KUDZAI K. BANDERA

KELTEK

PROJECT Tittle: LED Control Using PIC16F84A Microcontroller

OBJECTIVE:

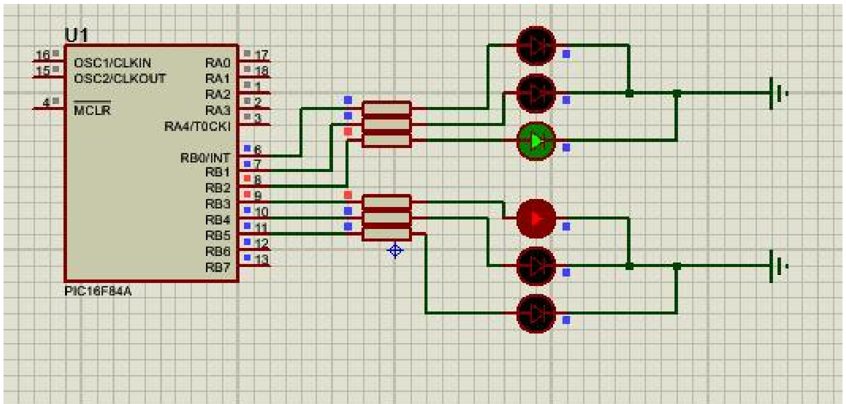
To design and simulate a simple LED control system using the PIC16F84A microcontroller in Proteus, showcasing basic digital output control and circuit simulation.

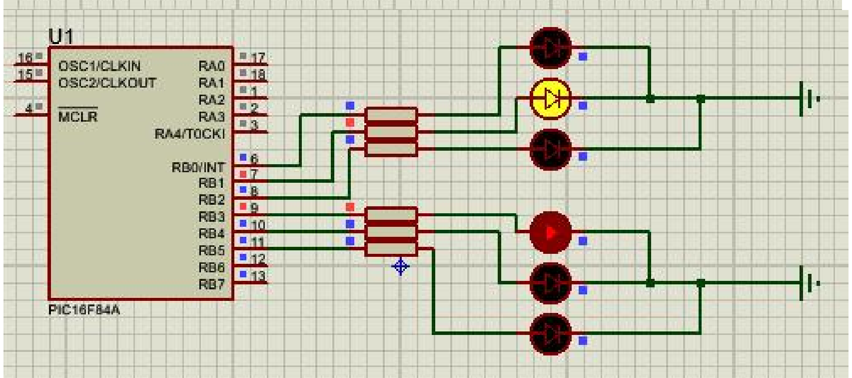
Tools and Components Used:

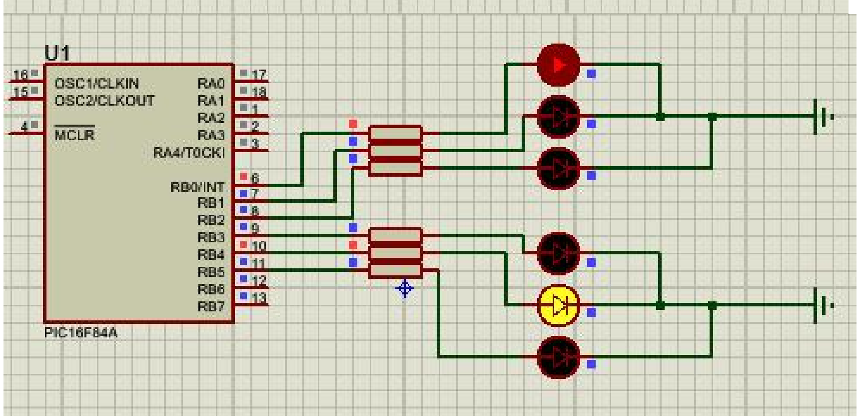
- **Microcontroller:** PIC16F84A
- **Simulation Software:** Proteus Design Suite
- **Components:**
    - LEDs (6 units)
    - Resistors (150Ω each)
    - Breadboard and virtual power supply
    - Microcontroller code (assembled to HEX format)

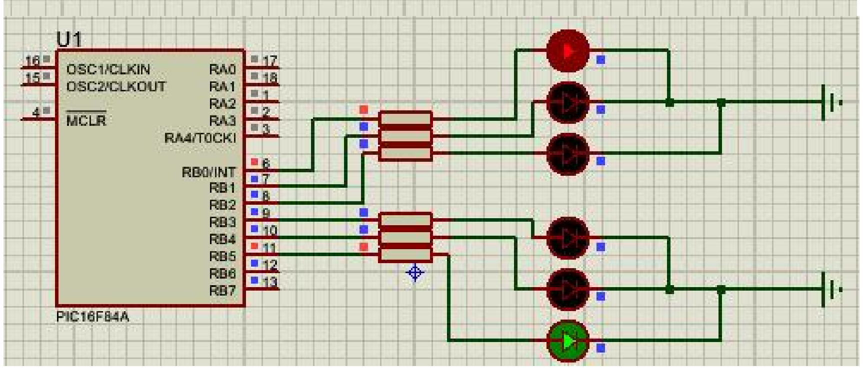
Circuit Design Description:

