodOff = (**SysCtlClockGet**() / 10)\*.57; //variable to determine off time

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOff -1); //loading timer to enter interrupt

**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))

{

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

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

}

**else**

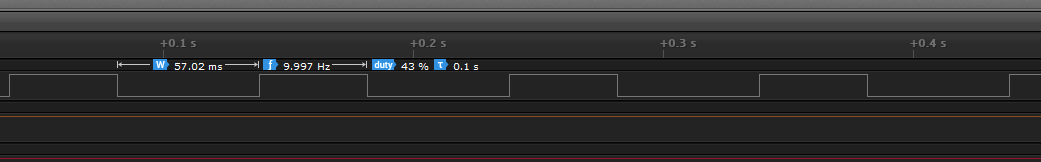
{

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

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOn -1);

}

}



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

**Task 02:**

Youtube Link: <https://youtu.be/j7UtwZqETt0>

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

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

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

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

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

uint32\_t ui32PeriodOn;

uint32\_t ui32PeriodOff;

**void** **Timer1ADelay**(**int** ttime){ //delays by .5seconds

**int** i;

SYSCTL\_RCGCTIMER\_R |=2;

TIMER1\_CTL\_R=0;

TIMER1\_CFG\_R=0x04;

TIMER1\_TAMR\_R=0x02;

TIMER1\_TAILR\_R=64000-1;

TIMER1\_TAPR\_R=625-1;

TIMER1\_ICR\_R=0x1;

TIMER1\_CTL\_R|=0x1;

**for**(i=0;i<ttime;i++){

**while**((TIMER1\_RIS\_R &0x01)==0); //wait for timeout

TIMER1\_ICR\_R = 0x1; //clear flag

}

}

**void** **ButtonInterrupt**(**void**){

**int** status = 0; //used to check which pin's interrupt is triggered (best practice)

status =**GPIOIntStatus**(GPIO\_PORTF\_BASE,true);

**GPIOIntClear**(GPIO\_PORTF\_BASE,GPIO\_INT\_PIN\_0); //Clears that pin's interrupt

**if**(status & GPIO\_INT\_PIN\_0){

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

Timer1ADelay(5); //delay for 1 sec

}

**SysCtlDelay**(1000000); //debouncing?

}

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

{

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

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

HWREG(GPIO\_PORTF\_BASE+GPIO\_O\_LOCK) = GPIO\_LOCK\_KEY;

HWREG(GPIO\_PORTF\_BASE+GPIO\_O\_CR) |= GPIO\_PIN\_0;

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

//set up button

**GPIOPinTypeGPIOInput**(GPIO\_PORTF\_BASE,GPIO\_PIN\_0); //sets button pin as input

**GPIOPadConfigSet**(GPIO\_PORTF\_BASE,GPIO\_PIN\_0,GPIO\_STRENGTH\_2MA,GPIO\_PIN\_TYPE\_STD\_WPU); //configures button settings

**GPIOIntTypeSet**(GPIO\_PORTF\_BASE,GPIO\_PIN\_0,GPIO\_RISING\_EDGE); //Determines what signal triggers interrupt

//GPIOIntRegister(GPIO\_PORTF\_BASE,ButtonInterrupt);//Associates pin group with interrupt(?)

**GPIOIntEnable**(GPIO\_PORTF\_BASE,GPIO\_INT\_PIN\_0); //enables pin interrupt

**IntEnable**(INT\_GPIOF);

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

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

ui32PeriodOn = (**SysCtlClockGet**() / 10)\*.43; //variable to determine on time

ui32PeriodOff = (**SysCtlClockGet**() / 10)\*.57; //variable to determine off time

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOff -1); //loading timer to enter interrupt

**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))

{

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

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

}

**else**

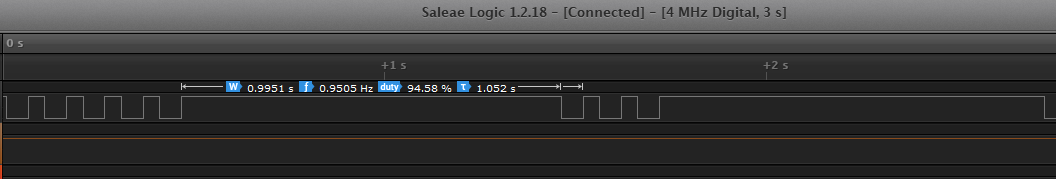
{

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

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOn -1);

}

}



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