**Date Submitted: 10/4/2018 8:04 PM**

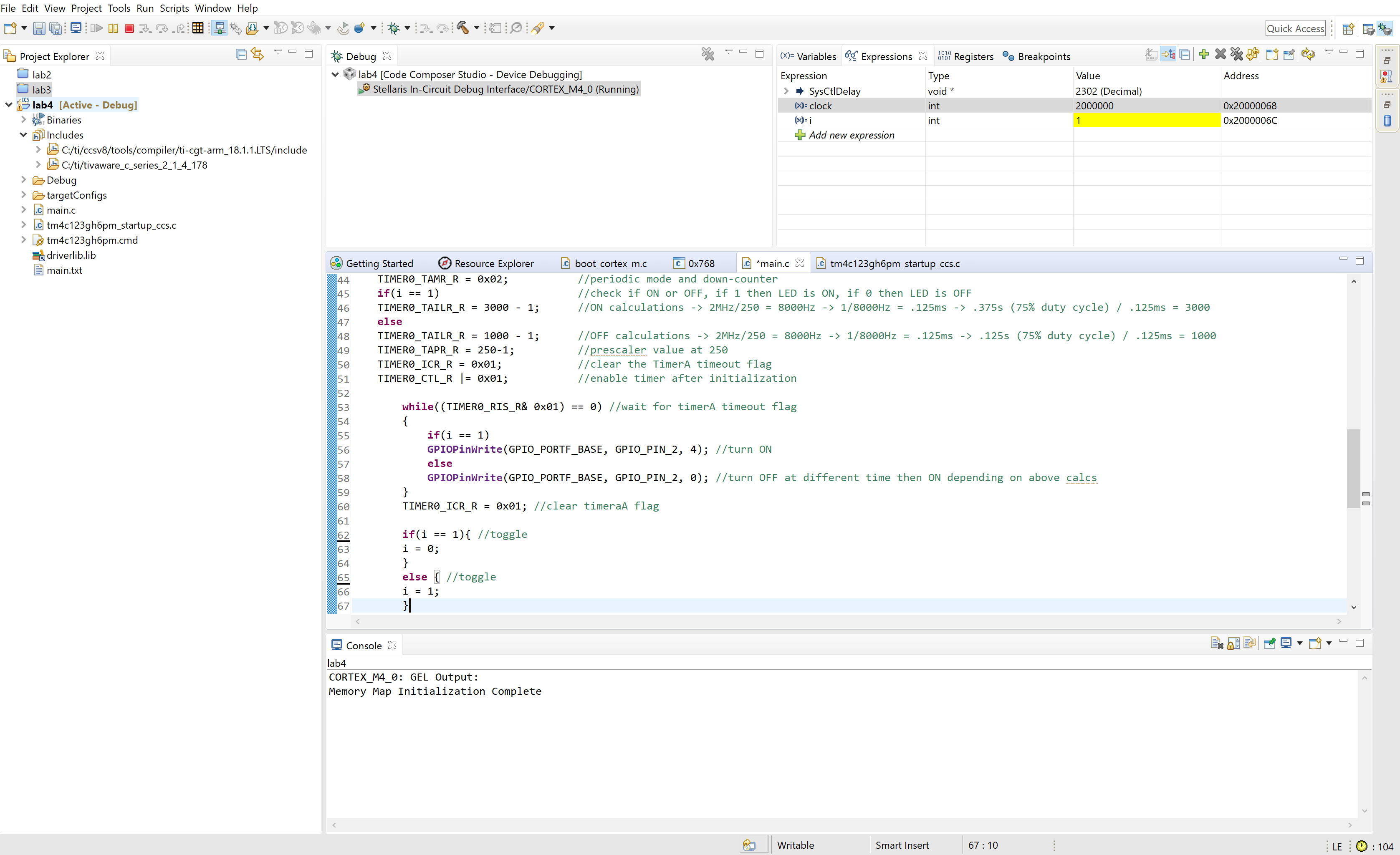
**Task 00: Execute provided code**

**Youtube Link: No Submission required for Task 00**

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

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

Youtube Link: <https://www.youtube.com/watch?v=q1mD8run9Qo>



Above is a screenshot when debugging has begun. To verify the clock frequency the function SysCtlClockGet() was used for the int variable “clock”. On the top right we can see that the clock variable has a value of 2 x 10^6 which would approximately be 2MHz. I used the calculations within the code to figure out how long 75% duty cycle would be at .375s (ON) and 0.125s (OFF).

