Abenezer Namaga

CPE 403 – 1001

TIVAC LAB4

|  |
| --- |
|  |

Github root directory: https://github.com/Ayertena/AdvEmbededSys

Date Submitted: 10/02/2018

**Task 00:** Execute the provided code, no submission is required.

------------------------------------------------------------------------------------------------------------------------------

**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://youtu.be/JDHiQs3F4bw>

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

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

**#include** "inc/hw\_memmap.h" // Macros defining the memory map of the Tiva C Series device.

// This includes defines such as peripheral base address locations

// such as GPIO\_PORTF\_BASE.

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

**#include** "driverlib/sysctl.h" // Defines and macros for System Control API of DriverLib.

// This includes API functions such as SysCtlClockSet and SysCtlClockGet.

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

// This includes API functions such as GPIOPinTypePWM and GPIOPinWrite.

**#include** "driverlib/interrupt.h"// Def macros for interrupt controller

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

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

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

{

uint32\_t ui32Period;

// system clock runs at 40MHz ()

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

// Enable the GPIO peripheral and configure the pins connected to the LEDs as outputs.

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

// Enable clock to the peripheral

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0);

// Configure Timer0 as 32 bit timer in periodic mode.

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

/\* Calculate Delay GPIO at 2Hz 75% duty cycle

generate an interrupt at

3/4 of the desired period.

\* First calculate the # of clock cycles required for 2Hz period by calling SysCtclockGet() and dividing desired frequency.

\* Then divide the result by (0.75 = 3/4).

\* Finally load the value you get into timers interval load register\*/

ui32Period = (SysCtlClockGet() / 2) \* 0.75;

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

IntEnable(INT\_TIMER0A); // Enable the interrupt in the timer module

TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); // Enables a specific event within the timer to generate an interrupt.

IntMasterEnable(); // Enables the specific vector associated with Timer0A.

TimerEnable(TIMER0\_BASE, TIMER\_A); // Enable 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 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 02:** Include a GPIO Interrupt to Task 02 from switch SW2 to turn ON and the LED for 1.5sec. Use a Timer1 to calculate the 1.5 sec delay. The toggle of the GPIO is suspended when executing the interrupt.

**Youtube Link:** <https://youtu.be/HVso_4skjW8>

**#include** <stdint.h> // Var definitions for the C99 std

**#include** <stdbool.h> // Boolean defn for the C99 std

**#include** "inc/tm4c123gh6pm.h" // device specific header file

**#include** "inc/hw\_memmap.h" //macros def of mem map for TivaC

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

**#include** "driverlib/sysctl.h" //defines macros for sys control APIs

**#include** "driverlib/interrupt.h"//defines macros for interrupt controller

**#include** "driverlib/gpio.h" //defines macros for GPIO APIs

**#include** "driverlib/timer.h" //Defines macros for Timer APIs

uint32\_t ui32Period;

uint32\_t ui32Period1;

**void** **GPIOF0IntHandler**(**void**);

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

{

// configure 40MHz clock

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

// Enable GPIO peripheral and configure pins connected to the LEDs as outputs

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

/\*Enable the clock to the peripheral

\*configure timer 0 as 32 bit timer in periodic mode

\*configure \*Timer0A:Timer0B or Timer0B:Timer0A\*/

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0);

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

/\* Calculate Delay GPIO at 2Hz 75% duty cycle

\* generate an interrupt at 3/4 of the desired period.

\* First calculate the # of clock cycles required for 2Hz period by calling SysCtclockGet()

\* and dividing it by the desired frequency.

\* Then divide the result by (0.75 = 3/4).

\* Finally load the value you get into timers interval load register

\*/

ui32Period = (SysCtlClockGet() / 2) \* 0.75; // ui32Period = 15,000,000

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

// Unlock Pin F0 to use an interrupt on SW2

SYSCTL\_RCGC2\_R |= 0x00000020; // 1) Activate clock for Port F

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // 2) Unlock GPIO Port F

GPIO\_PORTF\_CR\_R = 0x1F; // Allow changes to PF4-0, only PF0 needs to be unlocked, other bits can't be locked

GPIO\_PORTF\_AMSEL\_R = 0x00; // 3) Disable analog on PF

GPIO\_PORTF\_PCTL\_R = 0x00000000; // 4) PCTL GPIO on PF4-0

GPIO\_PORTF\_DIR\_R = 0x0E; // 5) PF4,PF0 in, PF3-1 out

GPIO\_PORTF\_AFSEL\_R = 0x00; // 6) Disable alt funct on PF7-0

GPIO\_PORTF\_PUR\_R = 0x11; // Enable pull-up on PF0 and PF4

GPIO\_PORTF\_DEN\_R = 0x1F; // 7) enable digital I/O on PF4-0

GPIOIntRegister(GPIO\_PORTF\_BASE, GPIOF0IntHandler); // Register the interrupt handler for PF0

GPIOIntTypeSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_FALLING\_EDGE); //SW2 goes low when pressed

GPIOIntEnable(GPIO\_PORTF\_BASE, GPIO\_PIN\_0); // Enable interrupts on PF0

IntEnable(INT\_TIMER0A); // Enable the interrupt in the timer module

TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); // Enables timer event to generate an interrupt

IntMasterEnable(); // Master interrupt enable API for all interrupts

// Enable timer\*/

TimerEnable(TIMER0\_BASE, TIMER\_A);

**while**(1)

{

}

}

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

{

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

// 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);

// Load timer so light is off for 25%

TimerLoadSet(TIMER0\_BASE, TIMER\_A, (ui32Period-1)\*0.25);

}

**else**

{

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

// Load timer so light is on for 75%

TimerLoadSet(TIMER0\_BASE, TIMER\_A, (ui32Period-1)\*0.75);

}

}

**void** **GPIOF0IntHandler**(**void**) // Interrupt handler for GPIO pin F0

{

uint32\_t delay;

GPIOIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_0); // Clear interrupt flag on pin F0

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4 ); // Turn on blue led for 1.5s

SysCtlDelay(20000000);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1); // Enable Timer1

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

ui32Period1 = (SysCtlClockGet()/10); // Delay 1.5s

delay = ui32Period1;

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

TimerEnable(TIMER1\_BASE, TIMER\_A);

**while** (TimerValueGet(TIMER1\_BASE, TIMER\_A) < (delay-10))

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

SysCtlDelay(2000000);

}

------------------------------------------------------------------------------------