Ministry of Higher Education and Scientific Research
Badji Mokhtar Annaba University
Faculty of technology
Departement of electronics

Practical Work Mp and Mc

PW n° 1

PIC16F84 Microcontroller Programming Environment

These practical sessions were developed for the Microprocessors and Microcontrollers lab module of L3 Automation, for the 2025/2026 academic year within the Department of Electronics Badji Mokhtar ANNABA University.

By Dr. MERABTI Nardjes

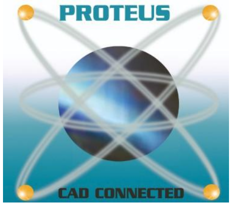
Objective

The main objective of this first lab session is to become familiar with the tools used to carry out the labs for this module. We focus on programming the Microchip PIC microcontroller (PIC16F84) and on the design and simulation environments (MPLAB and Proteus), and therefore on using the basic functions of the MPLAB development tool to:

- Create a project and edit a program
- Compile, debug, and simulate a program with MPLAB, and perform simulations using PROTEUS
- **Resources**
  PC equipped with MPLAB and Proteus software
- **MPLAB software from Microchip:**

MPLAB is a free programming environment for Microchip PIC microcontrollers. It includes an assembler editor, a full debugger, and all tools needed to manage device programmers. This integrated development environment (IDE) is essential for writing, compiling, and debugging code for the PIC16F84.

- **Proteus :**



Proteus is an excellent electronic simulation software. It is a schematic editor that includes analog, digital, and mixed-mode simulation. It can simulate and verify the behavior of the PIC16F84. The simulator allows you to check your circuit's functionality before physically building it, speeding up prototyping and reducing development costs.

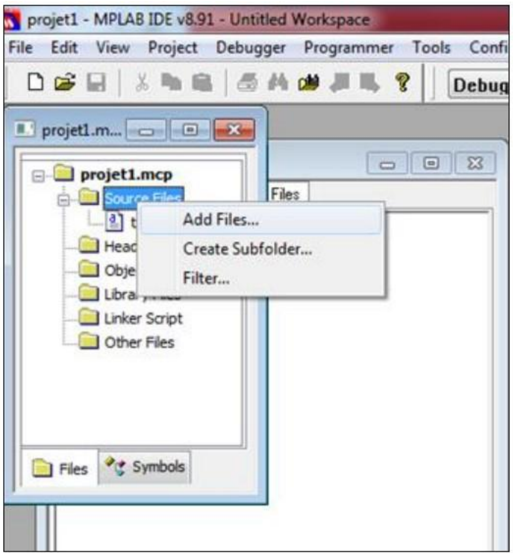
First Steps: Software Installation:

- ✓ **MPLAB IDE:** Download a version from the Microchip website and follow the installation instructions.
- ✓ **Proteus:** Obtain the student or professional version, depending on your needs. The installation process is generally straightforward.

● Exemple 1:

Creating a New Project with “Project Wizard”

- ➢ Click **Project >> Project Wizard…**
- ➢ Click **Next**, then select **PIC16F84A** from the **Device** dropdown menu.
- ➢ Click **Next >**, then select **Microchip MPASM Toolsuite** and **MPASM Assembler** to program in Assembly language.
- ➢ Click **Next >** enter the project name in the **Project Name** field, click **Next >**, then click **Finish**.



Enter the following program and save it in your directory:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
T1 EQU 0x30
MOVLW 0XFF
ADDLW B'01101010'
MOVWF T1
END
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

- ➢ **Compilation**

- Click **Project >> Build All...**, the menu icon, or press **Ctrl+F10** to compile the entire project.

  Errors and warnings appear in the **Output** window, along with the compilation result: **BUILD SUCCEEDED** or **BUILD FAILED**.