The circuit consists of a PIC16F84A microcontroller connected to 6 LEDs via its PORTA and PORTB pins. Each LED is connected in series with a 150Ω current-limiting resistor to prevent excessive current draw. The microcontroller outputs digital HIGH or LOW signals to control the LEDs. The power is supplied to the microcontroller via the VDD (Pin 14) and VSS (Pin 5) pins.


U1
16
15
4
OSC1/CLKIN
OSC2/CLKOUT
MCLR
RA0
RA1
RA2
RA3
RA4/T0CKI
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
17
18
1
2
3
6
7
8
9
10
11
12
13
PIC16F84A


U1
16
15
4
OSC1/CLKIN
OSC2/CLKOUT
MCLR
RA0
RA1
RA2
RA3
RA4/T0CKI
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
17
18
1
2
3
6
7
8
9
10
11
12
13
PIC16F84A


U1
16
15
4
OSC1/CLKIN
OSC2/CLKOUT
MCLR
RA0
RA1
RA2
RA3
RA4/T0CKI
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
17
18
1
2
3
6
7
8
9
10
11
12
13
PIC16F84A


U1
16
15
4
OSC1/CLKIN
OSC2/CLKOUT
MCLR
RA0
RA1
RA2
RA3
RA4/T0CKI
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
17
18
1
2
3
6
7
8
9
10
11
12
13
PIC16F84A

PROTEUS
SCHEMATIC DIAGRAM

Pin Mapping:

MCU Pin	Port	LED Connection
RB0	RB	LED1
RB1	RB	LED2
RB2	RB	LED3
RB3	RB	LED4
RB4	RB	LED5
RB5	RB	LED6

Code Summary:

I used a simple assembly or C code was written to continuously toggle the LEDs on and off. I then went on to compile the code to a HEX file and loaded into the PIC16F84A in Proteus.

```
// Example in C (using MPLAB or MikroC)

void main() {

   // Set PORTB as output
```

```
TRISB = 0x00;

// Clear PORTB (turn off all LEDs initially)
PORTB = 0;

// Infinite loop to continuously run LED pattern
while(1)
{
   // Step 1: Turn ON RB5 and RB0 (binary 00100001)
   PORTB = 0b00100001;
   delay_ms(5000);  // Wait for 5 seconds

   // Step 2: Turn ON RB4 and RB0 (binary 00010001)
   PORTB = 0b00010001;
   delay_ms(400);   // Wait for 400 ms

   // Step 3: Turn ON RB3 and RB0 (binary 00001001)
   PORTB = 0b00001001;
   delay_ms(300);   // Wait for 300 ms

   // Step 4: Turn ON RB3 and RB1 (binary 00001010)
   PORTB = 0b00001010;
   delay_ms(400);   // Wait for 400 ms

   // Step 5: Turn ON RB3 and RB2 (binary 00001100)
   PORTB = 0b00001100;
```

```
        delay_ms(5000);  // Wait for 5 seconds

        // Step 6: Back to RB3 and RB1 (00001010)
        PORTB = 0b00001010;
        delay_ms(400);   // Wait for 400 ms

        // Step 7: Back to RB3 and RB0 (00001001)
        PORTB = 0b00001001;
        delay_ms(300);   // Wait for 300 ms

        // Step 8: Back to RB4 and RB0 (00010001)
        PORTB = 0b00010001;
        delay_ms(400);   // Wait for 400 ms
    }
}
```

Actual program

```
void main() {
TRISB=0X00; PORTB=0;

    while(1)
    {

    PORTB=0B00100001;
    delay_ms(5000);
    PORTB=0B00010001; delay_ms(400);
    PORTB=0B00001001; delay_ms(300);

    PORTB=0B00001010; delay_ms(400);
    PORTB=0B00001100;
    delay_ms(5000);
    PORTB=0B00001010; delay_ms(400);
    PORTB=0B00001001; delay_ms(300);
    PORTB=0B00010001; delay_ms(400);

    }
}
```

Program Flow Summary:

- The code creates a **back-and-forth light movement effect** using PORTB.
- Some LEDs (e.g., RB0) stay ON longer across multiple patterns, acting as a reference or static signal.
- Timing varies between 300ms and 5s, giving both static display and motion effects.

