# EMBEDDED SYSTEMS DESIGN

## EXPERIMENT 4 : IMPLEMENTATION OF INTERRUPTS

**NAME : Dixshant Kumar Jha EE23MT025**
**Kaushik Shivanand Powar EE23MT011**

**FACULTY : Dr. ABHIJIT KSHIRSAGAR**

**AIM :**

To turn on the on-board LED on the press the switches on TM4C123GH6PM using Systick and GPIO Interrupts.

**MATERIALS REQUIRED:**

1. EK-TM4C123GH6PM Board
2. Code Compiler Studio
3. Oscilloscope
4. Datasheet

**PROCEDURE :**

1. Configure the GPIO pins of the LED and switches.
2. Enable the interrupt on GPIO using NVIC enable interrupt registers.
3. Enable Systick Interrupt using STCTRL register.
4. Using GPIO and Systick interrupts toggle LED for a duration of one seconds.

**CALCULATIONS:**

For a delay of 1 sec, the clock configured to the Systick counter is 16MHz, the value to be loaded into the Systick Reload Register is $16*10^6$.

**IMPORTANT POINTS:**

Cleared the interrupt flags in the start of the GPIO Interrupt Handler code, and immediately after the GPIO interrupts are masked to avoid any other interrupts during the duration when ISR is running. This is done with intention that even though the button is pressed multiple times , it will generate only a 1 sec pulse from the first press of the button and ignore other presses when the the LED is ON.

**Code:**

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

/* SysTick memory-mapped registers */
#define STCTRL *((volatile long *) 0xE000E010)      // control and status
#define STRELOAD *((volatile long *) 0xE000E014)    // reload value
#define STCURRENT *((volatile long *) 0xE000E018)   // current value

#define MASK_BITS   0x11        //  mask bits for user switch 1 and 2
#define ENABLE      (1 << 0)    // bit 0 of CSR to enable the timer
#define CLKINT      (1 << 2)    // bit 2 of CSR to specify CPU clock
#define INTEN       (1 << 1)    // enable systick interrupt

int main(void)
{
        SYSCTL_RCGC2_R |= 0x00000020;       /* enable clock to GPIOF */
        GPIO_PORTF_LOCK_R = 0x4C4F434B;     /* unlock commit register */
        GPIO_PORTF_CR_R = 0x01;             /* make PORTF0 configurable */
        GPIO_PORTF_DEN_R = 0x1F;            /* set PORTF pins 4 pin */
        GPIO_PORTF_DIR_R = 0x0E;            /* set PORTF4 pin as input user
switch pin */
        GPIO_PORTF_PUR_R = 0x11;            /* PORTF4 is pulled up */


        GPIO_PORTF_IM_R &= ~MASK_BITS;      // mask interrupts for sw1 and
sw2
        GPIO_PORTF_IS_R &= ~MASK_BITS;      // edge sensitive interrupts
        GPIO_PORTF_IEV_R  &= ~MASK_BITS;    // falling edge triggers the
interrupt
        GPIO_PORTF_IBE_R &= ~MASK_BITS;     // Interrupt is controlled by
GPIOIEV

        NVIC_PRI7_R = (NVIC_PRI7_R & 0xFF1FFFFF) | (2 << 21);  // 2 priority
of interrupt 30(port F)
        NVIC_EN0_R |= (1 << 30);            // enable interrupt on port F
        GPIO_PORTF_ICR_R = MASK_BITS;       // clears RIS and MIS for edge
sensitive interrupt
        GPIO_PORTF_IM_R |= MASK_BITS;       // unmask interrupts for sw1 and
sw2
        while(1)
        {
        }
}
```

```
void Systick_Handler(void)
       {
            GPIO_PORTF_DATA_R  = 0x00;      // turn off RED LED
            GPIO_PORTF_ICR_R = MASK_BITS;            // clear interrupt for
port F
            GPIO_PORTF_IM_R |= MASK_BITS;      // unmask interrupts for sw1
and sw2
      }

