Experiment No: 04

BECS 31421

INTERRUPT HANDLING WITH A PIC MICROCONTROLLER

Student Name : W.K.G.K Jayawardana

Student Number : EC/2021/006

Partner Name : S.M.A.D.R Pabasarani

Partner Number : EC/2021/007

Date Performed : 11/04/2025

Date of Submission :11/04/2025

DISCUSSION

In this lab, we explored how to use interrupts with the PIC16F628A microcontroller. Also, we mainly focused on handling external interrupts. They are key parts of microcontroller programming because they let the system respond immediately. In this we explored both internal and external interruptions.

` The configuration of interrupts was managed through specific control and flag registers such as INTCON, PIE1, and PIR1. The GIE and INTE bits in the INTCON register were particularly important for enabling global and external interrupts, respectively. Once an interrupt occurred, the microcontroller paused the main execution flow and executed an Interrupt Service Routine (ISR), ensuring timely handling of events.

In this lab, a C program was developed to toggle PORTA every 50 ms in the main loop. Upon an interrupt at RB0, the ISR toggled PORTB pins RB1 and RB2 with a 200 ms delay, demonstrating the interrupt's ability to momentarily divert control flow. The configuration ensured RA5 was disabled to prevent interference. The circuit was first simulated using PROTEUS with a HEX file and later implemented on hardware using the PICkit 3 and MPLAB IPE for programming.

