**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)

\*

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

\* 75% = 3/4 of CLK is 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/tm4c123gh6pm.h"

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

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

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

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

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

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

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

// duty cycle divisor for 75% duty

**double** DUTY\_75 = 1.3333;

uint32\_t ui32Period;

**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);

// disable GPIO lock for SW2

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

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

// enable TIMER0 and set it to periodic configuration

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

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

// enable TIMER1

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

//enable the GPIO peripheral and configure the pins connected to the switch as inputs.

**GPIOPinTypeGPIOInput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0);

//enables a specific event within the GPIO to generate an interrupt.

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

//sets interrupt to rising edge on GPIO

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

//enables the specific vector associated with GPIOF.

**IntEnable**(INT\_GPIOF);

/\*

\* 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\_BOTH);

// interrupt for TIMER1

**IntEnable**(INT\_TIMER1A);

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

// allows interrupts to operate

**IntMasterEnable**();

// enable TIMER0

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

// run program indefinitely

**while**(1)

{

}

}

// timer0, default LED blinking protocol

**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);

}

}

// portF, SW2 button protocol

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

{

// disable GPIOF interrupt

**IntDisable**(INT\_GPIOF);

// clear interrupt flag

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

// force set LED to 0 as a reset

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

// turn on LED again

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

// disable TIMER0

**IntDisable**(INT\_TIMER0A);

// assign TIMER1 to a 1.5s long value

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, **SysCtlClockGet**() \* 1.5);

// enable the timer

**TimerEnable**(TIMER1\_BASE, TIMER\_A);

// enable TIMER1 interrupt

**IntEnable**(INT\_TIMER1A);

// program will enter Timer1IntHandler function after 1.5s

}

// timer1, LED stays on until it enters this interrupt

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

{

// stop TIMER1

**TimerDisable**(TIMER1\_BASE, TIMER\_A);

// Clear the timer interrupt

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

// force set the LED to off

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

// reenable GPIO and TIMER0 interrupts

**IntEnable**(INT\_GPIOF);

**IntEnable**(INT\_TIMER0A);

// turn off TIMER1 until button is pressed again

**IntDisable**(INT\_TIMER1A);

}