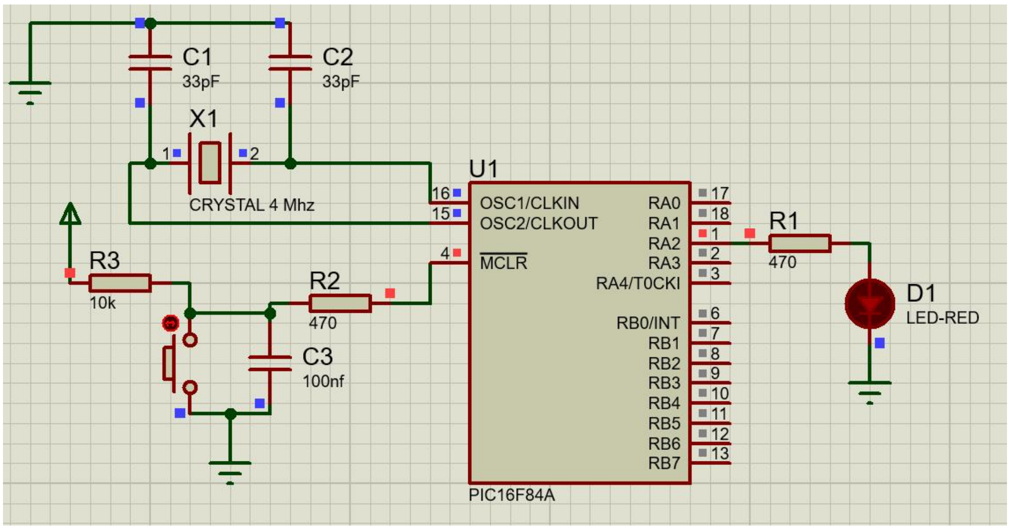
  Note: Double-clicking an error or warning in the Output window takes you directly to the relevant line in your code. If there are no compilation errors, you will see **BUILD SUCCEEDED** and the file TP.hex will be generated.

- **Simulation**

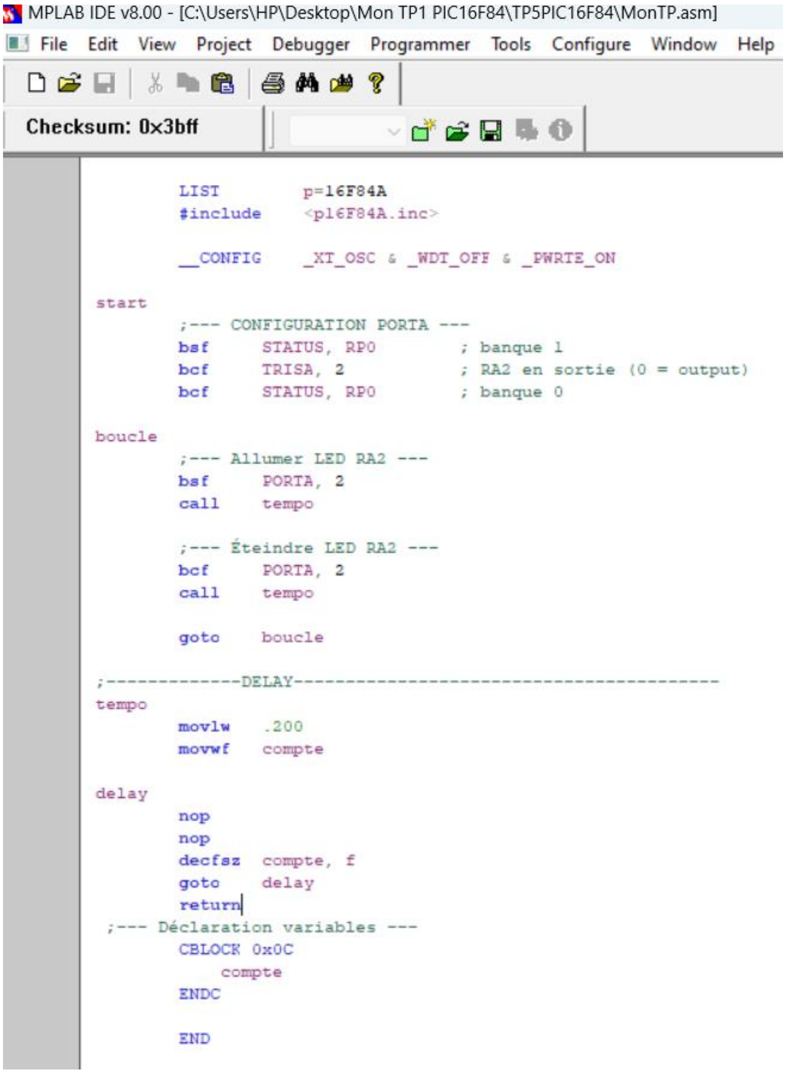
Using MPLAB SIM as the Debugger:

- ✓ Select **MPLAB SIM** as the debugger.
- ✓ Execute the program step by step and observe the evolution of registers such **STATUS** (**View → Special Function Registers**).
- ✓ Track registers and variables (**View → Watch**). Add a **Hex** column via the right-click menu on the column headers to see register values in hexadecimal

1. Creating a New MPLAB Project to Blink an LED:



Open **MPLAB X IDE;** Create a new project and select **PIC16F84** as the microcontroller; Write a simple program:


