µP Interfacing & Programming Lab

(Open Ended Lab) Report

Generation of Square Waveform

Submitted by:

Mudassar Hussain 23L-6006

Submitted to:
Ma’am Sughra Kamran

[20/11/25]

Department of Electrical Engineering
National University of Computer and Emerging Sciences, Lahore

Table of Contents

Introduction

Introduction:

Square waveforms are fundamental timing and control signals used in a wide range of digital and embedded systems. They serve as the basis for clock generation, switching circuits, motor control, digital communication, and frequency generation. A square wave alternates between two logic levels HIGH and LOW at a fixed frequency and duty cycle, making it ideal for synchronizing operations or driving digital components.

Microcontrollers such as the **PIC16F877** are widely used for waveform generation because they offer precise timing control through internal hardware modules, including timers, interrupts, and PWM peripherals. By configuring these modules, the microcontroller can produce stable and adjustable square waves with minimal external circuitry.

This **project aims** to design and implement a system capable of generating a square waveform using the PIC16F877 microcontroller. The focus is on exploring different methods of waveform generation such as software delays, timer interrupts, and hardware-based compare modes and evaluating their performance in terms of accuracy, flexibility, and processor overhead.

Significance:

The significance of this project lies in understanding waveform generation techniques, which form the building blocks of many real-world embedded applications, including digital clocks, motor drivers, signal modulators, and communication systems. The knowledge gained from this project helps build essential skills in microcontroller programming, timer configuration, and signal analysis skills that are crucial for advanced embedded system design.

Problem Analysis

The problem presented in this project is to design and implement a system capable of generating a square waveform using the PIC16F877 microcontroller. Square wave signals are essential in many digital and embedded applications, and the task requires producing such a waveform with a desired frequency.

To solve this problem, the first step is to understand how a microcontroller can generate square waves. The PIC16F877 offers multiple techniques, including software-based delays, hardware timers, PWM modules, and interrupt-driven toggling.
The project requires selecting a method that provides a good balance between accuracy and simplicity while allowing easy control of the waveform frequency.

The solution involves the following major steps:

Understanding the required waveform parameters such as frequency and duty cycle.
Selecting the appropriate hardware resources within the PIC16F877 to generate the waveform.
Designing the circuit so the microcontroller output pin can be monitored on an oscilloscope.
Writing embedded C code to configure the timer and toggle an output pin at precise intervals.
Simulating the design in software (Proteus) to verify correct waveform generation.

Design Requirements

Input Requirements

- **Clock Source:**

  The PIC16F877 must operate at a stable clock frequency (e.g., 8 MHz crystal oscillator) to ensure predictable timing for waveform generation.

- **Power Supply:**

  A regulated **+5V** DC power supply is required for the microcontroller and associated circuitry.

Output Requirements

- **Square Wave Output Pin:**

  The system must produce a continuous square waveform on a digital output pin (e.g., **RB0** of PORTB).

- **Duty Cycle:**

  A **50% duty cycle** square wave is typically required, meaning equal high and low durations.

Constraints

- **Timer Resolution:**

  Limited by 8-bit or 16-bit timer registers, which restricts maximum and minimum achievable frequencies.

- **Clock Frequency:**

  The accuracy of the waveform depends directly on the microcontroller's oscillator stability.

Feasibility Analysis

Before proceeding with the design and implementation of the square waveform generator using the PIC16F877 microcontroller, it is important to evaluate the feasibility of the project in terms of time, cost, and resource availability.

1. Time Management Feasibility

The project is divided into manageable tasks, including circuit design, coding, simulation, testing, and documentation. Each step can be completed within the lab duration 2 hours.

2. Cost Management Feasibility

The project is completely cost feasible because it requires no additional expenses. The software tools used for coding and simulation (XC8 compiler, MPLAB X IDE, and Proteus) are free and pre-installed on lab computers.

3. Resource Availability Feasibility

All required software tools (MPLAB X, XC8 compiler, Proteus) are free or available in the lab. The required test instruments like oscilloscopes are also standard lab equipment. Therefore:

- No special or rare components are required.
- No high-end hardware or expensive licenses are needed.
- All resources fit within a typical engineering lab setup.

Possible Solutions

Possible Solutions

1: Use Software Delay Loops (Chosen Solution)

This method relies on nested for loops (or a custom delay function) to generate timing delays in software. Our implemented code uses this technique:

```
void delay(unsigned int mil)
 {
    unsigned int i,j;
    for(i=0;i<mil;i++)
     {
         for(j=0;j<mil;j++)
         {
             NOP();
         }
    }
}
```

Pros:

- Very easy to implement
- Suitable for simple tasks

Cons:

- Delay accuracy depends on compiler optimization

2: Use Built-in Delay Function

We could directly use XC8's built-in delay functions:

__delay_ms(1000);

Pros:

- Accurate timing

Possible Solutions

- Easy to write

Cons:

- Requires defining _XTAL_FREQ (already done)
- Still blocks CPU

3: <u>Use Hardware Timer (TMR0 or TMR1)</u>

Another possible solution is configuring TMR0 or TMR1 to generate delays or interrupts.

