# TRAFFIC LIGHTS
# Report

**Name:** **احمد أسامة إبراهيم مصطفى النجار**

**Group: 1**

**A)** The PIC16F877A has a total of 40 pins, and they are divided into several groups based on their functionalities:

1- ***Port A (RA0 - RA5):*** General-purpose I/O pins. They can be configured as inputs or outputs depending on the application's requirements.

2- ***Port B (RB0 - RB7):*** General-purpose I/O pins. Like Port A, they can be used for both input and output operations.

3- ***MCLR/VPP (Pin 1):*** Master Clear Input/Voltage Programming Pin. This pin is used for external reset or in-circuit serial programming (ICSP).

4- ***VSS (Pin 20):*** Ground pin.

5- ***VDD (Pin 11):*** Power supply pin (+5V).

6- ***Oscillator Pins:*** These include OSC1/CLKIN (Pin 13) and OSC2/CLKOUT (Pin 14). They connect to an external oscillator or crystal to provide the clock signal for the microcontroller.

7- ***CCP Pins:*** CCP1 (Pin 15) and CCP2 (Pin 16) are Capture/Compare/PWM pins used for various timing and PWM (Pulse Width Modulation) applications.

8- ***USART Pins:*** TX (Pin 25) and RX (Pin 26) are used for serial communication using USART (Universal Synchronous Asynchronous Receiver Transmitter).

9- ***External Interrupt Pins:*** INT (Pin 33) and RB0/INT (Pin 32) are external interrupt pins that can trigger an interrupt when a specific event occurs.

10- ***A/D Converter Pins:*** AN0 - AN7 (Pins 2-9) are analog input pins for the onboard Analog-to-Digital Converter (ADC).

11- ***EEPROM Pins:*** These pins (VDD and VSS) are used for connecting an external EEPROM (Electrically Erasable Programmable Read-Only Memory).

12- ***PSP Pins:*** Parallel Slave Port (PSP) pins (PSP0 - PSP7) allow communication with external devices using a parallel interface.

## How to interface PIC16F877A with other hardware:

We need to identify the appropriate pins to connect with external components based on the desired functionalities. For example:

- To interface with a digital sensor or device, you would use the general-purpose I/O pins (Port A and Port B).
- For analog sensors or measurements, you can use the analog input pins (AN0 - AN7) and the onboard ADC.
- For communication with other devices, you can use the USART pins (TX and RX) or the CCP pins for PWM applications.
- If you need to use external memory, you can connect it to the appropriate pins.
- For interrupt-based applications, you can use the external interrupt pins (INT and RB0/INT).
- Make sure to configure the pins correctly in your code to ensure they operate in the desired input or output mode and to handle any interrupts or specific functionalities as required by your application. The PIC16F877A datasheet and reference manual will provide detailed information on how to interface the microcontroller with other hardware effectively.

## B) the functions of the main blocks in the PIC16F877A microcontroller:

### **ALU (Arithmetic Logic Unit):**

The ALU is a fundamental component of the central processing unit (CPU). Its primary purpose is to perform arithmetic (e.g., addition, subtraction) and logic (e.g., AND, OR, NOT) operations on binary data. The ALU takes data from the working registers, processes it based on the instructions fetched from memory, and generates the result as an output. It plays a crucial role in executing arithmetic and logical operations within the microcontroller.

### **Status and Control Registers:**

The Status and Control Registers are special registers within the PIC16F877A microcontroller that store important status flags and control bits. These registers include the STATUS register and the OPTION register, among others. The STATUS register contains various status bits, including the carry flag (C), zero flag (Z), and other flags related to arithmetic operations. The OPTION register holds control bits that configure various settings, such as the prescaler for timers and the external/interrupt edge select.

### **Program Counter (PC):**

The Program Counter is a 13-bit register that keeps track of the memory address of the next instruction to be fetched and executed. During normal program execution, the PC increments automatically after each instruction execution, pointing to the next instruction in memory. This allows the microcontroller to sequentially execute instructions stored in program memory.

## Flash Program Memory:

The Flash Program Memory is where the program instructions and data are stored. The PIC16F877A is a flash-based microcontroller, meaning its program memory can be electrically erased and reprogrammed. This allows developers to write, modify, and update programs onto the microcontroller. The program memory is non-volatile, meaning the program remains stored even when the power is turned off.