MPLAB IDE v8.00 - [C:\Users\HP\Desktop\Mon TP1 PIC16F84\TP5PIC16F84\MonTP.asm]
File Edit View Project Debugger Programmer Tools Configure Window Help
Checksum: 0x3bff

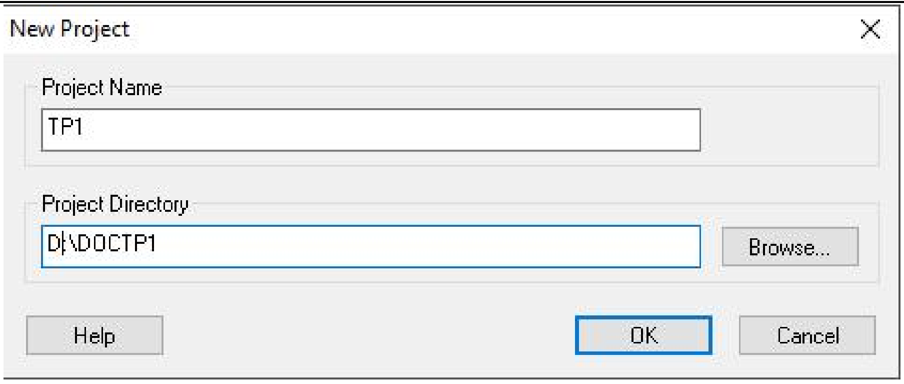
```
        LIST        p=16F84A
        #include    <p16F84A.inc>

        __CONFIG    _XT_OSC & _WDT_OFF & _PWRTE_ON

start
        ;--- CONFIGURATION PORTA ---
        bsf     STATUS, RP0         ; banque 1
        bcf     TRISA, 2            ; RA2 en sortie (0 = output)
        bcf     STATUS, RP0         ; banque 0

boucle
        ;--- Allumer LED RA2 ---
        bsf     PORTA, 2
        call    tempo

        ;--- Éteindre LED RA2 ---
        bcf     PORTA, 2
        call    tempo

        goto    boucle

;-------------DELAY----------------------------------------
tempo
        movlw   .200
        movwf   compte

delay
        nop
        nop
        decfsz  compte, f
        goto    delay
        return
 ;--- Déclaration variables ---
        CBLOCK 0x0C
            compte
        ENDC

        END
```

- **Simple Application Example:**
  To create a simple LED blinker, a program is written in Assembly language to turn a LED on and off, connected to a PIC pin, at regular intervals.

- **Simulation in Proteus:**

- Create a new schematic in **Proteus**.
- Place the necessary components: PIC16F84, resistors, LED, etc.
- Connect the components according to your circuit diagram.
- Import the compiled code from MPLAB into Proteus.
- Simulate the circuit and observe the LED behavior.

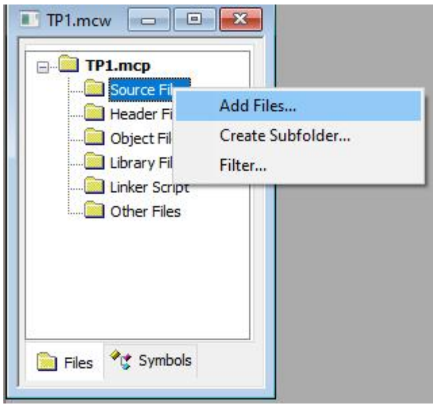
Click **Project → New**



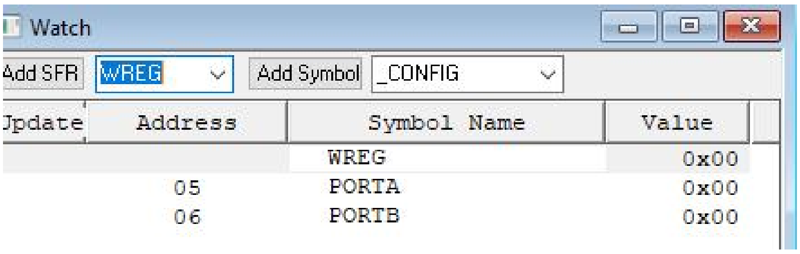
Project Name: TP1

Project Directory: D:\DOCTP1

Right-click and add the file **Programme1.asm** to the **TP1** project.