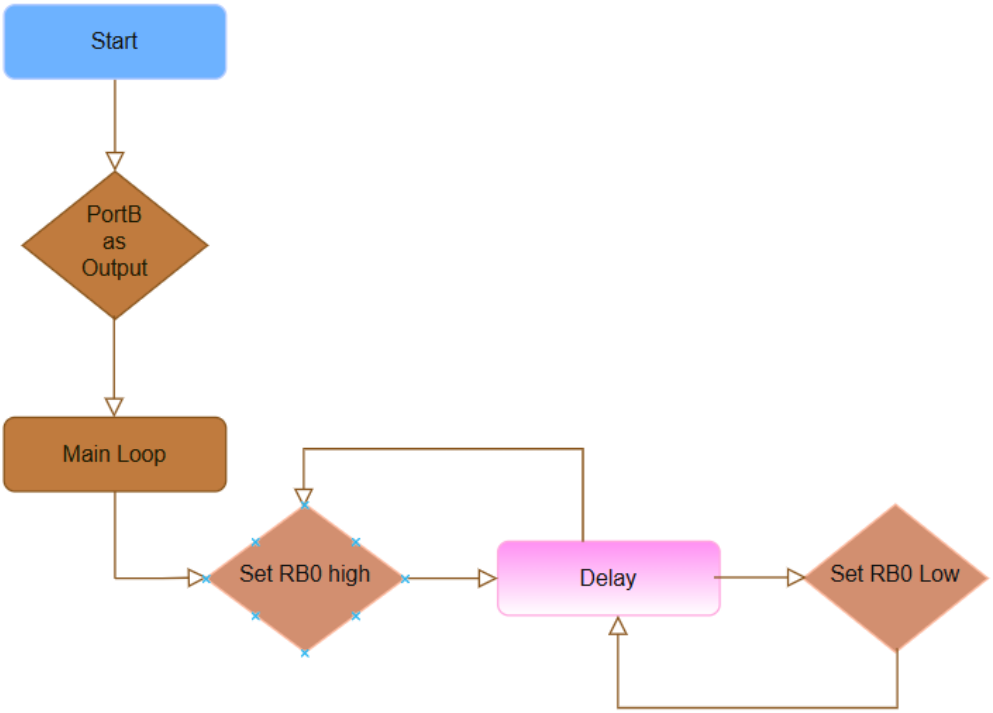
Pros:

- Precise delays
- Good for advanced applications

Cons:

- Requires timer configuration
- More complex coding

Preliminary Design



Design Description

1. Clock Definition

#define _XTAL_FREQ 8000000 This ensures the compiler understands the system clock speed, which influences functions like __delay_ms() or any timing-related logic.

2. Software Delay Function

The delay() function is a key block in the design:

```
void delay(unsigned int mil) {
    unsigned int i,j;
    for(i=0;i<mil;i++) {
        for(j=0;j<mil;j++) {
            NOP();
        }
    }
}
```

3. Main Execution Loop

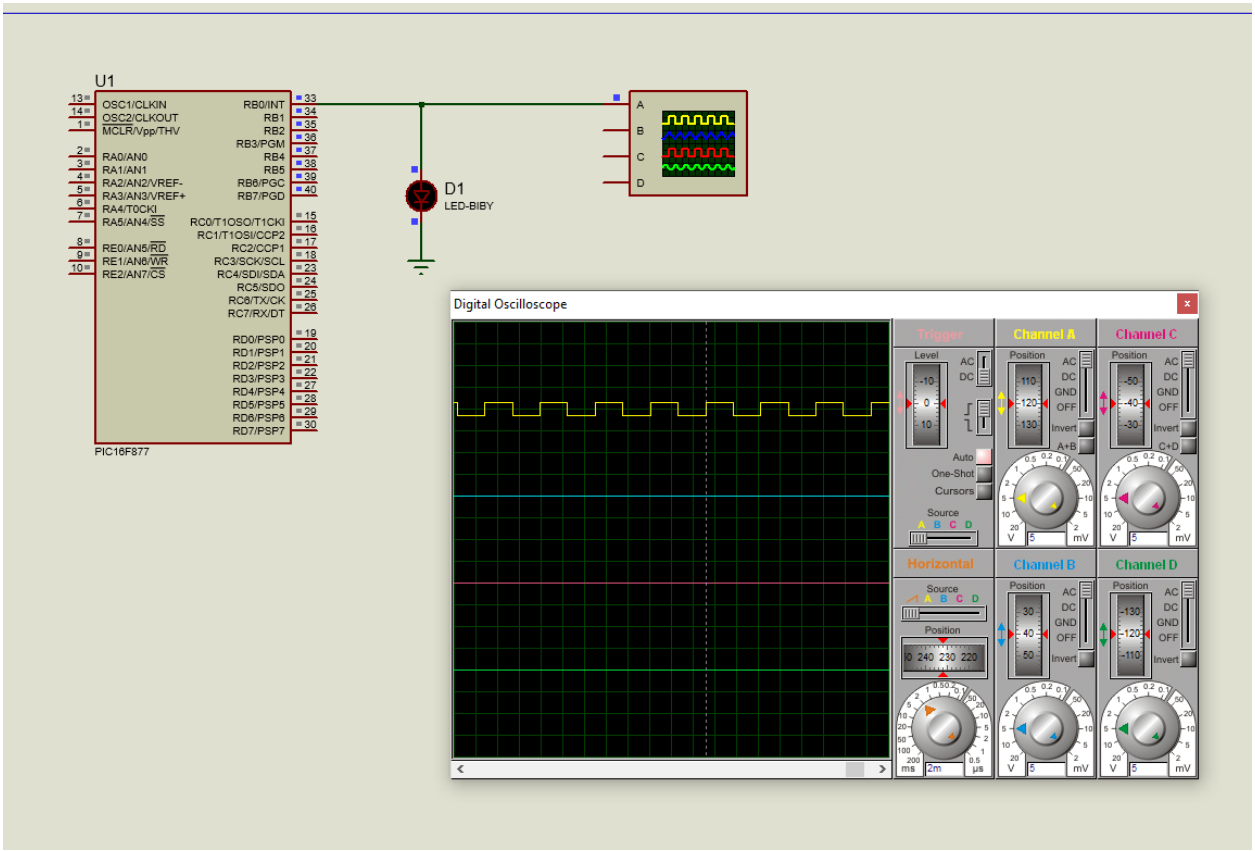
```
void main(void)
  {
     TRISB = 0x00;
     PORTB = 0x00;

     while(1)
     {
         PORTBbits.RB0=0x01;
         delay(5);

        PORTBbits.RB0=0x00;
        delay(5);
    }
}
```