SOURCE CODE

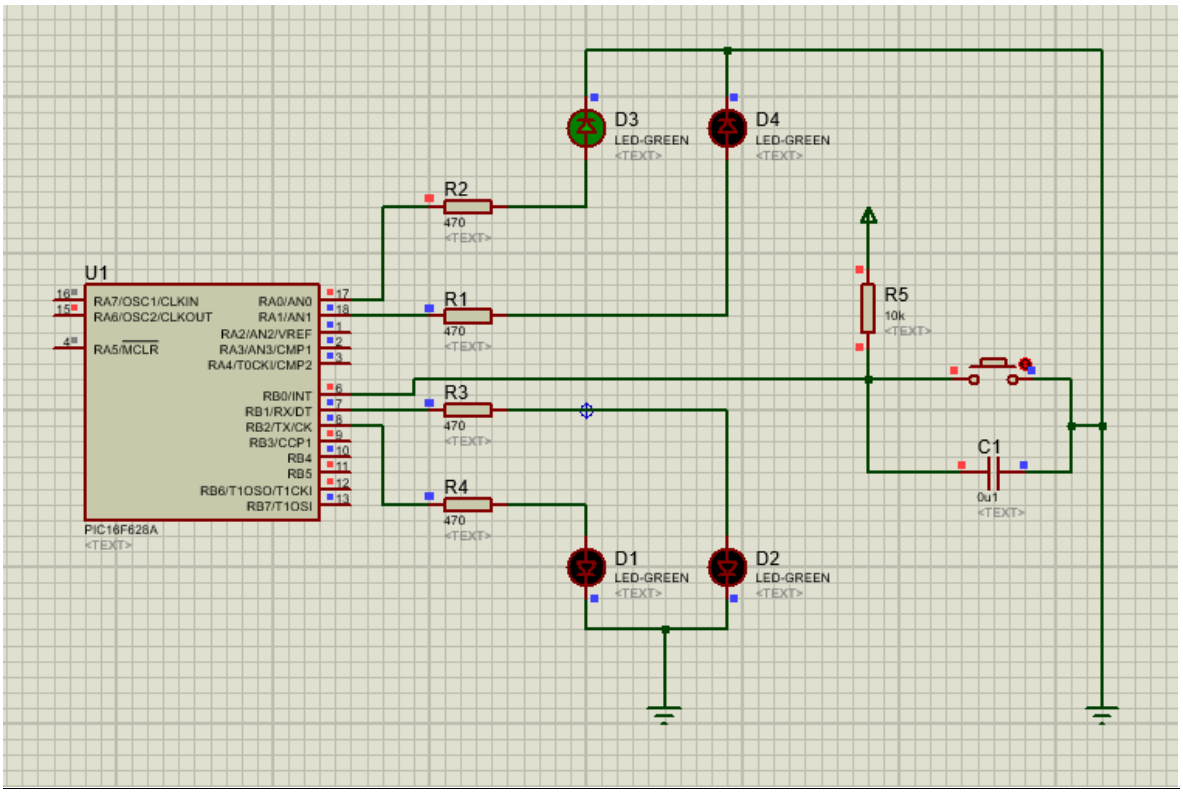
```
// STEP 1: DECLARE THE MAIN FUNCTION
VOID MAIN()
{
    // STEP 2: INITIALIZE CONFIGURATION SETTINGS
    TRISB = 0X01; // HINT: SET RB0 AS INPUT, OTHERS AS OUTPUT
    TRISA = 0X00; // HINT: SET ALL PORT A PINS AS OUTPUT
    CMCON = 0X07; // HINT: DISABLE COMPARATORS
    OPTION_REG = 0X00; // HINT: CONFIGURE OPTION REGISTER
    // STEP 3: ENABLE INTERRUPTS
    INTCON.GIE = 1; // HINT: ENABLE GLOBAL INTERRUPTS
    INTCON.PEIE = 1; // HINT: ENABLE PERIPHERAL INTERRUPTS
    INTCON.INTE = 1; // HINT: ENABLE RB0/INT INTERRUPT
    // STEP 4: DEFINE THE INFINITE LOOP
    WHILE (1){ // HINT: ENTER AN APPROPRIATE CONDITION FOR THE LOOP
      // STEP 5: SET INITIAL PORT VALUES
        PORTB.RB2 = 0; // HINT: SET RB2 TO LOW
        PORTA.RA0 = 1; // HINT: SET RA0 TO HIGH
        PORTA.RA1 = 0; // HINT: SET RA1 TO LOW
        DELAY_MS(50)// HINT: PAUSES THE EXECUTION FOR 100 MILLISECONDS
        // STEP 6: TOGGLE PORT VALUES
        PORTA.RA0 = 0; // HINT: SET RA0 TO LOW
        PORTA.RA1 = 1; // HINT: SET RA1 TO HIGH
        DELAY_MS(50)//HINT: PAUSES THE EXECUTION FOR 100 MILLISECONDS
        INTCON.INTF = 0; // HINT: CLEAR THE EXTERNAL INTERRUPT FLAG
    }
}
    // STEP 7: INTERRUPT SERVICE ROUTINE
VOID INTERRUPT() {
IF (INTCON.INTF) {  // HINT: CHECK THE EXTERNAL INTERRUPT FLAG (INTCON.INTF)
        // STEP 8: SET PORT VALUES UPON INTERRUPT
        PORTB.RB1 = 1; // HINT: SET RB1 TO HIGH
        PORTB.RB2 = 0; // HINT: SET RB2 TO LOW
```

