EEE 447 MICROPROCESSOR LABORATORY

EXPERIMENTAL WORK 6

INTERRUPT PROGRAMMING IN ASSEMBLY

Objective: In this experiment, students will:

- Understand the concept of **external hardware interrupts** on the PIC18F452.
- Learn how to configure and use **INT0 (RB0)** interrupt in assembly.
- Implement an **interrupt service routine (ISR)**.
- Observe the difference between polling and interrupt-driven execution.

Theory

The PIC18F452 has three external interrupt pins: INT0 (RB0), INT1 (RB1), and INT2 (RB2). When a rising or falling edge is detected on these pins (depending on configuration), an interrupt request is generated. Upon activation of these pins, the PIC18 gets interrupted in whatever it is doing and jumps to the vector table to perform the interrupt service routine.

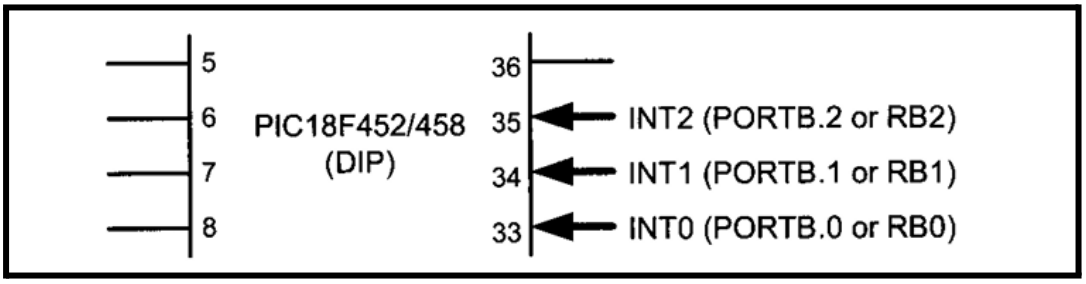

Figure 1 PIC18 External Hardware Interrupts Pins

Upon reset, all interrupt sources are disabled by default. This means the microcontroller will not respond to any interrupt requests unless both the **Global Interrupt Enable (GIE)** bit and the specific **interrupt enable bit** are set in software.

Interrupt (Pin)	Flag bit	Register	Enable bit	Register
INT0 (RB0)	INT0IF	INTCON	INT0IE	INTCON
INT1 (RB1)	INT1IF	INTCON3	INT1IE	INTCON3
INT2 (RB2)	INT2IF	INTCON3	INT2IE	INTCON3

Figure 2 Hardware Interrupt Flag Bits and Associated Registers

The D7 bit of INTCON (Interrupt Control) register is responsible for enabling and disabling the interrupts globally. The GIE bit makes the job of disabling all the interrupts easy.

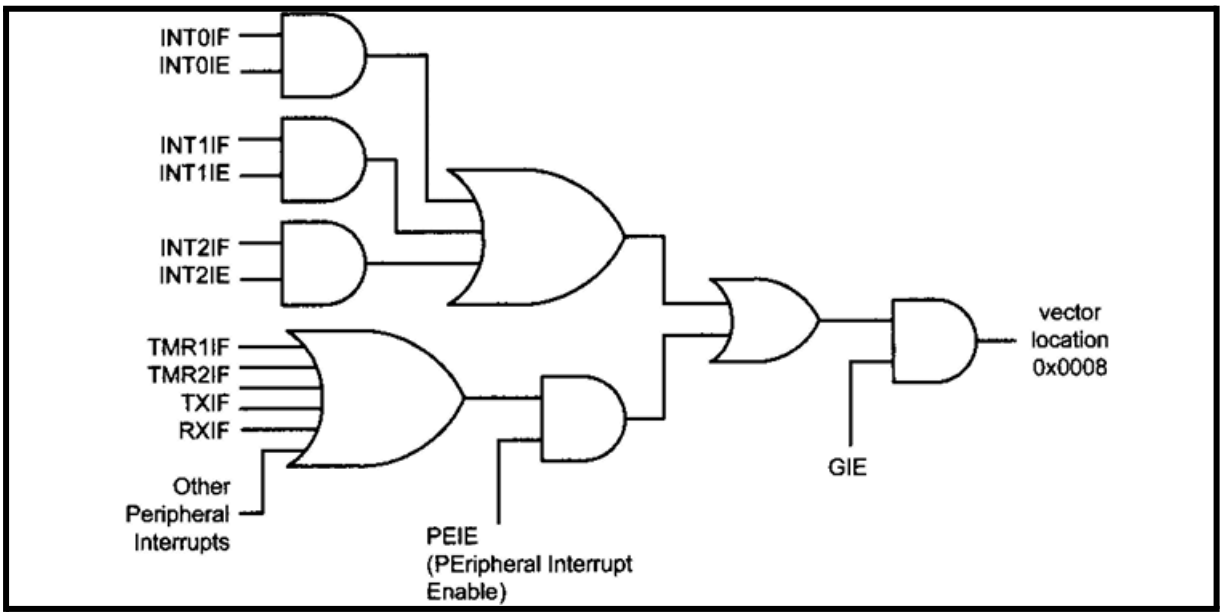

Figure 3 INT0 - INT2 Hardware Interrupts

To use INT0:

1. Configure RB0 as input.
2. Clear INT0IF (if needed).
3. Set INT0IE to enable INT0.
4. Set GIE to enable global interrupts.
5. Write an ISR (interrupt service routine) at address 0x08 (in Assembly) or in void interrupt block (in C).

Experimental Work

Step 1: Assembly Implementation

- In the main loop, PORTC LEDs will shift from right to left using bitwise logic.
- INT0 (RB0) will be configured as an interrupt source.
- When the button is pressed (falling edge), the processor jumps to ISR.
- The ISR blinks RD3 three times with software delay, then returns to main.

Step 2: C Implementation

- The same logic will be implemented in MPLAB XC8.
- PORTC LEDs will rotate in the main loop.
- INT0 will be configured using SFR bits in C.
- When INT0 triggers, the ISR will blink RD3 three times.

Step 3: Compile and Upload

- Assemble or compile the code using MPLAB X IDE.

- Load the generated HEX file into the PIC18F452 on the EasyPIC V7 board.

Step 4: Test and Observe

- Observe PORTC LEDs shifting in sequence.
- Press the button on RB0 to trigger the interrupt.
- Verify that RD3 LED blinks three times upon interrupt, then the sequence resumes.