**Date Submitted: 09/28/19**

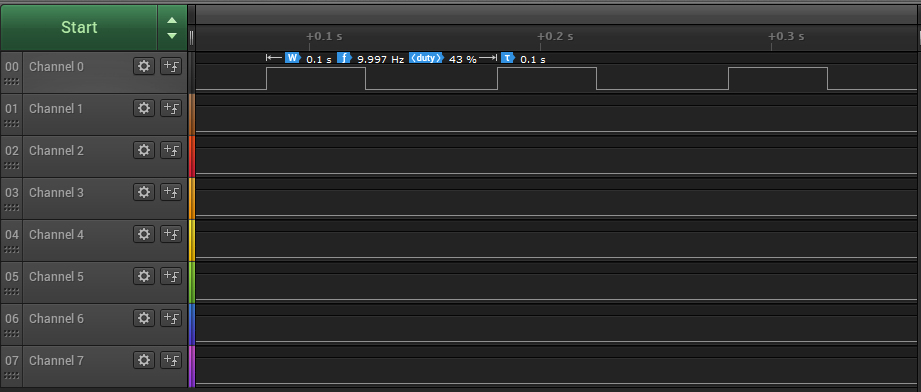
**Task 00: Execute provided code**

**Youtube Link:**

<https://youtu.be/f39u6m32AMY>

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

**Task 01:**

****

Youtube Link:

<https://youtu.be/f39u6m32AMY>

**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"

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

{

uint32\_t ui32Period;

**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);

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

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

ui32Period = (**SysCtlClockGet**() / 10) \* 0.43; //toggle GPIO at 10Hz with 43% duty cycle

**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**)

{

uint32\_t ui32Period;

// 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);

ui32Period = (**SysCtlClockGet**() / 10) \* 0.57;

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

}

**else**

{

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

ui32Period = (**SysCtlClockGet**() / 10) \* 0.43;

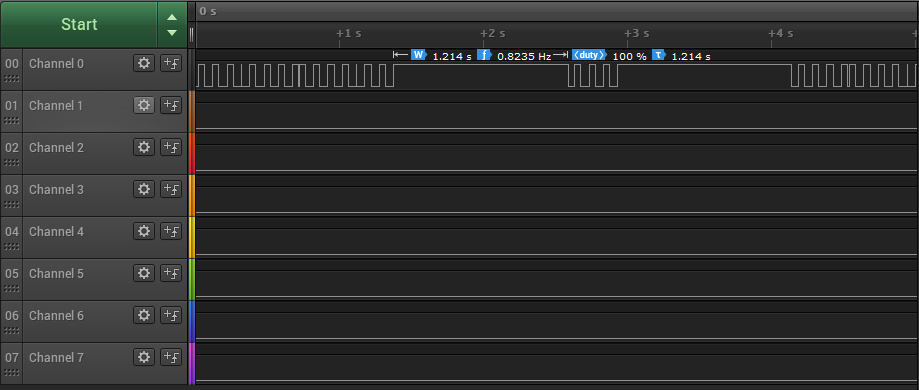
**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32Period -1);

}

}

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

**Task 02:**

****

Youtube Link:

<https://youtu.be/f39u6m32AMY>

**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"

**void** **timer1A\_delaySec**(**int** ttime)

{

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

**int** i;

SYSCTL\_RCGCTIMER\_R |= 2; //enable clock to timer block 1

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 = 64000 - 1; //TimerA interval load value reg

TIMER1\_TAPR\_R = 250 - 1; //TimerA Prescaler 16MHz/250=64000Hz

TIMER1\_ICR\_R = 0x1; //clear the TimerA timeout flag

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

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

**while** ((TIMER1\_RIS\_R & 0x1) == 0); //wait for TimerA timeout flag

TIMER1\_ICR\_R = 0x1; //clear the timer A timeout flag

}

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

}

**void** **configureTimer1A**()

{

uint32\_t ui32Period;

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

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); //configure timer operation as periodic

//Configure timer frequency

//Frequency is given by MasterClock / CustomValue

//ui32Period = (SysCtlClockGet() / 1) \* 0.5; //toggle GPIO at 1Hz with 50% duty cycle

//TimerLoadSet(TIMER1\_BASE, TIMER\_A, ui32Period -1);

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, 120000000);

**IntEnable**(INT\_TIMER1A); //Enable timer 1a interrupt

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); //timer 1a interrupt when timeout

**IntMasterEnable**(); //Enable Interrupts

**TimerEnable**(TIMER1\_BASE, TIMER\_A); //Start Timer 1a

}

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

{

uint32\_t ui32Period;

**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);

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

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

ui32Period = (**SysCtlClockGet**() / 10) \* 0.43; //toggle GPIO at 10Hz with 43% duty cycle

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

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

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

//Switch interrupt

**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);

//configure timer1a

configureTimer1A();

**while**(1)

{

}

}

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

{

uint32\_t ui32Period;

// 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);

ui32Period = (**SysCtlClockGet**() / 10) \* 0.57;

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

}

**else**

{

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

ui32Period = (**SysCtlClockGet**() / 10) \* 0.43;

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

}

}

**void** **PortFPin4IntHandler**(**void**){

//Clear the GPIO interrupt

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

//Read the current state of the GPIO pin and

//write back the opposite state

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

//Call TIMER 1 Delay

timer1A\_delaySec(3);

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

}

**void** **Timer1AIntHandler**(**void**){

//Required to launch next interrupt

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

//TimerIntClear(TIMER1\_BASE, TIMER\_A);

// 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);

}

}

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