Lab-5 Group -4 report

(210020050,210020007)

OBJECTIVE:

- Learn how to use GPIO interrupts to detect switch presses and toggle an LED.

PROBLEM STATEMENT:

- Write a program where the press of a switch is detected by a GPIO interrupt. On each button press, the interrupt handler should toggle the state of a red LED.

APPROACH:

GPIO INTERRUPT ROUTINE:

```
void GPIO_INT_HANDLER(void)
{
   for (int i = 0; i < 100000; i++);
   GPIO_PORTF_DATA_R ^= 0x02;
   GPIO_PORTF_ICR_R = 0x11;
}
```

MAIN ROUTINE:

```
int main(void)
{
   GPIO_INIT();
   GPIO_PORTF_IS_R = 0x00;
   GPIO_PORTF_IBE_R = 0x00;
   GPIO_PORTF_IEV_R = 0x00;
   GPIO_PORTF_IM_R = 0x11;
   NVIC_EN0_R =0x40000000;
  while(1){}
}
```

EXPLANATION:

GPIO Initialization (GPIO_INIT):

- The clock is enabled for **Port F** using the SYSCTL_RCGC2_R register.
- **Pin PF4** (connected to the switch) is configured as an input with a pull-up resistor to detect button presses.
- **Pin PF1** (connected to the red LED) is set as an output to control the LED's state.

Interrupt Configuration:

- The code configures **PF4** to detect falling edges (when the button is pressed) as the source of the interrupt.
- The interrupt is unmasked by setting the appropriate bit in the interrupt mask register (GPIO_PORTF_IM_R).
- The interrupt is enabled for **Port F** using the NVIC enable register (NVIC_EN0_R).

Interrupt Handler (GPIO_INT_HANDLER):

- When an interrupt is triggered by a button press, the interrupt handler toggles the red LED's state using XOR operation (GPIO_PORTF_DATA_R ^= 0x02).
- The interrupt flag is cleared using the interrupt clear register (GPIO_PORTF_ICR_R).

Main Loop:

- The main loop (while(1)) runs indefinitely, allowing the system to wait for interrupts.

OBSERVATION:

- When the switch connected to **PF4** is pressed, the interrupt is triggered, causing the **Red LED** connected to **PF1** to toggle its state (on or off).
- Each subsequent press of the button results in the LED toggling again, confirming the functionality of the GPIO interrupt.
- The system operates efficiently in an interrupt-driven manner, where the main loop remains idle, and the switch press is handled asynchronously.

RESULTS:

- The **GPIO interrupt** successfully detects button presses and executes the interrupt service routine (ISR), toggling the red LED as expected.
- The **falling edge** detection configuration of the interrupt ensures that the ISR is called only when the button is pressed, rather than released.
- The system remains responsive without requiring continuous polling of the button, demonstrating the efficiency of interrupt-based design.