EE690: Embedded Systems Lab

Lab Assignment 4

Group 10
EE23DP003 Daniel Dsa
EE22DP007 Pradeep Kumar M

GitHub repo: https://github.com/PEG-IITDH/lab-6-blinky-EE23DP003

EK-TM4C123: Implementation of GPIO and Systick Interrupts

1 Aim:

To use the GPIO interrupt and Systick timer interrupt to generate a precisely 1 second long LED pulse each time a button is pressed.

2 Procedure:

1. Enable clock to GPIO Port F and make the Port F configurable.
2. Configure the onboard LEDs as digital outputs and onboard switches as digital inputs.
3. Configure the GPIO PortF interrupt registers for edge detect.
4. Define functions for GPIO interrupt handler and Systick interrupt handler to turn on the LED for 1 second using the Systick timer each time a button is pressed.

3 Documents Referred:

1. TM4C123GH6PM microcontroller datasheet

4 Algorithm:

The steps to use the GPIO interrupt and Systick timer interrupt to generate a precisely 1 second long LED pulse each time a button is pressed is as follows:

1. GPIO port F is configured with LEDs as digital outputs and switches as digital inputs.

   Interrupts for Port F are enabled with falling edge detect interrupt trigger.
2. Interrupts are enabled for GPIO PortF in the NVIC register. The NVIC priority register is configured to give highest priority (0) to the PortF interrupts.
3. A while(1) loop is included in the main.c code to continuously run the program.
4. An interrupt handler function is written for handling GPIO interrupt whenever user switch is pressed. Whenever any of the on-board switch is pressed, the following

occurs in the GPIO_PORTF_Handler function:

(a) The GPIO interrupt flag is cleared using the Interrupt Clear Register (ICR). The interrupts for PORTF corresponding to the two user switches are then masked (disabled).

(b) The on-board LEDs (Red, Blue, Green) are turned ON.

(c) Systick initialization function (Systick_Init) is called. The Systick timer and Systick Interrupt is enabled. The Systick reload register is loaded with a value to produce 1 second delay .

5. An interrupt handler function is written for handling Systick interrupt, which gets triggered when the Systick timer count reaches zero. When the Systick interrupt occurs, the following occurs in the SystickHandler function:

   (a) The on-board LEDs (Red, Blue, Green) are turned OFF.

   (b) The interrupt flag for GPIO Port F is cleared.

   (c) The interrupts for PORTF corresponding to the two user switches are unmasked (enabled) to receive further interrupts.

5 Code:

Lines added / modified in tm4c123gh6pm_startup_ccs.c file:

```
Line 36:  extern void GPIO_PORTF_Handler(void);
Line 37:  extern void SystickHandler(void);
Line 86:  SystickHandler,         //The SysTick handler
Line 117: GPIO_PORTF_Handler     //GPIO_PORTF
```

