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

Experiment No: 04

BECS 31421

**INTERRUPT HANDLING WITH A PIC MICROCONTROLLER**

**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

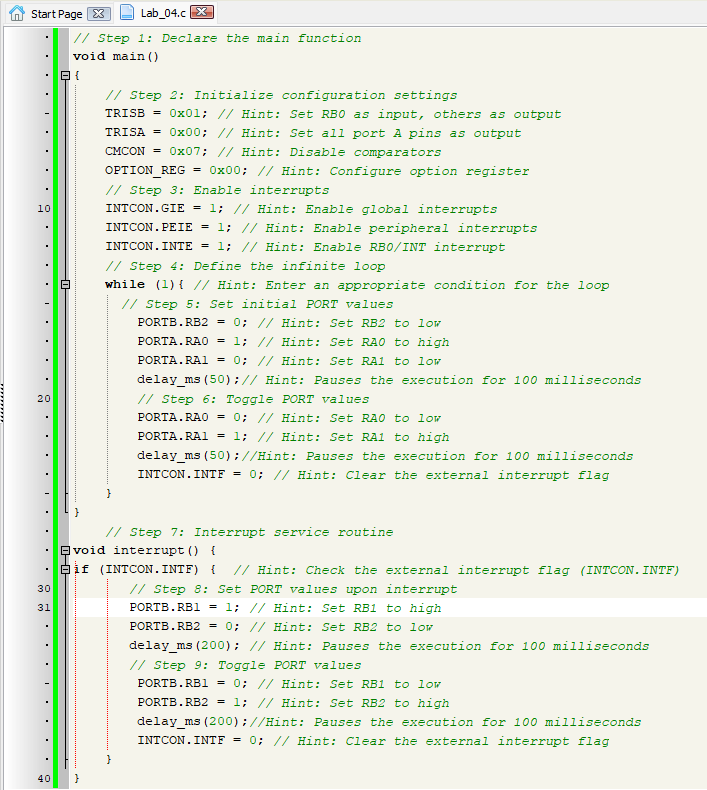
PORTB.RB2 = 1; // Hint: Set RB2 to high

delay\_ms(200);//Hint: Pauses the execution for 100 milliseconds

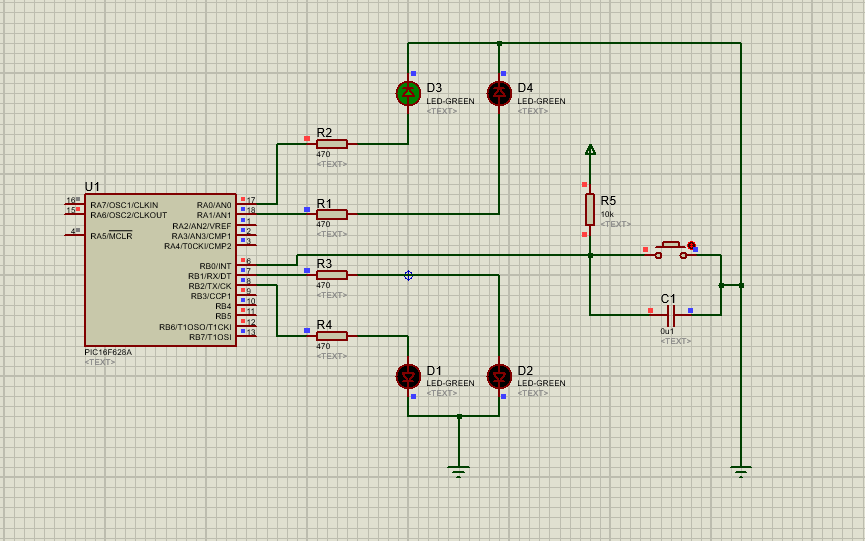
INTCON.INTF = 0; // Hint: Clear the external interrupt flag