void GPIOF_INT_Handler(void)
{
    GPIO_PORTF_ICR_R = MASK_BITS;            // clear interrupt for port F
    GPIO_PORTF_IM_R &= ~MASK_BITS;      // mask interrupts for sw1 and sw2
    GPIO_PORTF_DATA_R = 0x02;               // turn on RED LED
    STRELOAD = 1000000*16;                  // for delay of 1 sec
    STCURRENT = 0;
    STCTRL |= (CLKINT | ENABLE | INTEN);    // set internal clock, enable the
timer,enable interrupt
}
```

**Results :**

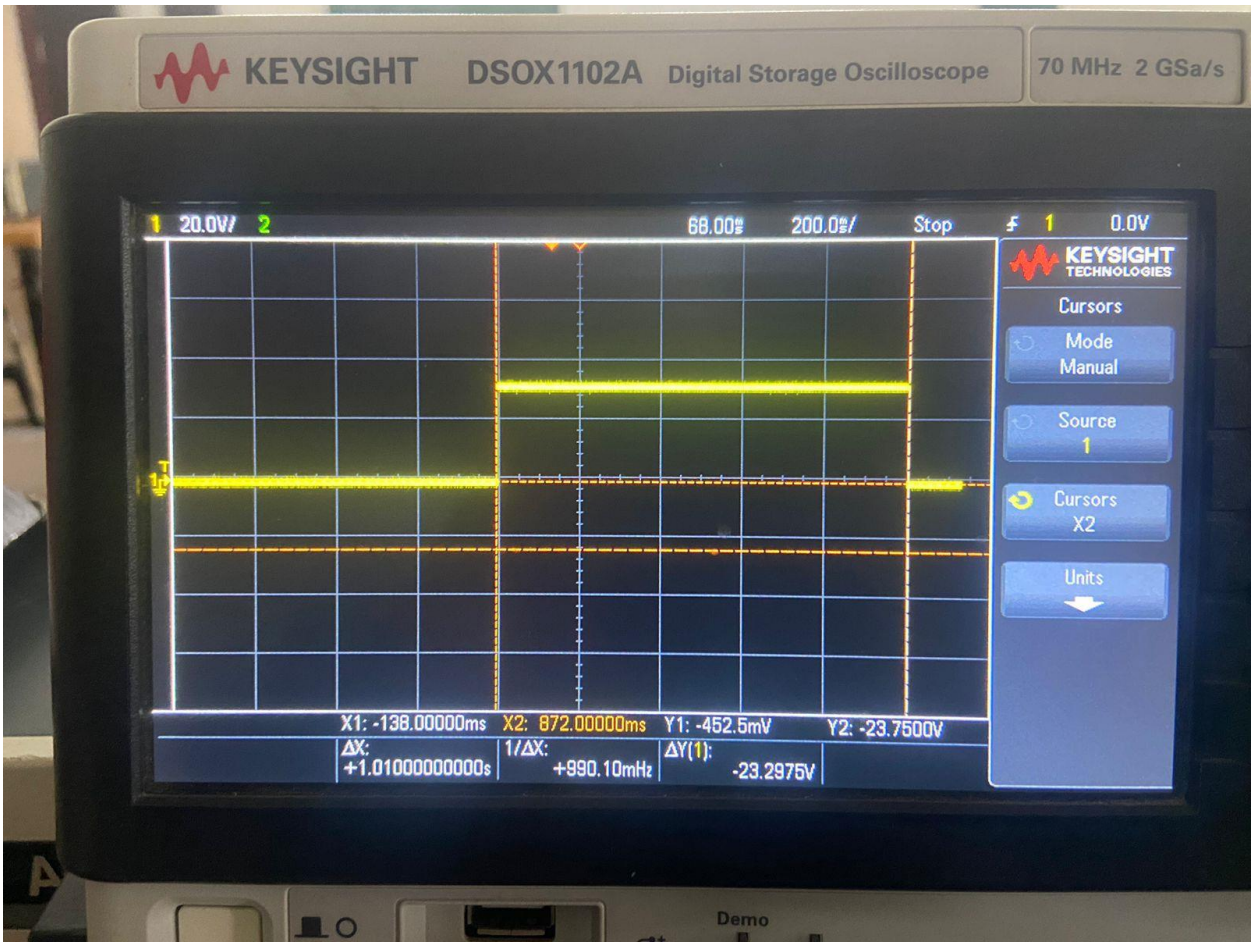


*Figure 1: Oscilloscope waveform for toggling LED on press of a switch*

LED is turned ON for a duration of 1 sec using GPIO and Systick Interrupts(without polling) which keeps the main loop free, free to be used for other functions if any.