**Task 00: Execute supplied code**

//

/\* Melissa Cordova

\* CPE 403 - LAB 4

\*

\* main.c

\*/

**#include** <stdint.h> //variable definitions for the C99 standard

**#include** <stdbool.h> //boolean definitions for the C99 standard

**#include** "inc/tm4c123gh6pm.h" //definitions for the interrupt and register assig

**#include** "inc/hw\_memmap.h" //macros defining the memory map of the TivaC

**#include** "inc/hw\_types.h" //defines common types and macros

**#include** "driverlib/sysctl.h" //defines and macros for System Control API

**#include** "driverlib/interrupt.h" //defines and macros for NVIC Controller

**#include** "driverlib/gpio.h" //Defines and macros for GPIO API of DriverLib

**#include** "driverlib/timer.h" //defines and macros for Timer API of DriverLib

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

{

uint32\_t ui32Period; //variable ui32Period with unsigned 32-bit int

//system clock to run at 40MHz

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

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); //enable GPIO peripheral

//configure pins

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

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

//configures Timer 0 as a 32-bit timer in periodic mode

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

//toggle GPIO at 10Hz and a 50% duty cycle and interrupt at 1/2 period

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

//load into Timer's Interval Load register

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

**IntEnable**(INT\_TIMER0A); //enable specific vector associated with Timer0A

//master interrupt enable API for all interrupts

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

**IntMasterEnable**();

**TimerEnable**(TIMER0\_BASE, TIMER\_A); //enable 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 opposites 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 01: Change the toggle of the GPIO at 50Hz and at 75% duty cycle and verify**

**#include** <stdint.h> //variable definitions for the C99 standard

**#include** <stdbool.h> //boolean definitions for the C99 standard

**#include** "inc/tm4c123gh6pm.h" //definitions for the interrupt and register assig

**#include** "inc/hw\_memmap.h" //macros defining the memory map of the TivaC

**#include** "inc/hw\_types.h" //defines common types and macros

**#include** "driverlib/sysctl.h" //defines and macros for System Control API

**#include** "driverlib/interrupt.h" //defines and macros for NVIC Controller

**#include** "driverlib/gpio.h" //Defines and macros for GPIO API of DriverLib

**#include** "driverlib/timer.h" //defines and macros for Timer API of DriverLib

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

{

uint32\_t ui32Period; //variable ui32Period with unsigned 32-bit int

//system clock to run at 40MHz

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

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); //enable GPIO peripheral

//configure pins

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

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

//configures Timer 0 as a 32-bit timer in periodic mode

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

//toggle GPIO at 50Hz and at 75% duty cycle

ui32Period = (**SysCtlClockGet**()/50) /2;

//load into Timer's Interval Load register

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

**IntEnable**(INT\_TIMER0A); //enable specific vector associated with Timer0A

//master interrupt enable API for all interrupts

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

**IntMasterEnable**();

**TimerEnable**(TIMER0\_BASE, TIMER\_A); //enable 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 opposites 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);

//75% duty cycle

**SysCtlDelay**(2000000);

}

}

**Task 02: Include a GPIO Interrupt to task 02 from switch SW2 to turn ON and the LED for 2 sec. Use a Timer1 to calculate the 2 sec delay. The toggle of the GPIO is suspended when executing the interrupt.**

**#include** <stdint.h> //variable definitions for the C99 standard

**#include** <stdbool.h> //boolean definitions for the C99 standard

**#include** "inc/tm4c123gh6pm.h" //definitions for the interrupt and register assig

**#include** "inc/hw\_memmap.h" //macros defining the memory map of the TivaC

**#include** "inc/hw\_types.h" //defines common types and macros

**#include** "driverlib/sysctl.h" //defines and macros for System Control API

**#include** "driverlib/interrupt.h" //defines and macros for NVIC Controller

**#include** "driverlib/gpio.h" //Defines and macros for GPIO API of DriverLib

**#include** "driverlib/timer.h" //defines and macros for Timer API of DriverLib

**void** **IntGPIOF0**(**void**);

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

{

uint32\_t ui32Period; //variable ui32Period with unsigned 32-bit int

//system clock to run at 40MHz

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

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); //enable GPIO peripheral

//configure pins

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

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

//configures Timer 0 as a 32-bit timer in periodic mode

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

//toggle GPIO at 50Hz and at 75% duty cycle

ui32Period = (**SysCtlClockGet**()/50) /2;

//load into Timer's Interval Load register

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

//Unlock PINF0 to use interrupt for SW2

SYSCTL\_RCGC2\_R |= 0x00000020; //activate clock

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // GPIO Port F

GPIO\_PORTF\_CR\_R = 0x1F;

GPIO\_PORTF\_AMSEL\_R = 0x00; // disable analog

GPIO\_PORTF\_PCTL\_R = 0x00000000;

GPIO\_PORTF\_DIR\_R = 0x0E; // set PF0 in, PF4, and PF3-1 out

GPIO\_PORTF\_AFSEL\_R = 0x00; // disable PF7-0

GPIO\_PORTF\_PUR\_R = 0x11; // enable PF0 and PF4

GPIO\_PORTF\_DEN\_R = 0x1F; // enable digital I/O

//register the interrupt handler for PF0

**GPIOIntRegister**(GPIO\_PORTF\_BASE, IntGPIOF0);

//SW2 goes low when pressed

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

//enable interrupts on PF0

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

**IntEnable**(INT\_TIMER0A); //enable specific vector associated with Timer0A

//master interrupt enable API for all interrupts

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

**IntMasterEnable**();

**TimerEnable**(TIMER0\_BASE, TIMER\_A); //enable 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 opposites 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);

**SysCtlDelay**(2000000);

}

}

**void** **IntGPIOF0**(**void**)

{

uint32\_t delay;

//clear interrupt flag on pin F0

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

//Turn on Blue LED

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);

//Enable TIMER1 peripheral

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

//Set TIMER1 to periodic mode

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

delay = (**SysCtlClockGet**()/2);

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, (delay-1));

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

**while** (**TimerValueGet**(TIMER1\_BASE, TIMER\_A) < (delay-2));

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

}