Challenges and Fixes:

- **Issue:** LEDs were not lighting up initially.
  Fix: Resistor values were too high; reducing them to **150Ω** allowed sufficient current for LED illumination.
- **Lesson Learned:** Proper selection of resistor values is crucial in LED circuits to balance between brightness and current limitation.

Conclusion:

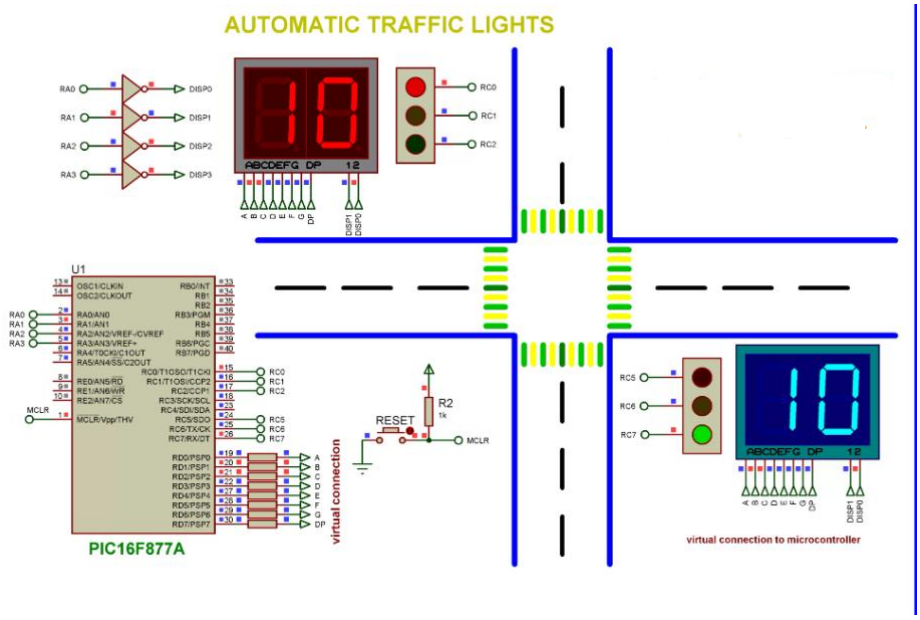
This project demonstrates a fundamental application of microcontroller-based digital output control using the PIC16F84A. Although this particular project utilized only PORTB, it laid a strong foundation for understanding microcontroller pin configuration, simulation testing, and basic embedded C programming.

Practical Applications of the LED Control System

This project demonstrates fundamental microcontroller-based control techniques that can be scaled for real-world applications. The basic principle of toggling outputs can be applied to many domains. Below are some practical applications:

1. Traffic Light and Pedestrian Crossing Simulators

Description:
Your timing and LED pattern logic simulate how traffic lights work—especially pedestrian signals where timing is crucial. The long delay (5 seconds) followed by quicker blinking patterns mimics pedestrian wait and walk phases.



---

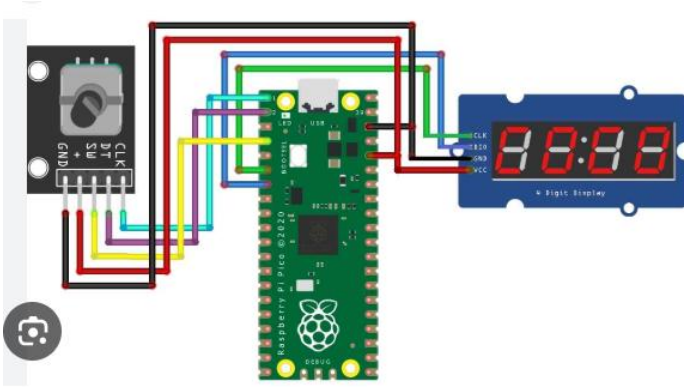
2. Industrial Indicator Panels

Description:
LEDs controlled by microcontrollers are widely used in manufacturing plants to indicate machine states (e.g., ON, OFF, ERROR).

---

3. Traffic Light Simulation and Control

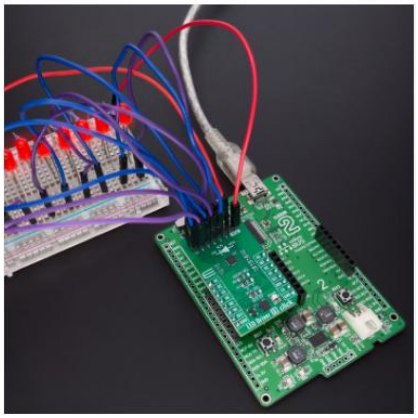
Description:
Basic LED sequencing mimics real-world traffic light control systems, which can be scaled using additional logic and sensors.



---

4. Educational Training Kits

Description:
This project can be a core module in educational kits that teach students how to interface microcontrollers with outputs.



---

5. Basic Security or Warning Systems