Arduino and Interrupts and timers Lab report

Exercise 1 – External Interrupt (Button on INT0)

1. Arduino Code

```
const int buttonPin = 2;   // INT0
const int ledPin = 13;
volatile bool ledState = false;

void toggleLED() {
  ledState = !ledState;
  digitalWrite(ledPin, ledState);
}

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(buttonPin), toggleLED, FALLING);
}

void loop() {
  // Main loop does nothing – ISR handles LED toggle
}
```

2. Bare-Metal Code

```
#define F_CPU 16000000UL
```

```
#include <avr/io.h>
#include <avr/interrupt.h>

ISR(INT0_vect) {
  PORTB ^= (1 << PORTB5);  // Toggle LED on PB5 (Pin 13)
}

int main(void) {
  DDRB |= (1 << DDB5);        // LED as output
  DDRD &= ~(1 << DDD2);       // PD2 (INT0) as input
  PORTD |= (1 << PORTD2);     // Enable pull-up resistor

  EICRA |= (1 << ISC01);      // Falling edge trigger on INT0
  EIMSK |= (1 << INT0);       // Enable INT0 interrupt
  sei();                      // Enable global interrupts

  while (1);
}
```

Explanation

Condition	EIFR (External Interrupt Flag Register)
Normal (no interrupt)	Flag bit **INTF0 = 0** (no interrupt request pending).
After interrupt triggered	**INTF0 = 1**, set when falling edge detected; cleared automatically when ISR runs or by writing 1 to the flag bit.

Exercise 2 – Pin Change Interrupt on Pin 8

1. Arduino Code

```
const int buttonPin = 8;
const int ledPin = 13;
volatile bool ledState = false;

ISR(PCINT0_vect) {
  if (digitalRead(buttonPin) == LOW) {
    ledState = !ledState;
    digitalWrite(ledPin, ledState);
  }
}

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);
  PCICR |= (1 << PCIE0);    // Enable pin change interrupt group 0 (PORTB)
  PCMSK0 |= (1 << PCINT0);  // Enable interrupt on PB0 (pin 8)
  sei();                    // Global interrupt enable
}

void loop() {}
```

2. Bare-Metal Code

```
#include <avr/io.h>
#include <avr/interrupt.h>

ISR(PCINT0_vect) {
  if (!(PINB & (1 << PINB0))) {
    PORTB ^= (1 << PORTB5);
  }
}

int main(void) {
  DDRB |= (1 << DDB5);     // LED output (Pin 13)
  DDRB &= ~(1 << DDB0);    // Pin 8 as input
  PORTB |= (1 << PORTB0);  // Enable pull-up

  PCICR |= (1 << PCIE0);   // Enable PCINT group 0 (PORTB)
  PCMSK0 |= (1 << PCINT0); // Enable PB0 interrupt
  sei();                   // Enable global interrupts

  while (1);
}
```

Explanation

Register	Purpose	Example Setting
PCICR	Enables pin change interrupt groups	`(1 << PCIE0)`
PCMSK0	Selects pins in PORTB	`(1 << PCINT0)`

PCIFR	Flag register, cleared by writing 1	Used by hardware on ISR entry

Exercise 3 – Timer1 Compare Match Interrupt

Bare-Metal Code

#include <avr/io.h> #include <avr/interrupt.h>

ISR(TIMER1_COMPA_vect) { PORTB ^= (1 << PORTB5); // Toggle LED }

int main(void) { DDRB |= (1 << DDB5); // Pin 13 output

TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10); // CTC mode, prescaler 1024 OCR1A = 15625; // 1 second (for 16 MHz clock) TIMSK1 |= (1 << OCIE1A); // Enable compare match interrupt sei(); // Enable global interrupts

while (1); }

Register Summary

Register	Bits	Purpose
TCCR1B	WGM12, CS12, CS10	CTC mode, prescaler = 1024
OCR1A	—	Compare match value = 15625 for 1 Hz
TIMSK1	OCIE1A	Enable Timer1 Compare Match A interrupt
TIFR1	OCF1A	Flag cleared on ISR execution

Conclusion

Through this lab, we successfully:

- Implemented external, pin change, and timer interrupts using both Arduino functions and register-level (bare-metal) programming.
- Understood the role of key registers such as EICRA, EIMSK, EIFR, PCICR, PCMSKx, TIMSK1, and SREG.
- Verified interrupt latency and flag behaviors using simple LED toggle tests.

Github Link

https://github.com/Djaber-DB/cyber-physical-systems