Lab 6: Edge-Triggered Interrupts

Farnam Adelkhani

Part 1 -- Device A

The goal of this lab section is to implement a two-bit rotary counter, using two input switches and two LED outputs. The input switches will be SW1(PF4) and SW2(PF0). The output LEDs will be red(PF1) and blue(PF2).

Code:

```
//! A very simple example that interfaces with the blud LED (PF2) and SW2 (PF0)
//! using direct register access. When SW2 is pressed, the LED is turned on. When
//! SW2 is released, the LED is turned off. Interrupt on PF0 is configured as
//! edge-triggered on both edges
//
//*****************************************************************************

#include <stdint.h>
#include <stdbool.h>
#include "C:\ti\TivaWare_C_Series-2.1.2.111\examples\boards\ek-
tm4c123gxl\switch_delay_interrupt_TivaWare\switch_delay_interrupt_PinMux.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "inc/hw_gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "inc/tm4c123gh6pm.h"       //manually added by the programmer
volatile unsigned long count=0;
//*****************************************************************************

void
PortFunctionInit(void)
{
   //
   // Enable Peripheral Clocks
   //
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

   //
   // Enable pin PF2 for GPIOOutput
   //
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);


                        // Enable pin PF1 for GPIOOutput
   //
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);
                        //

                        //
   // Enable pin PF4 for GPIOInput
   //
```

```
GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4);



                        //
    // Enable pin PF0 for GPIOInput
    //

    //
    //First open the lock and select the bits we want to modify in the GPIO commit register.
    //
HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
HWREG(GPIO_PORTF_BASE + GPIO_O_CR) = 0x1;

    //
    //Now modify the configuration of the pins that we unlocked.
    //
GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0);

                        //************************************************************
                        //          The code above is automatically generated by PinMux
                        //  The code below should be manually added by the programmer
                        //************************************************************

                        //Enable pull-up on PF4 and PF0
                        GPIO_PORTF_PUR_R |= 0x11;

}

void
Interrupt_Init(void)
{
IntEnable(INT_GPIOF);                                          // enable interrupt 30 in NVIC
(GPIOF)
            IntPrioritySet(INT_GPIOF, 0x00);          // configure GPIOF interrupt priority as 0
            GPIO_PORTF_IM_R |= 0x11;                                   // arm interrupt on PF0
            GPIO_PORTF_IS_R &= ~0x11;                      // PF0 is edge-sensitive
            GPIO_PORTF_IBE_R &= ~0x11;                            // PF0 both edges trigger
  GPIO_PORTF_IEV_R &= ~0x11;                    // PF0 falling edge event
            IntMasterEnable();                                          // globally enable interrupt
}

//interrupt handler
void GPIOPortF_Handler(void){
                        // Delay for a bit.
                                    SysCtlDelay(2000000);

  GPIO_PORTF_DATA_R ^= 0x02;


            //SW1 is pressed
            if(GPIO_PORTF_RIS_R&0x10)
            {
                        //Acknowledge flag for PF4
                        GPIOIntClear(GPIO_PORTF_BASE, GPIO_PIN_4);

                        //PF1 is on
                        if((GPIO_PORTF_DATA_R&0x02)==0x00)
                        {
                                        //Toggle PF2
                                        GPIO_PORTF_DATA_R ^=0x04;
                        }
                        //Counter increase by 1
                        count++;
            }

            //SW2 is pressed
 if(GPIO_PORTF_RIS_R&0x01)
            {
                        //Acknowledge flag for PF0
                        GPIOIntClear(GPIO_PORTF_BASE, GPIO_PIN_0);

                        //PF1 is off
                        if((GPIO_PORTF_DATA_R&0x02)==0X02)
                        {
                                        //Toggle PF2
                                        GPIO_PORTF_DATA_R ^=0x04;
                        }
                        //Counter decrease by 1
                        count--;
            }
}



int main(void)
{

                        //initialize the GPIO ports
                        PortFunctionInit();

                        //configure the GPIOF interrupt
                        Interrupt_Init();

   //
   // Loop forever.
   //
while(1)
   {

   }
}
```

<u>Explanation of program:</u>

_____As a result of the above interrupts the program functions as follows. When SW1 is pressed, the counter increments and the red led turns on. When further incrementing the counter the system will turn blue and then display both colors as it counts
00 -> 01 -> 10 -> 11. System B or launchpad B will indicate a green led being on whenever the blue led is on.