**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"

**int** clock; //check clock freq

**int** i; //global variable

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

{

uint32\_t ui32Period;

**SysCtlClockSet**(SYSCTL\_SYSDIV\_2|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_4MHZ|SYSCTL\_OSC\_MAIN); //4MHz/2 = 2MHz

**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);

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

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

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

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

**while**(1)

{

}

}

**void** **Timer0IntHandler**(**void**)

{

**while**(1)

{

clock = **SysCtlClockGet**(); //double check clock freq at 2MHz

**TimerIntClear**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //clear the timer interrupt

SYSCTL\_RCGCTIMER\_R |= 0x01; //Activate Timer0

TIMER0\_CTL\_R = 0; //disable timer before initialization

TIMER0\_CFG\_R = 0x04; //16 - bit option

TIMER0\_TAMR\_R = 0x02; //periodic mode and down-counter

**if**(i == 1) //check if ON or OFF, if 1 then LED is ON, if 0 then LED is OFF

TIMER0\_TAILR\_R = 3000 - 1; //ON calculations -> 2MHz/250 = 8000Hz -> 1/8000Hz = .125ms -> .375s (75% duty cycle) / .125ms = 3000

**else**

TIMER0\_TAILR\_R = 1000 - 1; //OFF calculations -> 2MHz/250 = 8000Hz -> 1/8000Hz = .125ms -> .125s (75% duty cycle) / .125ms = 1000

TIMER0\_TAPR\_R = 250-1; //prescaler value at 250

TIMER0\_ICR\_R = 0x01; //clear the TimerA timeout flag

TIMER0\_CTL\_R |= 0x01; //enable timer after initialization

**while**((TIMER0\_RIS\_R& 0x01) == 0) //wait for timerA timeout flag

{

**if**(i == 1)

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

**else**

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //turn OFF at different time then ON depending on above calcs

}

TIMER0\_ICR\_R = 0x01; //clear timeraA flag

**if**(i == 1){ //toggle

i = 0;

}

**else** { //toggle

i = 1;

}

}

}

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

**Task 02:** 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.

Youtube Link: <https://www.youtube.com/watch?v=8Oq5vfw7LnA>

The below code is to implement the above Task 01 by toggling an LED in the ON state at 0.375s to the OFF state at 0.125s. While Timer0 is handling the LED toggling, there is another interrupt for SW2. When SW2 is pressed, the toggling will stop and another LED will stay ON for 1.5s using Timer1 and then after, the toggling will continue with Timer0.

**Modified Code:**

**#define** PART\_TM4C123GH6PM

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "stdlib.h"

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

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

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

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

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

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

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

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

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

**#include** "driverlib/pin\_map.h"

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

**#include** "driverlib/rom\_map.h"

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

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

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

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

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

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

**#include** <string.h>

**#include** "inc/tm4c123gh6pm.h"

**int** clock; //check clock freq

**int** i; //global variable

**#define** LED\_PERIPH SYSCTL\_PERIPH\_GPIOF

**#define** LED\_BASE GPIO\_PORTF\_BASE

**#define** RED\_LED GPIO\_PIN\_1

**#define** BLUE\_LED GPIO\_PIN\_2

**#define** GREEN\_LED GPIO\_PIN\_3

**#define** Button\_PERIPH SYSCTL\_PERIPH\_GPIOF

**#define** ButtonBase GPIO\_PORTF\_BASE

**#define** Button GPIO\_PIN\_0

**#define** ButtonInt GPIO\_INT\_PIN\_0

**volatile** uint8\_t value=0;

**void** **Timer0IntHandler**(**void**)

{

**while**(1)

{

**float** delayTime = 1.5;

uint32\_t status=0;

uint8\_t state=0;

status = **GPIOIntStatus**(ButtonBase,true);

