**Date Submitted: 10/04/2018**

**Assignment Youtube Playlist:**

[**https://www.youtube.com/playlist?list=PL4oTyvRrubXfa0Y\_DK3FNHwv8qt7QG9KZ**](https://www.youtube.com/playlist?list=PL4oTyvRrubXfa0Y_DK3FNHwv8qt7QG9KZ)

**Task 00: Execute the provided code, no submission is required.**

**Youtube Link:** [**https://youtu.be/kZVQlXmWlzw**](https://youtu.be/kZVQlXmWlzw)

**Original Code (added comments):**

**#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;

//Creating a variable to hold the period.

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

//40MHz total system clock.

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

//Enable the GPIOF peripheral.

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

//Set GPIOF pins 1, 2, 3 as output.

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

//Enable Timer 0 peripheral.

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

//Set Timer 0 in periodic configuration.

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

//Set period to be equal to 1/20 the clock frequency, for a total blink rate of 10Hz, 50% duty cycle.

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

//Load previously-calculated period into timer.

**IntEnable**(INT\_TIMER0A); //Enable vector associated with Timer 0A.

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //Enable interrupt from Timer 0A.

**IntMasterEnable**(); //Enable interrupt controller.

**TimerEnable**(TIMER0\_BASE, TIMER\_A); //Enable and start the timer.

**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);

}

**else**

{

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

}

}

**Modifications to tm4c123gh6pm\_startup\_ccs.c:**

**extern** **void** **Timer0IntHandler**(**void**); //Declare Timer 0 interrupt handler.

Timer0IntHandler, // Timer 0 subtimer A

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

**Task 01: Change the toggle of the GPIO at 2 Hz using Timer0 with 75% duty cycle and verify the waveform generated.**

For task 01 I calculated the periods required for the LED to be on and off for a total period of 2Hz and at a 75% duty cycle. As each separate cycle was triggered, I then loaded the period of the opposite cycle into the timer to begin again, alternating between the 25% and 75% portions of the total period.

Youtube Link: <https://youtu.be/8D1p9sdGpGE>

**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 ui32PeriodOn;

uint32\_t ui32PeriodOff;

//Variables of convenience to store periods for LED-on and LED-off.

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

{

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

//40MHz total clock.

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

//Enable GPIOF peripheral.

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

//Set GPIOF pins 1, 2, 3 as outputs.

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

//Enable Timer 0 peripheral.

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

//Configure timer 0 to be periodic.

uint32\_t ui32Clock = **SysCtlClockGet**(); //Retrieve system clock and store in variable.

ui32PeriodOn = (ui32Clock / 2) \* 75 / 100; //Calculate period of on-cycle.

ui32PeriodOff = (ui32Clock / 2) \* 25 / 100; //Calculate period of off-cycle.

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOn -1); //Load on-period into timer 0.

**IntEnable**(INT\_TIMER0A); //Enable vector associated with Timer 0A.

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //Enable interrupt from Timer 0A.

**IntMasterEnable**(); //Enable interrupt controller.

**TimerEnable**(TIMER0\_BASE, TIMER\_A); //Enable and start the timer.

**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);

//If LED is on, turn it off and begin the off-period timer.

}

**else**

{

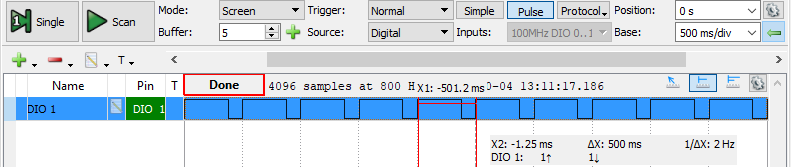
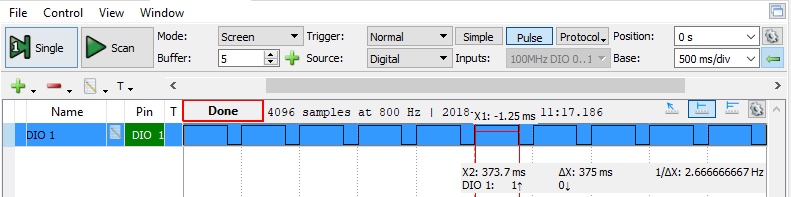
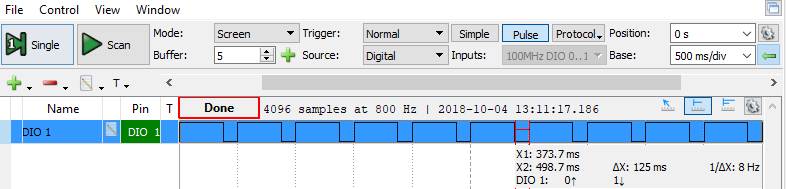
**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4);

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

//If LED is off, turn it on and begin the on-period timer.

}

}



Total off-time 125ms, on-time 375ms, total period 500ms.

