```
/*
 * interrupt_counter_tut_2B.c
 *
 *  Created on:  Unknown
 *      Author:  Ross Elliot
 *     Version:       1.1
 */

/*********************************************************************************************

* VERSION HISTORY
*********************************************************************************************
*     v1.1 - 01/05/2015
*           Updated for Zybo ~ DN
*
*     v1.0 - Unknown
*           First version created.
*********************************************************************************************/

#include "xparameters.h"
#include "xgpio.h"
#include "xtmrctr.h"
#include "xscugic.h"
#include "xil_exception.h"
#include "xil_printf.h"

// Parameter definitions
#define INTC_DEVICE_ID            XPAR_PS7_SCUGIC_0_DEVICE_ID
#define TMR_DEVICE_ID       XPAR_TMRCTR_0_DEVICE_ID
#define BTNS_DEVICE_ID      XPAR_AXI_GPIO_0_DEVICE_ID
#define LEDS_DEVICE_ID      XPAR_AXI_GPIO_1_DEVICE_ID
#define INTC_GPIO_INTERRUPT_ID XPAR_FABRIC_AXI_GPIO_0_IP2INTC_IRPT_INTR
#define INTC_TMR_INTERRUPT_ID XPAR_FABRIC_AXI_TIMER_0_INTERRUPT_INTR

#define BTN_INT             XGPIO_IR_CH1_MASK
#define TMR_LOAD            0xF8000000

XGpio LEDInst, BTNInst;
```

```c
XScuGic INTCInst;
XTmrCtr TMRInst;
static int led_data;
static int btn_value;
static int tmr_count;

//----------------------------------------------------
// PROTOTYPE FUNCTIONS
//----------------------------------------------------
static void BTN_Intr_Handler(void *baseaddr_p);
static void TMR_Intr_Handler(void *baseaddr_p);
static int InterruptSystemSetup(XScuGic *XScuGicInstancePtr);
static int IntcInitFunction(u16 DeviceId, XTmrCtr *TmrInstancePtr, XGpio *GpioInstancePtr);

//----------------------------------------------------
// INTERRUPT HANDLER FUNCTIONS
// - called by the timer, button interrupt, performs
// - LED flashing
//----------------------------------------------------


void BTN_Intr_Handler(void *InstancePtr)
{
      // Disable GPIO interrupts
      XGpio_InterruptDisable(&BTNInst, BTN_INT);
      // Ignore additional button presses
      if ((XGpio_InterruptGetStatus(&BTNInst) & BTN_INT) !=
                  BTN_INT) {
                  return;
            }
      btn_value = XGpio_DiscreteRead(&BTNInst, 1);
      // Increment counter based on button value
      // Reset if centre button pressed
      led_data = led_data + btn_value;

    XGpio_DiscreteWrite(&LEDInst, 1, led_data);
    (void)XGpio_InterruptClear(&BTNInst, BTN_INT);
    // Enable GPIO interrupts
    XGpio_InterruptEnable(&BTNInst, BTN_INT);
}

void TMR_Intr_Handler(void *data)
{
      if (XTmrCtr_IsExpired(&TMRInst,0)){
            // Once timer has expired 3 times, stop, increment counter
            // reset timer and start running again
            if(tmr_count == 3){
                  XTmrCtr_Stop(&TMRInst,0);
                  tmr_count = 0;
                  led_data++;
                  XGpio_DiscreteWrite(&LEDInst, 1, led_data);
                  XTmrCtr_Reset(&TMRInst,0);
                  XTmrCtr_Start(&TMRInst,0);

            }
            else tmr_count++;
      }
}



//----------------------------------------------------
// MAIN FUNCTION
//----------------------------------------------------
int main (void)
{
  int status;
  //----------------------------------------------------
  // INITIALIZE THE PERIPHERALS & SET DIRECTIONS OF GPIO
  //----------------------------------------------------
  // Initialise LEDs
  status = XGpio_Initialize(&LEDInst, LEDS_DEVICE_ID);
  if(status != XST_SUCCESS) return XST_FAILURE;
  // Initialise Push Buttons
  status = XGpio_Initialize(&BTNInst, BTNS_DEVICE_ID);
  if(status != XST_SUCCESS) return XST_FAILURE;
  // Set LEDs direction to outputs
  XGpio_SetDataDirection(&LEDInst, 1, 0x00);
  // Set all buttons direction to inputs
  XGpio_SetDataDirection(&BTNInst, 1, 0xFF);


  //----------------------------------------------------
  // SETUP THE TIMER
  //----------------------------------------------------
  status = XTmrCtr_Initialize(&TMRInst, TMR_DEVICE_ID);
  if(status != XST_SUCCESS) return XST_FAILURE;
  XTmrCtr_SetHandler(&TMRInst, TMR_Intr_Handler, &TMRInst);
  XTmrCtr_SetResetValue(&TMRInst, 0, TMR_LOAD);
  XTmrCtr_SetOptions(&TMRInst, 0, XTC_INT_MODE_OPTION | XTC_AUTO_RELOAD_OPTION);


  // Initialize interrupt controller
  status = IntcInitFunction(INTC_DEVICE_ID, &TMRInst, &BTNInst);
  if(status != XST_SUCCESS) return XST_FAILURE;

  XTmrCtr_Start(&TMRInst, 0);




  while(1);

  return 0;
}

//----------------------------------------------------
// INITIAL SETUP FUNCTIONS
//----------------------------------------------------

int InterruptSystemSetup(XScuGic *XScuGicInstancePtr)
{
      // Enable interrupt
      XGpio_InterruptEnable(&BTNInst, BTN_INT);
```

```
	XGpio_InterruptGlobalEnable(&BTNInst);

	Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT,
										(Xil_ExceptionHandler)XScuGic_InterruptHandler,
										XScuGicInstancePtr);
	Xil_ExceptionEnable();


	return XST_SUCCESS;

}

int IntcInitFunction(u16 DeviceId, XTmrCtr *TmrInstancePtr, XGpio *GpioInstancePtr)
{
	XScuGic_Config *IntcConfig;
	int status;

	// Interrupt controller initialisation
	IntcConfig = XScuGic_LookupConfig(DeviceId);
	status = XScuGic_CfgInitialize(&INTCInst, IntcConfig, IntcConfig->CpuBaseAddress);
	if(status != XST_SUCCESS) return XST_FAILURE;

	// Call to interrupt setup
	status = InterruptSystemSetup(&INTCInst);
	if(status != XST_SUCCESS) return XST_FAILURE;

	// Connect GPIO interrupt to handler
	status = XScuGic_Connect(&INTCInst,
								INTC_GPIO_INTERRUPT_ID,
								(Xil_ExceptionHandler)BTN_Intr_Handler,
								(void *)GpioInstancePtr);
	if(status != XST_SUCCESS) return XST_FAILURE;


	// Connect timer interrupt to handler
	status = XScuGic_Connect(&INTCInst,
								INTC_TMR_INTERRUPT_ID,
								(Xil_ExceptionHandler)TMR_Intr_Handler,
                                    (void *)TmrInstancePtr);
      if(status != XST_SUCCESS) return XST_FAILURE;

      // Enable GPIO interrupts interrupt
      XGpio_InterruptEnable(GpioInstancePtr, 1);
      XGpio_InterruptGlobalEnable(GpioInstancePtr);

      // Enable GPIO and timer interrupts in the controller
      XScuGic_Enable(&INTCInst, INTC_GPIO_INTERRUPT_ID);

      XScuGic_Enable(&INTCInst, INTC_TMR_INTERRUPT_ID);


      return XST_SUCCESS;
}
```