**GPIOIntClear**(ButtonBase,status);

**if**(status & ButtonInt == ButtonInt){ //this means the button interrupt has been pressed

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //turn off LED

TIMER0\_CTL\_R = 0; //disable timer

**TimerIntClear**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //clear the timer interrupt

timer1A\_delaySec(); //run 1.5s delay function

}

clock = **SysCtlClockGet**(); //double check clock freq at 2MHz

**TimerIntClear**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //clear the timer interrupt

SYSCTL\_RCGCTIMER\_R |= 0x01; //Activate Timer0

TIMER0\_CTL\_R = 0; //disable timer before initialization

TIMER0\_CFG\_R = 0x04; //16 - bit option

TIMER0\_TAMR\_R = 0x02; //periodic mode and down-counter

**if**(i == 1) //check if ON or OFF, if 1 then LED is ON, if 0 then LED is OFF

TIMER0\_TAILR\_R = 3000 - 1; //ON calculations -> 2MHz/250 = 8000Hz -> 1/8000Hz = .125ms -> .375s (75% duty cycle) / .125ms = 3000

**else**

TIMER0\_TAILR\_R = 1000 - 1; //OFF calculations -> 2MHz/250 = 8000Hz -> 1/8000Hz = .125ms -> .125s (75% duty cycle) / .125ms = 1000

TIMER0\_TAPR\_R = 250-1; //prescaler value at 250

TIMER0\_ICR\_R = 0x01; //clear the TimerA timeout flag

TIMER0\_CTL\_R |= 0x01; //enable timer after initialization

**while**((TIMER0\_RIS\_R& 0x01) == 0) //wait for timerA timeout flag

{

**if**(i == 1)

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

**else**

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //turn OFF at different time then ON depending on above calcs

}

TIMER0\_ICR\_R = 0x01; //clear timeraA flag

**if**(i == 1){ //toggle

i = 0;

}

**else** { //toggle

i = 1;

}

}

}

**void** **timer1A\_delaySec**(**void**)

{

clock = **SysCtlClockGet**();

SYSCTL\_RCGCTIMER\_R |= 2; //enable clock

TIMER1\_CTL\_R = 0; //disable Timer before initialization

TIMER1\_CFG\_R = 0x04; //16-bit option

TIMER1\_TAMR\_R = 0x02; //periodic mode and down-counter

TIMER1\_TAILR\_R = 12000-1; //TimerA interval load value reg

TIMER1\_TAPR\_R = 250 - 1; //TimerA prescaler 2MHz/250 = 8000Hz => 1/8000 = .125ms => 1.5s/.125ms = 12000

TIMER1\_ICR\_R = 0x01; //clear the TimerA timeout flag

TIMER1\_CTL\_R |= 0x01; //enable Timer A after initialization

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

**while**((TIMER1\_RIS\_R & 0x1) == 0)

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2) ; //delay and wait

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0); //turn off red LED

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

}

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

{

uint32\_t ui32Period;

//Set the clock to 2Mhz

**SysCtlClockSet**(SYSCTL\_SYSDIV\_2|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_4MHZ);

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

**SysCtlPeripheralEnable**(LED\_PERIPH);

**SysCtlDelay**(3);

HWREG(GPIO\_PORTF\_BASE+GPIO\_O\_LOCK) = GPIO\_LOCK\_KEY; //unlock the SW2 Pin

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

//Button Interrupt

**GPIOPinTypeGPIOInput**(ButtonBase, Button);

**GPIOPadConfigSet**(ButtonBase ,Button,GPIO\_STRENGTH\_2MA,GPIO\_PIN\_TYPE\_STD\_WPU);

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

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

//Timer Interrupt

**GPIOPinTypeGPIOOutput**(LED\_BASE, RED\_LED| BLUE\_LED | GREEN\_LED);

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

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

ui32Period = **SysCtlClockGet**() ; //set to 2MHz

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

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

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

**while**(1){

}

}