Main.c file code:

```
/* Lab-4: Systick timer interrupt and GPIO interrupt to generate a precisely 1 second long LED
pulse each time a button is pressed */

#include <stdint.h>
#include <stdbool.h>
#include <tm4c123gh6pm.h>


//Definitions for SysTick CSR (Control and Status Register)
#define ENABLE (1 << 0)      // CSR bit 0 to enable the SysTick timer
#define INT_EN (1<<1)       // CSR bit 1 to generate interrupt to NVIC when Systick reaches 0
#define CLK_SRC (1<<2)     // CSR bit 2 to take system clock

//GPIO and Systick initialization and interrupt handler function declarations
void GPIO_PORTF_Init(void);       //GPIO Port F initialization
void Systick_Init(void);          //SysTick initialization
void GPIO_PORTF_Handler(void);   //GPIO Port F interrupt handler
void SystickHandler(void);       // GPIO Port F Systick handler
```

```c
int main(void)
{

    GPIO_PORTF_Init();        //Initialize GPIO Port F

    //NVIC Configuration
    NVIC_EN0_R |= (1<<30);   //Interrupts enabled for PortF --> interrupt number 30
    NVIC_PRI7_R |= (0<<21) | (0<<22) | (0<<23) ; //Interrupt 30 set to priority highest (0)
priority

    while(1);               // wait indefinitely

}


void GPIO_PORTF_Init(void)
{
    SYSCTL_RCGC2_R |= 0x00000020;   //enable clock to GPIOF
    GPIO_PORTF_LOCK_R = 0x4C4F434B; //Unlock GPIO Port F commit register
    GPIO_PORTF_CR_R = 0x1F;         //Make PORTF configurable

    GPIO_PORTF_DEN_R = 0x1F;        //Enable digital functions on PortF
    GPIO_PORTF_DIR_R = 0x0E;        //Set LEDs as outputs (1), switches as inputs (0)
    GPIO_PORTF_PUR_R = 0x11;        //Pull-up resistor for user switches enabled

    // Enabling Interrupts

     //GPIOIS -- Interrupt Sense  --> 0 for edge detect                           --> 0x00
     GPIO_PORTF_IS_R = 0x00;

     //GPIOIBE -- Both Edges      --> 0 for GPIOIEV controls interrupt generation --> 0x00
     GPIO_PORTF_IBE_R = 0x00;

     //GPIOIEV -- Interrupt Event --> 0 for falling edge triggers interrupt       --> 0x00
     GPIO_PORTF_IEV_R = 0x00;

     //GPIOIM -- Interrupt Mask   --> 0 for masked interrupt (disabled)           --> 0x00
     GPIO_PORTF_IM_R = 0x00;

     //GPIOICR -- Interrupt clear --> cleared by writing 1 to corresponding bit   --> 0x11
     GPIO_PORTF_ICR_R = 0x11;

     //GPIOIM -- Interrupt Mask   --> 1 for unmasked interrupt (enabled)          --> 0x11
     GPIO_PORTF_IM_R = 0x11;


     //GPIORIS -- Raw Int Status (Read Only)
     //GPIOMIS -- Masked Int Status (Read Only)
}


void Systick_Init(void)
{
    NVIC_ST_CURRENT_R = 0x00;                       // SysTick current value register cleared
    NVIC_ST_RELOAD_R = 16000000;                   // Reload value when counter reaches 0
(1sec delay)
    NVIC_ST_CTRL_R |= (ENABLE | INT_EN | CLK_SRC); //Enable Systick, Interrupt and use System
clock
}
```

```
void GPIO_PORTF_Handler(void)        // called whenever GPIO interrupt occurs
{
        GPIO_PORTF_ICR_R = 0x11;     // Clear interrupts from PF0 and PF4
        GPIO_PORTF_IM_R = 0x00;      // Interrupts from PortF masked (disabled)
        Systick_Init();             // Call the Systick function
        GPIO_PORTF_DATA_R = 0x0E;   // Turn on Red (PF1), Blue (PF2) and Green (PF3) LEDs
}


void SystickHandler(void)        // called when Systick reaches 0 and interrupt gets triggered
{
    GPIO_PORTF_DATA_R &=~ 0x0E; //turn off Red, Blue and Green LEDs
    GPIO_PORTF_ICR_R = 0x11;    // Clear any pending interrupts from PF0 and PF4
    GPIO_PORTF_IM_R = 0x11;    // Interrupts from pin PF0 and PF4 unmasked (enabled)
}
```

6 Results:

When any of the user switch is pressed, the GPIO PortF interrupt is triggered, which turns on the onboard LED's and enables the Systick timer to run for 1 second. Upon completion of 1 second, Systick generates a Systick interrupt, in which the onboard LED's are turned off. This can be seen in Fig. 1 to 3.