## Instruction Register (IR):

The Instruction Register is a temporary storage unit that holds the instruction fetched from program memory before execution. When an instruction is fetched, it is transferred to the Instruction Register. The microcontroller's circuitry then decodes the instruction to determine the corresponding operation to be performed.

**Instruction Decoder:** The Instruction Decoder is a crucial part of the microcontroller's CPU. Its primary function is to analyze the instruction fetched from program memory and translate it into specific signals that control the operation of various blocks within the microcontroller. The decoder identifies the type of instruction, the addressing modes, and the operands involved, enabling the correct execution of the instruction by the ALU or other relevant blocks.

## C) why doesn’t a LED connect to RA4 (Port A, Pin 4) of a microcontroller work correctly:

### Weak Internal Pull-up Resistor:

Some microcontrollers, including the PIC16F877A, have internal pull-up resistors that can be enabled for input pins. If the RA4 pin is accidentally configured as an input with the internal pull-up resistor enabled, it can prevent the LED from turning on. Make sure the RA4 pin is correctly configured as an output to drive the LED.

### Oscillator Configuration:

Microcontrollers like the PIC16F877A require an external oscillator or clock source for proper operation. If the oscillator configuration is not set correctly in the microcontroller's configuration bits, the internal clock might not be running at the expected frequency or not running at all. As a result, the LED flashing may not occur as intended. Verify that the oscillator settings are appropriate for your application.

### Brownout Reset:

Some microcontrollers have a brownout reset feature that prevents the chip from operating when the supply voltage drops below a certain threshold. If the supply voltage is too low, the microcontroller might not function correctly, and the LED won't flash. Ensure that the power supply voltage meets the microcontroller's minimum voltage requirements.

## Watchdog Timer:

If the microcontroller's watchdog timer is enabled and not being serviced correctly in the code, it can lead to a reset of the microcontroller, affecting the LED operation. Make sure to either disable the watchdog timer if not needed or service it properly in the code to prevent unintentional resets.

## Code-related Issues:

Review the code that controls the LED flashing. Check for errors in the timing and logic of the flashing routine. Ensure that the LED control code is not overwritten or interfered with by other parts of the program.

## Electrical Noise:

In some cases, electrical noise or interference from other components or signals might affect the proper functioning of the microcontroller's I/O pins. Consider using proper decoupling capacitors and minimizing noise sources in the circuit.

**<u>Incorrect Fuse Settings (for AVR microcontrollers):</u>**

For AVR microcontrollers like the ATMega328P, fuse settings control various aspects of the chip's behavior, including clock source, startup time, etc. Incorrect fuse settings can result in unexpected behavior or prevent the microcontroller from operating correctly.

Double-checking the hardware connections, code, and configuration settings should help identify the root cause of the LED not working correctly with the RA4 pin. Debugging and troubleshooting step-by-step will lead to a successful solution.

## D) let's compare the characteristics of the ATMega328P and PIC16F877A microcontrollers:

**<u>Memory Size:</u>**

***<u>ATMega328P:</u>*** The ATMega328P has 32KB of flash program memory and 2KB of SRAM.

***<u>PIC16F877A:</u>*** The PIC16F877A has 14KB of flash program memory and 368 bytes of RAM.

**<u>Power Consumption:</u>**

***ATMega328P:*** The ATMega328P is known for its low power consumption and features various sleep modes, making it suitable for power-sensitive applications.

***PIC16F877A:*** While the PIC16F877A is reasonably power-efficient, it may not offer the same range of low-power modes as the ATMega328P.

## Pin Count:

***ATMega328P:*** The ATMega328P comes in various packages, with 28 or 32 pins (DIP, TQFP, QFN, etc.).

***PIC16F877A:*** The PIC16F877A has 40 pins (DIP or QFP package).

Examples of embedded systems where ATMega328P might be a better choice than PIC16F877A:

## Battery-Powered IoT Devices:

The low power consumption of the ATMega328P makes it a great choice for battery-powered Internet of Things (IoT) devices. These devices often operate on limited power sources, and the ATMega328P's ability to enter various low-power modes can significantly extend battery life.

## <u>Wearable Devices:</u>

Wearable devices, such as fitness trackers or smartwatches, require a microcontroller that balances performance with power efficiency. The ATMega328P's lower power consumption and smaller form factor (with its various package options) can be advantageous in such applications, especially when space and battery life are critical factors.