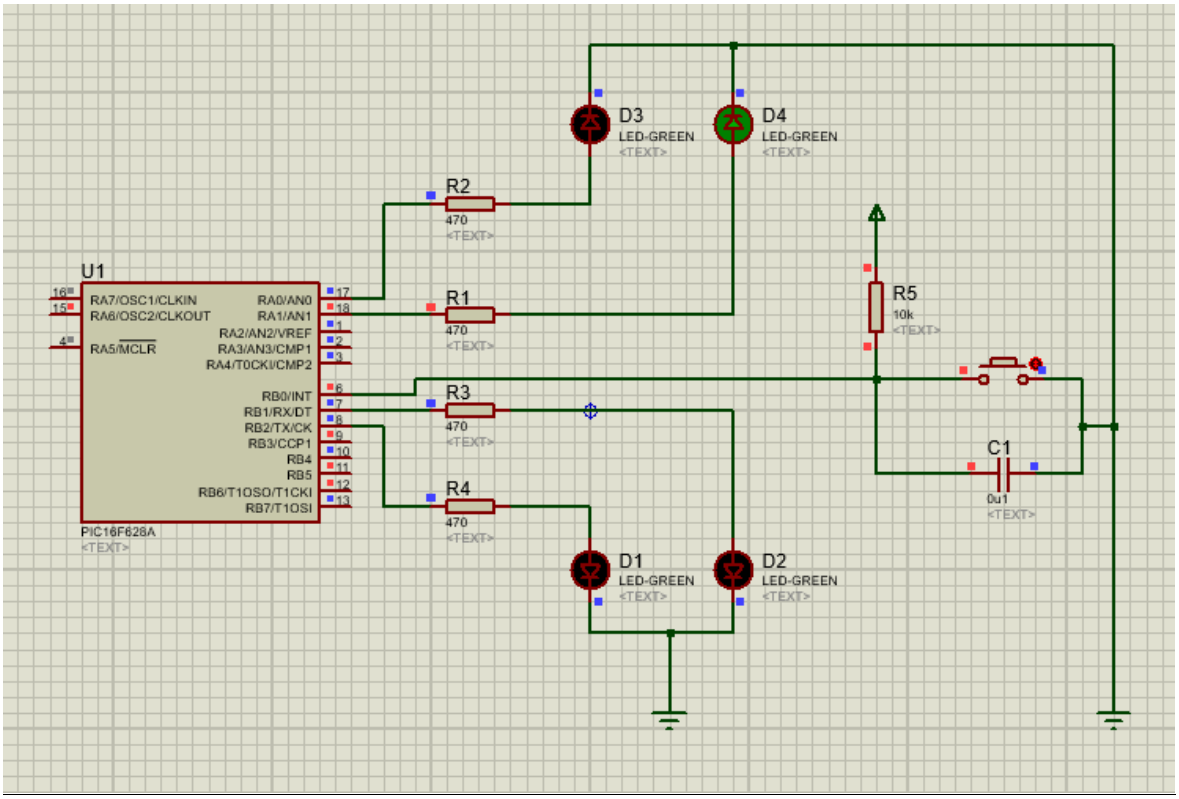
```
        DELAY_MS(200); // HINT: PAUSES THE EXECUTION FOR 100 MILLISECONDS
        // STEP 9: TOGGLE PORT VALUES
         PORTB.RB1 = 0; // HINT: SET RB1 TO LOW
         PORTB.RB2 = 1; // HINT: SET RB2 TO HIGH
         DELAY_MS(200)//HINT: PAUSES THE EXECUTION FOR 100 MILLISECONDS
         INTCON.INTF = 0; // HINT: CLEAR THE EXTERNAL INTERRUPT FLAG
    }
}
```

