**Date Submitted: 9/28**

**Task 00: Execute provided code**

**Youtube Link:** [**https://youtu.be/YAtDtCXSb-U**](https://youtu.be/YAtDtCXSb-U)

**#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"

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

{

uint32\_t ui32Period = 0;

// Set up the clock f = 40 MHz

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

// Set up the GPIO

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

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

// Set up TIMER0

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

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

// (40 MHz / 10 Hz) \* 50% DC

ui32Period = (**SysCtlClockGet**() / 10) / 2;

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

// Enable interrupts

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

// Start TIMER0A

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

**while**(1)

{

}

}

// Timer0 ISR

**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);

}

**else**

{

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

}

}

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

**Task 01:**

Youtube Link: <https://youtu.be/0EMcB5y-u9s>

**Modified Code:**

**The primary change is having two period variables [ui32PeriodHigh and ui32PeriodLow]. The ui32PeriodHigh is when the wave is high for 43 ms whereas ui32PeriodLow is when the wave us low [ui32PeriodLow = T – ui32PeriodHigh = 100 ms – 43 ms = 57 ms. These variables will set the Timer0 in the ISR where the control statements are located.**

**#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 = 0;

uint32\_t ui32PeriodLow = 0;

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

{

// Set up the clock f = 40 MHz

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

// Set up the GPIO

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

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

// Set up TIMER0

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

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

// (40 MHz / 10 Hz) \* 43% DC

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

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

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

// Enable interrupts

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

// Start TIMER0A

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

**while**(1)

{

}

}

// Timer0 ISR

**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, ui32PeriodLow - 1);

}

**else**

{

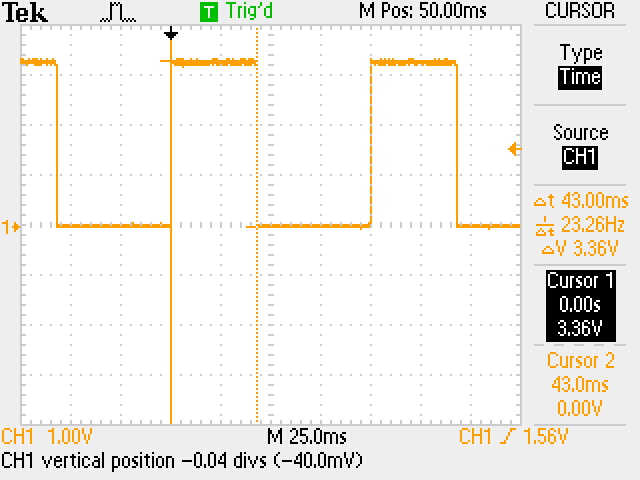
**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4);

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

}

}

**Measured waveform using the oscilloscope:**

****

**The measured wave has a period of 4 x 25 ms = 100 ms which is 10 Hz as expected. Using the cursor, the measured time when the wave is set to high is 43 ms which is the 43% duty cycle of the entire wave.**

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

**Task 02:**

Youtube Link: <https://youtu.be/tL6CFz75OXk>

**Modified Code:**

**First change is creating a function [void timer1Adelay(int delayT)] for the 1 second delay using Timer1. Since the clock runs at 40 MHz, Timer1 will have a value of 40E6 – 1 for one second. The inputs to the function will be the number of seconds of delay [in this task, delayT = 1 for 1 second]. Next change is setting up SW2 ISR which is the same procedure as in task00, except with a switch and the requirement of unlocking PF0.**

**#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 ui32PeriodHigh = 0;

uint32\_t ui32PeriodLow = 0;

// Timer1 delay function where delayT is the number of seconds to delay.

**void** **timer1A\_delay**(**int** delayT)

{

**int** i;

**TimerDisable**(TIMER1\_BASE, TIMER\_A); // Disable Timer1A

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); // Configure Timer1 to be periodic

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, 39999999); // 1 Hz

TIMER1\_ICR\_R = 0x1; // Clear the Timer1A timeout flag

**TimerEnable**(TIMER1\_BASE, TIMER\_A); // Enable Timer1A

**for**(i = 0; i < delayT; i++)

{

**while**((TIMER1\_RIS\_R & 0x1) == 0); // Check for timer-out

TIMER1\_ICR\_R = 0x1; // Clear the Timer1A timeout flag

}

}

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

{

// Set up the clock f = 40 MHz

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

// Set up the GPIO

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // Configure the PF GPIO\_Reg

// Unlock and lock the GPIO\_PF0

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

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_CR) |= 0x01;

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_LOCK) = 0;

// PF1, PF2, PF3 set to output for LEDS

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

// Configure PF0 as input for SW2

**GPIOPinTypeGPIOInput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0);

// Set PF0 as a weak pull-up

**GPIOPadConfigSet**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0,

GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

// Set up interrupt for PF0 [SW2]

**GPIOIntEnable**(GPIO\_PORTF\_BASE, GPIO\_INT\_PIN\_0);

// Detect input on rising edge

**GPIOIntTypeSet**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_RISING\_EDGE);

// Set up TIMER0

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

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

// Set up TIMER1

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

// (40 MHz / 10 Hz) \* 43% DC

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

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

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

// Enable interrupts

**IntEnable**(INT\_TIMER0A); // TIMER0A

**IntEnable**(INT\_GPIOF); // Enable the GPIOF interrupt

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

**IntMasterEnable**();

// Start TIMER0A

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

**while**(1)

{

}

}

// Timer0 ISR

// Toggles the the LED for f= 10 Hz with DC = 43%

**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, ui32PeriodLow - 1);

}

**else**

{

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

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

}

}

// SW2 ISR

// 1) Suspend Timer0A to prevent LED from toggling

// 2) Use Timer1 to turn on the LED for 1 sec

// 3) Re-enable Timer0 ISR

**void** **SW2IntHandler**(**void**)

{

**GPIOIntClear**(GPIO\_PORTF\_BASE, GPIO\_INT\_PIN\_0); // Clear the GPIO interrupt

**TimerDisable**(TIMER0\_BASE, TIMER\_A); // Disable Timer0A

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3, 4); // Turn on the LED

// Call Timer1 delay function for 1 second

timer1A\_delay(1);

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3, 0); // Turn off the LED

**TimerEnable**(TIMER0\_BASE, TIMER\_A); // Re-enable Timer0A

}