Enrique Saldana

https://githb.com/enri10

Date Submitted: 10/23/18

Task00: Execute the supplied code, no submission required.

LAB04 Task00: https://youtu.be/F9o_R3H3WYw

Task 01: Change the toggle of the GPIO at 2 Hz using Timer0 with 75% duty cycle and verify the waveform generated.
2 Hz Toggle => Total delay = 0.5 Sec, 0.375 sec ON, and 0.125 sec OFF
LAB04 Task 01: https://youtu.be/buvrCZ-ooZs

2 Hz Toggle gives us a period of 0.5 sec and a 50% duty cycle meaning 0.25 sec ON and 0.25 sec OFF.
In order to get the LED to maintain ON for a total of 0.375 sec we must make a delay of 0.125 sec.
Current Period of clock = 1/40 MHz= 25 ns
Delay = $(25*10^{-9})x(5*10^{6})$ = 0.125s

```
#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)
{
    //unsigned 32-bit variable
    uint32_t ui32Period;

    //Configure system clock to run at 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    //Enable GPIO peripheral
    //Configure LED's as outputs
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //Enable timer0 peripheral
    //Configure Timer 0 as a 32-bit timer in periodic mode
```

```
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    //Calculate number of clock cycles required for 2Hz
    //Load period into the Timer's Interval Load register
    ui32Period = (SysCtlClockGet() / 2) / 2;
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    //Enable specific vector associated with TIMER0A
    //Enable interrupt to be generated on timeout of TIMER0A
    //Master interrupt enable API for all interrupts
    IntEnable(INT_TIMER0A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    //Start the timer
    TimerEnable(TIMER0_BASE, TIMER_A);

    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);
        //LED stays on for an extra 0.125s totaling 0.375s
        SysCtlDelay(5000000);
    }
}
```

Task 02: 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.

Lab04 Task02: https://youtu.be/Z9-tyoko_1c

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
```

```c
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "inc/hw_gpio.h"   // library to unlock SW2 as an input

int main(void)
{
    //unsigned 32-bit variable
    uint32_t ui32Period;

    //Configure system clock to run at 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    //Enable GPIO peripheral
    //Configure LED's as outputs
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //Unlock SW2 to be used as an input
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;

    //Enable Timer0 peripheral
    //Configure Timer 0 as a 32-bit timer in periodic mode
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    //Enable Timer1 peripheral
    //Configure Timer 1 as a 32-bit timer in periodic mode
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    //Calculate number of clock cycles required for 2Hz
    //Load period into the Timer's Interval Load register
    ui32Period = (SysCtlClockGet() / 2) / 2;
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    //Load period into the Timer's Interval Load register
    TimerLoadSet(TIMER1_BASE, TIMER_A, SysCtlClockGet() -1);

    //Enable specific vector associated with TIMER0A
    //Enable interrupt to be generated on timeout of TIMER0A
    //Master interrupt enable API for all interrupts
    IntEnable(INT_TIMER0A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    //Start the timer0
    TimerEnable(TIMER0_BASE, TIMER_A);

    //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);


    while(1)
    {

    }
}
void Timer0IntHandler(void)
{
    //Master interrupt disable API for all interrupts
    IntMasterDisable();

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

void PortFPin0IntHandler(void)
{
    //Master interrupt disable API for all interrupts
    IntMasterDisable();

    // Clear the GPIO interrupt
    GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_0);

    //Start Timer1
    TimerEnable(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);
```

```
        //Delay of approximately 1.5s
        SysCtlDelay(60000000/3);
    }

    //Stop Timer1
    TimerDisable(TIMER1_BASE, TIMER_A);
    //Master interrupt enable API for all interrupts
    IntMasterEnable();
}
```