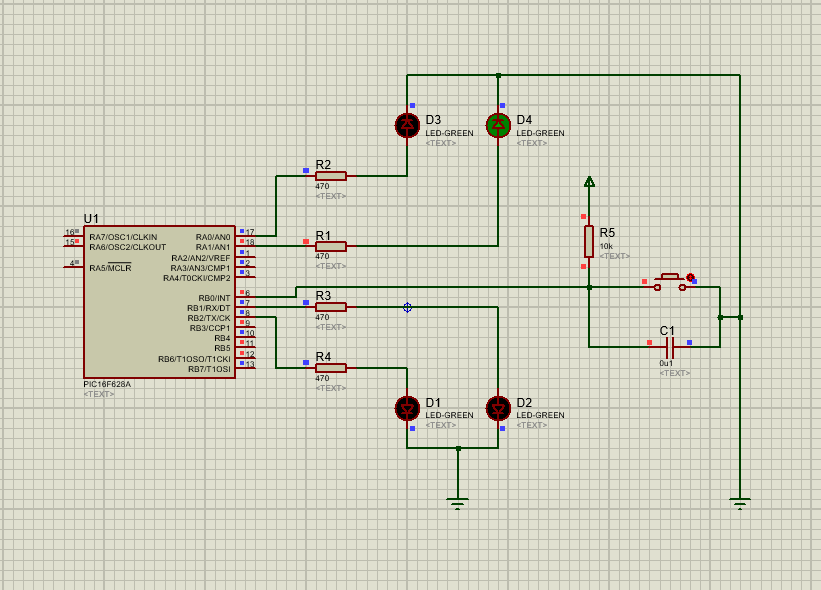
}

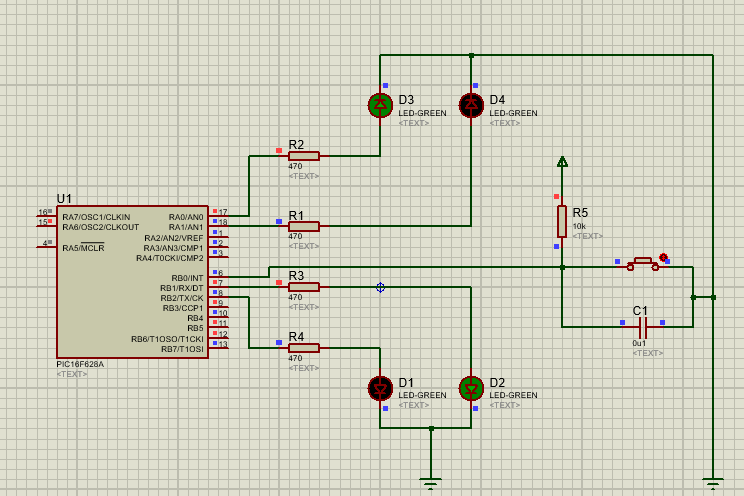
}

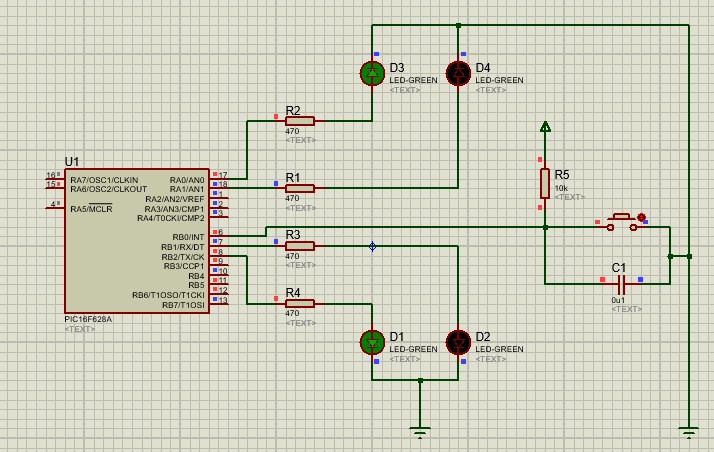
****

**SIMULATION SCREENSHOTS**

****

****

****

****