**Date Submitted: 09/28/2019**

**Task 00:**

**Youtube Link:**

<https://youtu.be/2tAWkruLQ1E>

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

**Task 01:**

Youtube Link:

<https://youtu.be/vHafU0sSxRk>

**Modified Code:**

#include <stdint.h>

#include <stdbool.h>

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\tm4c123gh6pm.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\hw\_memmap.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\hw\_types.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\sysctl.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\interrupt.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\gpio.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\timer.h"

void configureTimer0();

void configureTimer1A();

void PORTFPin4IntHandler();

int main(void)

{

// Clock Setup

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

// GPIO Configuration

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

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

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

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

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

IntEnable(INT\_GPIOF);

configureTimer0();

while (1)

{

}

}

void configureTimer0()

{

// Timer Configuration

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0); // Must call before calling peripheral specific driverlib function, or else Fault ISR

IntMasterEnable();

TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configures Timer 0 as a 32-bit timer in periodic mode (combines Timer 0A and 0B)

/\* Delay Calculation

\*

\* To toggle GPIO at 2hz and 75% Duty cycle

\* You need to generate an interrupt at 50% of the desired period.

\* Steps:

\* 1.) Call SysCltClockGet()

\* 2.) Divide that by desired frequency

\* 3.) Divide the whole thing by the duty cycle

\*

\* This calculation is loaded into the timer's INTERVAL LOAD RESISTER using TimerLoadSet from the Timer API (driverLib)

\*/

TimerLoadSet(TIMER0\_BASE, TIMER\_A, 0); // Since the interrupt fires at zero, you must subtract 1.

/\* Interrupt Enable

\* You have to enable the interrupt in both:

\* 1.) Timer Module

\* 2.) Nested Vector Interrupt Controller (NVIC)

\*

\* IntMasterEnable() is the master interrupt enable API for all interrupts.

\* IntEnable() enables the specific vector associated with Timer0A.

\* TimerIntEnable() enables a specific event within the timer to generate an interrupt.

\*/

IntEnable(INT\_TIMER0A);

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

/\* Timer Enable

\* This will start the timer and interrupts will begin triggering the timeouts

\*/

TimerEnable(TIMER0\_BASE, TIMER\_A);

}

/\* Timer Interrupt Handler

\* Since this program is interrupt driven, we must add an interrupt handler (ISR) for the timer.

\* We must first clear the interrupt source

\* Then toggle the GPIO pin based on the current state.

\*

\* In case my last program left any LEDs powered on, I called GPIOPinWrite() to turn off all LEDs.

\*/

void Timer0IntHandler(void)

{

int32\_t ui32PeriodHigh = 0.43 \* (SysCtlClockGet() / 10);

int32\_t ui32PeriodLow = 0.57 \* (SysCtlClockGet() / 10);

// 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);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3, 0); // Turn off all LEDs

}

else

{

TimerLoadSet(TIMER0\_BASE, TIMER\_A, ui32PeriodHigh);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4); //Turns on blue LED

}

}

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

**Task 02:**

Youtube Link:

<https://youtu.be/0UQ_8bBNFQE>

**Modified Code:**

#include <stdint.h>

#include <stdbool.h>

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\tm4c123gh6pm.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\hw\_memmap.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\hw\_types.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\sysctl.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\interrupt.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\gpio.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\timer.h"

void configureTimer0();

void configureTimer1A();

void PORTFPin4IntHandler();

int main(void)

{

// Clock Setup

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

// GPIO Configuration

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

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

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

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

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

IntEnable(INT\_GPIOF);

configureTimer0();

while (1)

{

}

}

void configureTimer1A()

{

int32\_t ui32PeriodHigh = (SysCtlClockGet() / 1);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

IntMasterEnable();

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

TimerLoadSet(TIMER1\_BASE, TIMER\_A, 0);

IntEnable(INT\_TIMER1A);

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

TimerEnable(TIMER1\_BASE, TIMER\_A);

}

void configureTimer0()

{

// Timer Configuration

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0);

IntMasterEnable();

TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configures Timer 0 as a 32-bit timer in

TimerLoadSet(TIMER0\_BASE, TIMER\_A, 0); // Since the interrupt fires at zero, you must subtract 1.

IntEnable(INT\_TIMER0A);

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

TimerEnable(TIMER0\_BASE, TIMER\_A);

}

void Timer0IntHandler(void)

{

int32\_t ui32PeriodHigh = 0.43 \* (SysCtlClockGet() / 10);

int32\_t ui32PeriodLow = 0.57 \* (SysCtlClockGet() / 10);

// 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);

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

}

else

{

TimerLoadSet(TIMER0\_BASE, TIMER\_A, ui32PeriodHigh);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4); //Turns on blue LED

}

}

void Timer1IntHandler(void)

{

TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

SysCtlDelay(10000000);

}

void PORTFPin4IntHandler(void) {

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

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //Turns off blue LED

configureTimer1A();

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2); //Turns on blue LED

}

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