Joseph Sharp Halpin
CpE 403 Section 1001
10/4/2018

Task01:
Youtube Link: https://youtu.be/jioJl_o8xXI

Code:

tm4c123gh6pm_startup_ccs.c | *main.c | main() at main.c:11 0x5a4

```c
#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;    //value to hold the timer

    //set the clock frequency to 16MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    //enable the GPIO
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    //configure the GPIO pins
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //enable timer0 to run
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    //configure timer0 for timing
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    //set the period value of 2Hz with 75% duty cycle into ui32Period
    ui32Period = (SysCtlClockGet() / 2) / 1.25;
    //load the period value into timer0
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    //enable interrupt
    IntEnable(INT_TIMER0A);
    //enable timer0 interrupt
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    //enable all API interrupts
    IntMasterEnable();

    //enable timer
    TimerEnable(TIMER0_BASE, TIMER_A);

    //loop forever
    while(1)
    {
    }
}
```

```
48 void Timer0IntHandler(void)
49 {
50     // Clear the timer interrupt
51     TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
52
53     // Read the current state of the GPIO pin and
54     // write back the opposite state
55     if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
56     {
57         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
58     }
59     else
60     {
61         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
62     }
63 }
```

Task02:
Youtube Link:

Code:

```
1  #include <stdint.h>
2  #include <stdbool.h>
3  #include "inc/tm4c123gh6pm.h"
4  #include "inc/hw_memmap.h"
5  #include "inc/hw_types.h"
6  #include "driverlib/sysctl.h"
7  #include "driverlib/interrupt.h"
8  #include "driverlib/gpio.h"
9  #include "driverlib/timer.h"
10 #include "inc/hw_gpio.h"
11 int main(void)
12 {
13     //set the clock frequency to 16MHz
14     SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
15
16     //enable GPIO F
17     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
18     //enable GPIO E
19     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
20
21     HWREG(GPIO_PORTE_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
22     HWREG(GPIO_PORTE_BASE + GPIO_O_CR) |= GPIO_PIN_0;
23
24     //set the output pins for the LED
25     GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
26     //enables the pins connected to the switch as inputs
27     GPIOPinTypeGPIOInput(GPIO_PORTE_BASE, GPIO_PIN_0);
28     //enables a specific event within the GPIO to generate an interrupt
29     GPIOIntEnable(GPIO_PORTE_BASE, GPIO_INT_PIN_0);/////
30     //sets interrupt to rising edge on GPIO
31     GPIOIntTypeSet(GPIO_PORTE_BASE, GPIO_INT_PIN_0, GPIO_RISING_EDGE);////
32
33     IntEnable(INT_GPIOE);////;
34     //IntMasterEnable();
35
36     //loop forever
37     while(1)
38     {
39     }
40 }
41
42 void PortEPin0IntHandler(void)
43 {
44     // Clear the GPIO interrupt
45     GPIOIntClear(GPIO_PORTE_BASE, GPIO_INT_PIN_0);
46     // Read the current state of the GPIO pin and
47     // write back the opposite state
48     if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
49     {
50         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
51     }
```

```
    else
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
    }
    timer1A_delaySec(1);
}

void timer1A_delaySec(int ttime){
    int i;

    SYSCTL_RCGCTIMER_R |= 2;        //enable clock to timer block 1
    TIMER1_CTL_R = 0;               //disable timer before initialization
    TIMER1_CFG_R = 0x04;            //16-bit option
    TIMER1_TAMR_R = 0x02;           //periodic mode and down-counter
    TIMER1_TAILR_R = 64000 - 1;     //timera interval load vlaue reg
    TIMER1_TAPR_R = 250 - 1;        //timera prescaler
    TIMER1_ICR_R = 0x1;             //clear the timera timeout flag
    TIMER1_CTL_R |= 0x01;           //enable timer a after initialization
    for(i=0; i<ttime; i++)
    {
        while((TIMER1_RIS_R & 0x1) == 0);
        TIMER1_ICR_R = 0x1;         //clear the timera timeout flag
    }
}
```