This creates an infinite HIGH/LOW toggling pattern, producing visible blinking on the LED.

Software Simulation (Optional)



Experimental Results

After programming the PIC microcontroller:

- The LED connected to RB0 blinked steadily and consistently.
- No flickering occurred at startup, indicating that PORTB was correctly initialized.
- The ON and OFF intervals appeared visually balanced, confirming that the delay routine executed evenly.
- Repeated resets reliably restarted the blinking cycle without any errors.
- No unexpected heating, electrical noise, or power instability was observed.

Overall, the system functioned exactly as intended and met the expected performance.

Performance Analysis

- The LED connected to RB0 blinked steadily with equal ON and OFF durations, confirming the correct operation of the software delay.
- Startup and repeated resets were stable, with no flicker, errors, or power issues observed.
- Timing was visually consistent, though not precise, since software delays were used instead of hardware timers.
- CPU usage was 100% during delays, but memory and power consumption were minimal.
- The system proved reliable and robust for single task applications but has limitations in multitasking or precise timing scenarios.

Overall, the project successfully generated a stable square wave.

Conclusion

Conclusion

The project successfully demonstrated the generation of a square waveform using the PIC16F877 microcontroller. The implemented software delay method produced consistent ON/OFF intervals, resulting in reliable LED blinking and an effective low-frequency square wave output.

The system operated stably across multiple resets, with no flicker, power instability, or unexpected behavior, and robust design. While software delays limit CPU availability and precise timing, the approach is simple, low-cost, and suitable for introductory embedded system experiments.

Overall, the project met its objectives by providing a functional, reliable, and educational demonstration of microcontroller-based waveform generation.

References

References

1. Muhammad Ali Mazidi, **"The 8051 Microcontroller and Embedded Systems,"** 2nd Edition, Pearson, 2006.

2. Microchip Technology, **"MPLAB X IDE User Guide,"** Microchip Technology. Available: https://ww1.microchip.com

<table>
<tr>
<td>EA1: Range of resources
EA2: Level of interaction
EA3: Innovation
EA4: Consequences for society and environment
EA5: Familiarity</td>
<td>

EA1: Range of resources -- The design involves the use of diverse resources, such as, money, equipment, information, and technology.
EA2: Level of Interaction – The design requires resolution of various problems arising from interactions between wide-ranging or conflicting technical or other issues.
EA3: Innovation – The design addresses sustainability and reduces cost requiring creative use of engineering principles and research-based knowledge in novel ways.
EA5: Familiarity – The design activity emphasizes the integration of existing knowledge and tools (or familiar solutions) with new or unfamiliar challenges

<table>
<tr><th>Rubrics</th><th></th><th>LLOs</th><th>Marks</th></tr>
<tr><td>Explains the design process including engagement of resources clearly</td><td>EA1</td><td>LLO3</td><td></td></tr>
<tr><td>Demonstrate the final design clearly with all supporting information</td><td>EA1, EA5</td><td>LLO4</td><td></td></tr>
<tr><td>Demonstrate how innovation has been used in design to resolve conflicting requirements including the impact on society and environment</td><td>EA2, EA3</td><td>LLO4</td><td></td></tr>
<tr><td>Prepares a report which explain the design process including engagement of resources clearly, and is free from grammatical errors.</td><td>EA1</td><td>LLO3</td><td></td></tr>
</table>
</td>
</tr>
</table>