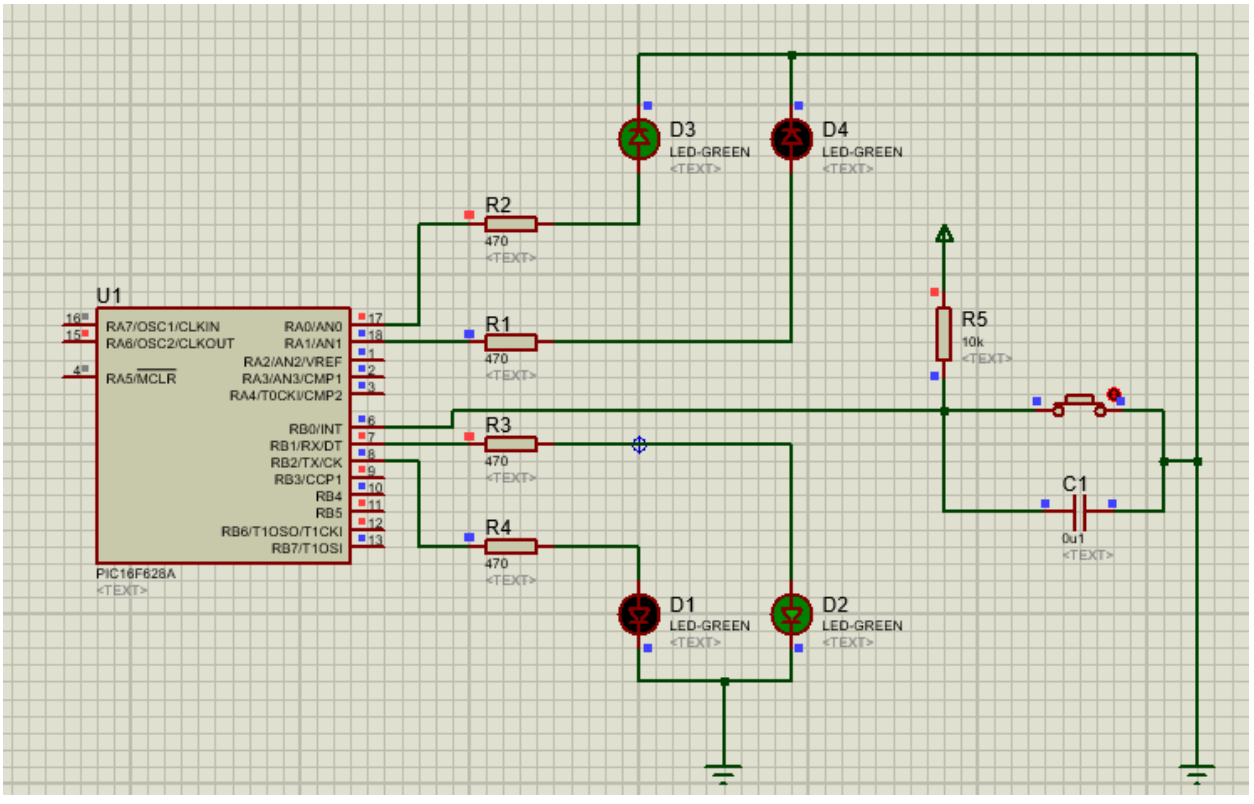
Start Page | Lab_04.c

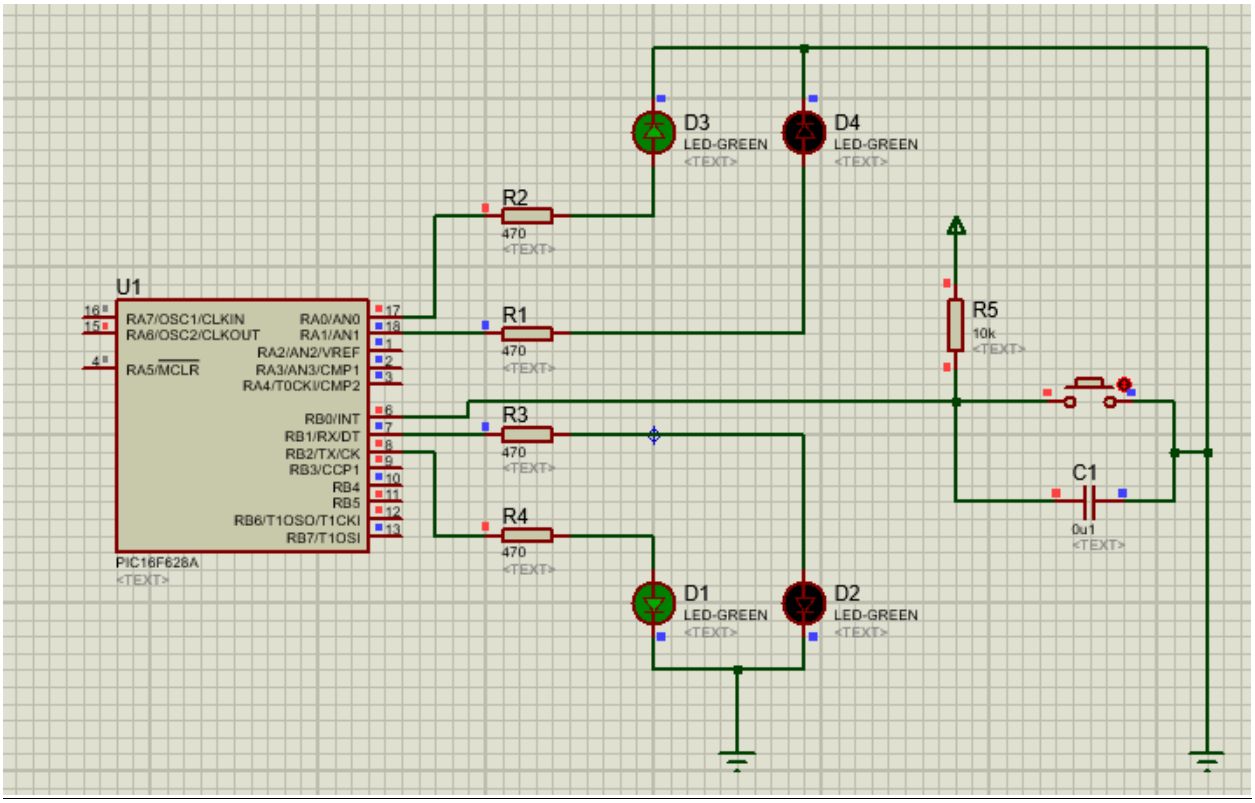
```
// Step 1: Declare the main function
void main()
{
    // Step 2: Initialize configuration settings
    TRISB = 0x01; // Hint: Set RB0 as input, others as output
    TRISA = 0x00; // Hint: Set all port A pins as output
    CMCON = 0x07; // Hint: Disable comparators
    OPTION_REG = 0x00; // Hint: Configure option register
    // Step 3: Enable interrupts
    INTCON.GIE = 1; // Hint: Enable global interrupts
    INTCON.PEIE = 1; // Hint: Enable peripheral interrupts
    INTCON.INTE = 1; // Hint: Enable RB0/INT interrupt
    // Step 4: Define the infinite loop
    while (1){ // Hint: Enter an appropriate condition for the loop
      // Step 5: Set initial PORT values
        PORTB.RB2 = 0; // Hint: Set RB2 to low
        PORTA.RA0 = 1; // Hint: Set RA0 to high
        PORTA.RA1 = 0; // Hint: Set RA1 to low
        delay_ms(50);// Hint: Pauses the execution for 100 milliseconds
        // Step 6: Toggle PORT values
        PORTA.RA0 = 0; // Hint: Set RA0 to low
        PORTA.RA1 = 1; // Hint: Set RA1 to high
        delay_ms(50);//Hint: Pauses the execution for 100 milliseconds
        INTCON.INTF = 0; // Hint: Clear the external interrupt flag
