**Task 0:**

Execute the supplied code, no submission required.

**#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;

// ==================================================

// Configure system clock

// CLK = 40MHz = (400MHz PLL / (5 \* 2))

// --------------------------------------------------

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

// enable periphery for GPIO\_F and set pins 1 through 3 as output

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

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

// enable TIMER\_0 and set it to periodic configuration

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

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

/\*

\* compute period

\* PERIOD: (CLK / Hz) / (DUTY FRACTION.)

\*

\* (40MHz / 10Hz) / 2)

\*/

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

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

// interrupt for TIMER0

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

// enable TIMER0

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

**while**(1)

{

}

}

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

{

// Clear the timer interrupt

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

// 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 1:**

Change the toggle of the GPIO at 2 Hz using Timer0 with 75% duty cycle and verify the

waveform generated.

**#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**)

{

**double** DUTY\_75 = 1.33333;

uint32\_t ui32Period;

// ==================================================

// Configure system clock

// CLK = 40MHz = (400MHz PLL / (5 \* 2))

// --------------------------------------------------

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

// enable periphery for GPIO\_F and set pins 1 through 3 as output

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

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

// enable TIMER\_0 and set it to periodic configuration

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

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

/\*

\* compute period

\* PERIOD: (CLK / Hz) / (DUTY FRACTION.)

\*

\* recall 50% => 2, 20% duty => 5

\* 75% = 3/4 of CLK so 15MHz at 2Hz

\*

\* (40MHz / 2Hz) / 1.3333) = 15000000

\*/

ui32Period = (**SysCtlClockGet**() / 2) / DUTY\_75;

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

// interrupt for TIMER0

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

// enable TIMER0

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

**while**(1)

{

}

}

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

{

// Clear the timer interrupt

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

// 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 2:**

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.

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

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

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

// stores LED pin data

uint8\_t ui8PinData;

// LED array for asserting specific color configurations

// R, G, B, RG, RB, GB, RGB

uint8\_t array[7]={2,8,4,10,6,12,14};

// index counter

uint8\_t i = 0;

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

{

// ==================================================

// Configure system clock

// CLK = 40MHz = (400MHz PLL / (5 \* 2))

// --------------------------------------------------

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

// enable periphery for GPIO\_F and set pins 1 through 3 as output

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

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

**while**(1)

{

// set LED's on depending on pin data

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3, ui8PinData);

/\*

\* DELAY: ((seconds) / ( (1/CLK) \* 3)

\*

\* For 0.425 seconds

\* ((0.425) / ( (1/CLK) \* 3) = 56.7x10^5

\*

\* So close approximation gives

\* 40MHz / 7 = 57.1x10^5

\*/

**SysCtlDelay**(**SysCtlClockGet**() / 7);

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0x00);

// delay to turn off LEDs

**SysCtlDelay**(**SysCtlClockGet**() / 7);

// assign pin values through predefined array for LED colors

ui8PinData = array[i];

// iterate through LED configuration array

i++;

// at the 7th index location, reset LED counter

**if**(i == 7)

i=0;

}

}