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

**Task 02: Include a GPIO Interrupt to Task 02 from switch SW2 to turn ON and the LED for 1.5 sec. Use a Timer1 to calculate the 1.5 sec delay. The toggle of the GPIO is suspended when executing the interrupt.**

For task 02 I added a second timer as well as an interrupt connected to Port F pin 0. To use this pin, I needed to disable the lock which also included adding an additional include of hw\_gpio.h in order to access the GPIO\_0\_LOCK and GPIO\_0\_CR values. The function is the same as task 01, but when the button is pressed, Timer 0 is stopped and the blue LED is turned on. Timer 1 is started and when the interrupt for timer 1 is triggered, the blue LED is turned off and timer 0 is re-loaded at the off-stage and restarted. Additional modification of the startup C file was required, detailed below.

Youtube Link: <https://youtu.be/CUKKmKS3NZE>

**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"

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

uint32\_t ui32PeriodOn;

uint32\_t ui32PeriodOff;

uint32\_t ui32PeriodButton;

//Variables of convenience to store periods for LED-on, LED-off, and Button-pressed LED-on.

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

{

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

//40MHz clock.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); //Enable GPIOF peripheral.

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

//Set GPIOF pins 1, 2, 3 as outputs

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

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

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

//Write to registers to unlock GPIO Port F Pin 0, normally reserved for debugging.

**GPIOPinTypeGPIOInput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0); //Set Port F pin 0 as input.

**GPIOIntEnable**(GPIO\_PORTF\_BASE, GPIO\_INT\_PIN\_0); //Enable Pin F0 as input.

**GPIOIntTypeSet**(GPIO\_PORTF\_BASE, GPIO\_INT\_PIN\_0, GPIO\_RISING\_EDGE); //Set rising edge detected as input for pin F0.

**IntEnable**(INT\_GPIOF); //Enable interrupt for GPIO Port F.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0); //Enable Timer 0 peripheral.

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); //Configure timer 0 to be periodic.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1); //Enable Timer 1 peripheral.

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); //Configure timer 1 to be periodic.

uint32\_t ui32Clock = **SysCtlClockGet**(); //Retrieve system clock and store in variable.

ui32PeriodOn = (ui32Clock / 2) \* 75 / 100; //Calculate period of on-cycle.

ui32PeriodOff = (ui32Clock / 2) \* 25 / 100; //Calculate period of off-cycle.

ui32PeriodButton = (ui32Clock \* 1.5); //Calculate period of LED on when button pressed.

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOn -1); //Load on-period into timer 0.

**IntEnable**(INT\_TIMER0A); //Enable vector associated with Timer 0A.

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, ui32PeriodButton -1); //Load button on-period into timer 1.

**IntEnable**(INT\_TIMER1A); //Enable vector associated with Timer 1A.

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //Enable interrupt from Timer 0A.

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); //Enable interrupt from Timer 1A.

**IntMasterEnable**();

**TimerEnable**(TIMER0\_BASE, TIMER\_A); //Enable and start timer 0A.

**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);

//If LED is on, turn it off and begin the off-period timer.

}

**else**

{

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

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

//If LED is off, turn it on and begin the on-period timer.

}

}

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

{

// Clear the timer interrupt

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

**TimerDisable**(TIMER1\_BASE, TIMER\_A); //Disable Timer 1A.

//When Timer 1A is triggered, turn blue LED off.

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

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32PeriodOff -1); //Load off-cycle period into Timer 0A.

**TimerEnable**(TIMER0\_BASE, TIMER\_A); //Enable Timer 0A.

}

**void** **PortFPin0IntHandler**(**void**)

{

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

**TimerDisable**(TIMER0\_BASE, TIMER\_A); //Disable Timer 0A.

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4); //Turn on Blue LED.

**TimerEnable**(TIMER1\_BASE, TIMER\_A); //Enable Timer 1A.

}

**Modifications to tm4c123gh6pm\_startup\_ccs.c:**

**extern** **void** **PortFPin0IntHandler**(**void**); //Declare PortF interrupt handler.

**extern** **void** **Timer0IntHandler**(**void**); //Declare Timer 0 interrupt handler.

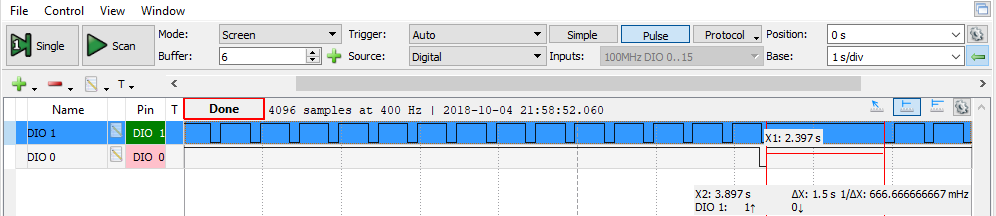
**extern** **void** **Timer1IntHandler**(**void**); //Declare Timer 1 interrupt handler.

Timer0IntHandler, // Timer 0 subtimer A

Timer1IntHandler, // Timer 1 subtimer A

PortFPin0IntHandler, // GPIO Port F

Below is shown the trigger on the rising edge of switch 2 causing a 1.5 second lighting on Port F pin 2 (Blue LED) for 1.5 seconds, before resuming the pattern.

****

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