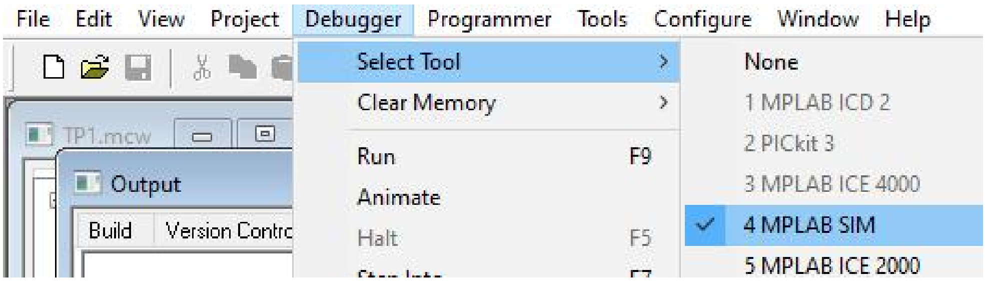


- ➢ Click **Configure → Select Device** and select **PIC16F84**.
- ➢ Compile the project: **Project → Build All**.
- ➢ Verify the creation of the new file **programme.hex**. Open this file with a text editor like **Notepad** and draw conclusions.
- ➢ View the registers via **View → Watch**.

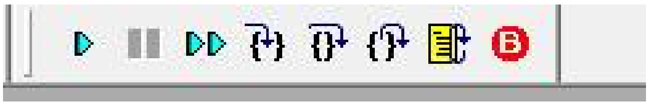


Jpdate	Address	Symbol Name	Value
		WREG	0x00
	05	PORTA	0x00
	06	PORTB	0x00

Select MPLAB SIM



Using step-by-step execution, check the register values in the **Watch** window.



In Proteus, open the project ; Load **programme.hex** into the PIC; Verify the program's operation by starting the simulation.

Part 2:

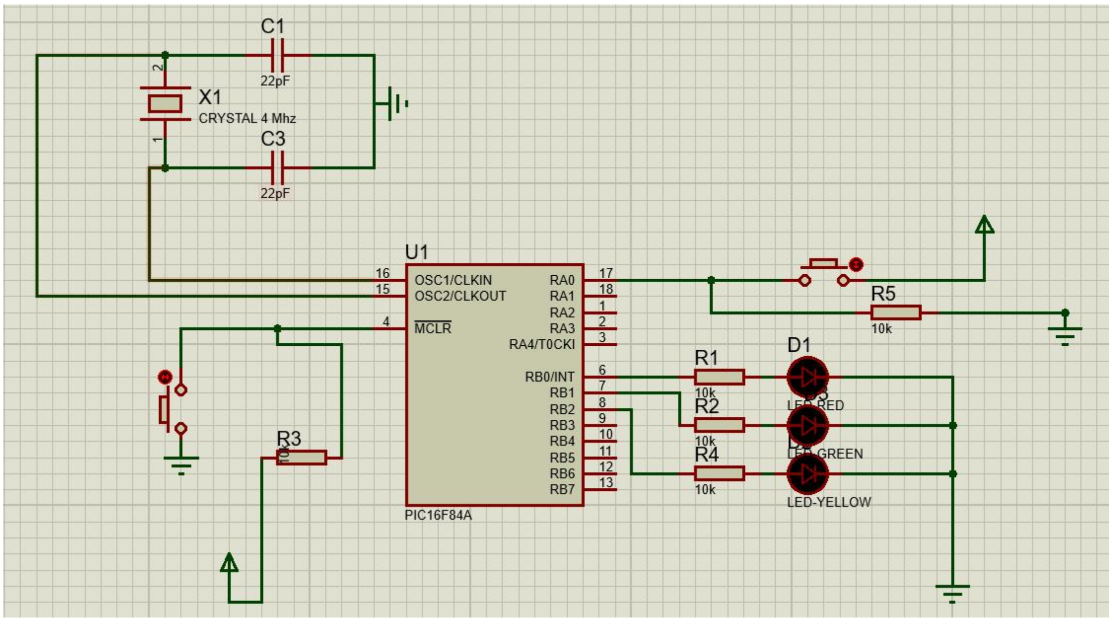
Work to do:

Create the following schematic

Create a traffic light with three LEDs: red, yellow, and green

A traffic light system can be built using three LEDs connected to three output pins of the PIC16F84A:

- **Red LED**: Stop
- **Yellow LED**: Prepare
- **Green LED**: Go



The program must:

- ✓ Turn ON the **red LED** (others OFF) for a fixed time
- ✓ Turn ON the **green LED** for another time
- ✓ Turn ON the **yellow LED** for a short warning time
- ✓ Repeat the sequence continuously