    }
}
    // Step 7: Interrupt service routine
void interrupt() {
if (INTCON.INTF) {  // Hint: Check the external interrupt flag (INTCON.INTF)
      // Step 8: Set PORT values upon interrupt
      PORTB.RB1 = 1; // Hint: Set RB1 to high
      PORTB.RB2 = 0; // Hint: Set RB2 to low
      delay_ms(200); // Hint: Pauses the execution for 100 milliseconds
      // Step 9: Toggle PORT values
       PORTB.RB1 = 0; // Hint: Set RB1 to low
       PORTB.RB2 = 1; // Hint: Set RB2 to high
       delay_ms(200);//Hint: Pauses the execution for 100 milliseconds
       INTCON.INTF = 0; // Hint: Clear the external interrupt flag
    }
}
```

SIMULATION SCREENSHOTS






U1
RA7/OSC1/CLKIN
RA6/OSC2/CLKOUT
RA5/MCLR
RA0/AN0
RA1/AN1
RA2/AN2/VREF
RA3/AN3/CMP1
RA4/T0CKI/CMP2
RB0/INT
RB1/RX/DT
RB2/TX/CK
RB3/CCP1
RB4
RB5
RB6/T1OSO/T1CKI
RB7/T1OSI
PIC16F628A
R1
470
R2
470
R3
470
R4
470
R5
10k
C1
0u1
D1
LED-GREEN
D2
LED-GREEN
D3
LED-GREEN
D4
LED-GREEN


U1
RA7/OSC1/CLKIN
RA6/OSC2/CLKOUT
RA5/MCLR
RA0/AN0
RA1/AN1
RA2/AN2/VREF
RA3/AN3/CMP1
RA4/T0CKI/CMP2
RB0/INT
RB1/RX/DT
RB2/TX/CK
RB3/CCP1
RB4
RB5
RB6/T1OSO/T1CKI
RB7/T1OSI
PIC16F628A
R1
470
R2
470
R3
470
R4
470
R5
10k
C1
0u1
D1
LED-GREEN
D2
LED-GREEN
D3
LED-GREEN
D4
LED-GREEN