Execution results:

_____As a result when the program is compiled onto the launchpad the counter either increments or decrements based on which switch button is pressed. The code successfully helps the device cycle through the two-bit counter in the pattern 00, 01, 10, 11 either forwards or backwards.

Part 2 -- Device B

Code:

```
#include <stdint.h>
#include <stdbool.h>
#include "C:\ti\TivaWare_C_Series-2.1.2.111\examples\boards\ek-
tm4c123gxl\switch_counter_interrupt_TivaWare\switch_counter_interrupt_TivaWare.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "inc/hw_gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/gpio.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/interrupt.h"

//*****************************************************************************
#define blue_mask 0x04
#define read_pin 0x80

void
PortFunctionInit(void)
{
   //
   // Enable Peripheral Clocks
   //
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    //
    // Enable pin PA7 for GPIOInput
    //
GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, read_pin);

    //
    // Enable pin PF2 for GPIOOutput
    //
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, blue_mask);
GPIO_PORTA_PUR_R |=0X11;
}



void
Interrupt_Init(void)
{
IntEnable(INT_GPIOA);                                                   // enable interrupt 30 in NVIC
(GPIOA)
        IntPrioritySet(INT_GPIOA, 0x00);            // configure GPIOF interrupt priority as 0
        GPIO_PORTA_IM_R |= read_pin;                // arm interrupt on PF0 and PF4
        GPIO_PORTA_IS_R &= ~read_pin;     // PF0 and PF4 are edge-sensitive
  GPIO_PORTA_IBE_R |= read_pin;             // PF0 and PF4 not both edges trigger
  GPIO_PORTA_IEV_R &= ~0x80;  // PF0 and PF4 falling edge event
        IntMasterEnable();                    // globally enable interrupt
}

//interrupt handler
void GPIOPortA_Handler(void)
{


        //SW1 is pressed
        if(GPIO_PORTA_RIS_R&read_pin)
        {
                // acknowledge flag for PF4
                GPIOIntClear(GPIO_PORTA_BASE, read_pin);
                        if((GPIO_PORTA_DATA_R&read_pin)==read_pin)
                                GPIO_PORTF_DATA_R |= blue_mask;         //Turn on LED.
                        else
                                GPIO_PORTF_DATA_R &= ~blue_mask;        //Turn off LED.
                }
        }
```

```
int main(void)
{


                        //initialize the GPIO ports
                        PortFunctionInit();


            //configure the GPIOF interrupt
                        Interrupt_Init();


    //
    // Loop forever.
    //
while(1)
    {


    }
}
```

Explanation of program:

_____This program is setup to take an input from PF7 and display a green light whenever its input is positive. This program is setup to respond to the output from launchpad A, which is the counter.

Execution results:

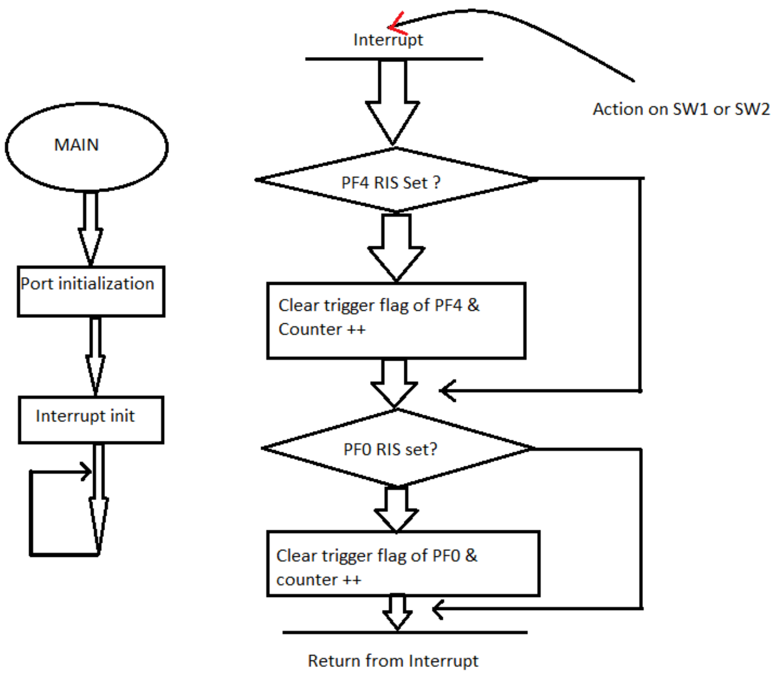
As a result of the system anytime the input to launchpad B is high, the led will turn on and display the green color. The green led will again turn off whenever the counter is decremented.