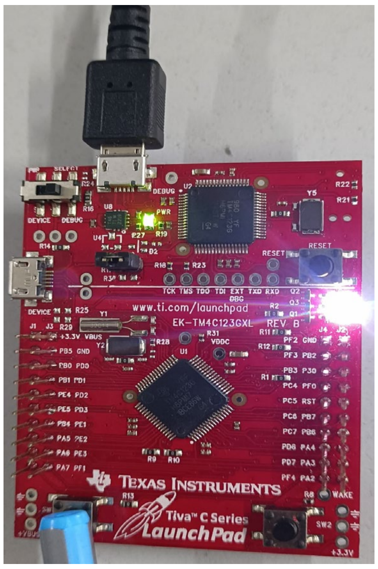

Figure 1 : Hardware Interrupt Triggered

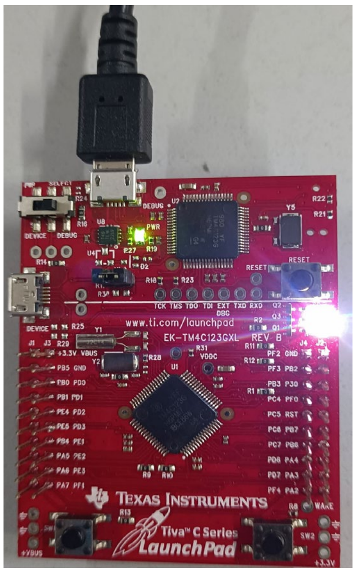

Figure 2: LED remains on as Systick counts for 1 sec

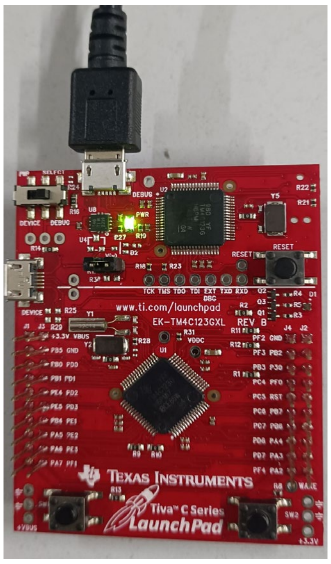

Figure 3: Systick interrupt triggered

The above operation can be verified using an oscilloscope, as shown in Fig. 4 below:

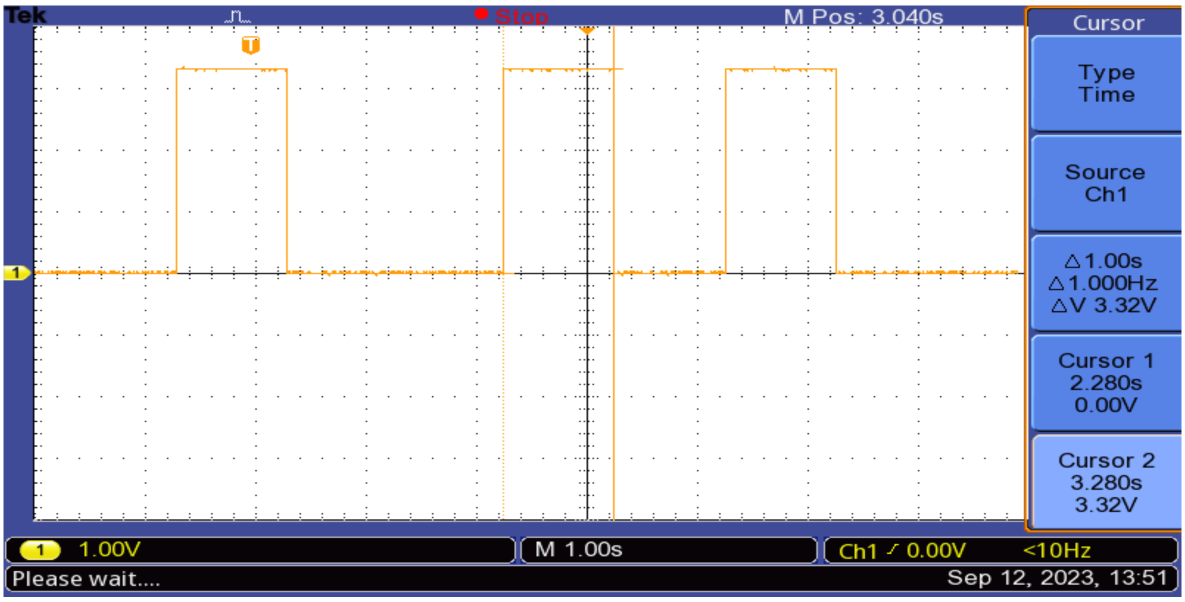

Figure 4: Oscilloscope data for PF1 showing 1 second long pulse each time switch is pressed.

7 Conclusion

The use of the GPIO interrupt for detection of onboard switch press and Systick interrupt when timer counts down to zero allows the microcontroller to perform other necessary tasks if required till the interrupt is generated. This approach is better than periodically polling the user switch and timer, leading to better utilisation of processor resources.