Discussion and Suggestions:

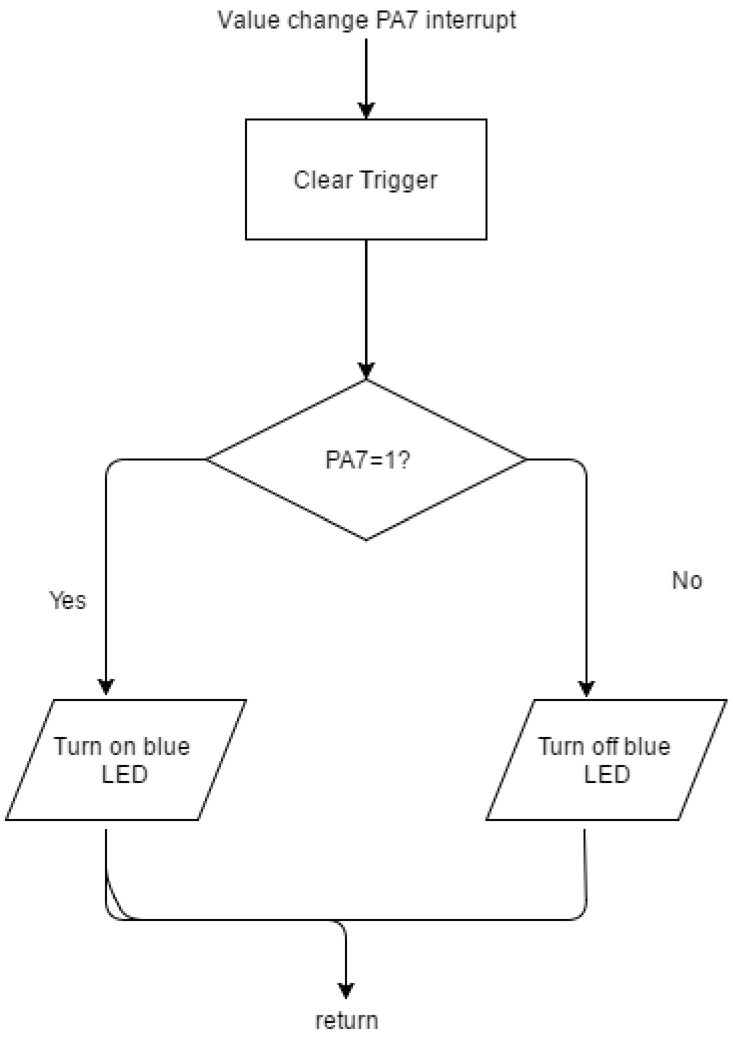
This was the most difficult lab for our group to implement. There was so much to put together and get working together. The biggest challenge was learning about interrupts as well as learning to get the launchpad to count in response to the switches being pressed, while also changing colors to the indicated led. The challenging aspects of this lab are what make it interesting. This was a rewarding lab to get implemented. Interrupts are my favorite part so far because they allow the user to have so much power over the functionality in the program.

Our group does not have any tips for the professor in improving this lab assignment. However, it would benefit the students to clarify things into a package that makes clear sense. Our group looked at several videos from Volvano on his Tilaunchpad class page and that was helpful in understand the required inputs. Rewarding laboratory assignment when implemented.

Flow Charts:



interrupt


Value change PA7 interrupt
Clear Trigger
PA7=1?
Yes
No
Turn on blue LED
Turn off blue LED
return