UNIVERSITE HASSIBA BENBOUALI DE CHLEF
Faculté de Technologie
Département d'Electronique

MEMOIRE DE MASTER

Domaine : SCIENCES ET TECHNOLOGIES

Filière : : Electronique

Spécialité : Electronique des systèmes embarqués

Etude et réalisation d'un affichage lumineux à base de pic 16F877

Par

Farouk Douba

Ilyes Fellague

Encadreur :

M.Boualem Soumia

Maitre de conference classe-B-

Chlef, Juin 2025

Remerciements

On tient à remercier tout d'abord notre Dieu qui nous a donné le courage et la volonté et qui nous a aidé et montré le chemin du savoir.

Nos plus sincères remerciements à notre encadrante **Dr Soumia Boualem** pour sa patience, sa disponibilité, ses efforts, ses précieux conseils qui nous ont été très utiles et ses critiques objectives sur la démarche de notre travail. Permettez-nous Madame de vous exprimer notre reconnaissance et notre respect.

Et je remercié aussi le chef de département et tous les enseignements de la filière de génie électrique. Enfin, nous tenons à exprimer nos remerciements les plus sincères à toute notre famille et nos amis qui nous ont toujours soutenus.

Sommaire

Chapitre I : Généralités sur les Pic 16F877

Chapitre II : Les techniques d'affichage

Chapitre III : Les logiciels utilisé

Chapitre IV : Réalisation et simulation

Liste des figures

Chapitre I :

Chapitre II :

Chapitre III :

Chapitre IV :

Liste des Tableaux

Chapitre I :

Chapitre II :

Chapitre III :

Chapitre IV :

Résumé

L’ objectif de notre travail consiste a un système d'affichage intelligent basé sur le microcontrôleur PIC16F877, combinant une matrice LED 8x8 pour des motifs dynamiques et un écran LCD 16x2 pour l'affichage textuel. Le PIC gère simultanément le multiplexage de la matrice LED et la communication avec l'afficheur LCD. La matrice permet des animations et messages défilants grâce à un algorithme de balayage rapide, tandis que le LCD affiche des informations statiques. L'ensemble est programmé en C sous MikroC et simulé avec Proteus avant implémentation. Ce système polyvalent trouve des applications dans l'affichage publicitaire, les panneaux d'information et les dispositifs interactifs.

ملخص

الهدف من عملنا هو نظام عرض ذكي يعتمد على متحكم **PIC16F877**، يجمع بين مصفوفة **LED 8x8** للأنماط الديناميكية وشاشة **LCD 16x2** لعرض النص. تتولى وحدة التحكم **PIC** إدارة إرسال مصفوفة **LED** والتواصل مع شاشة **LCD** في نفس الوقت. تتيح المصفوفة إمكانية عرض الرسوم المتحركة والرسائل المتحركة بفضل خوارزمية المسح السريع، في حين تعرض شاشة **LCD** معلومات ثابتة. تم برمجة كل شيء في **C** تحت **MikroC** وتم محاكاته باستخدام **Proteus** قبل التنفيذ. يجد هذا النظام متعدد الاستخدامات تطبيقات في شاشات الإعلانات ولوحات المعلومات والأجهزة التفاعلية.

Summary

The objective of our work is a smart display system based on the PIC16F877 microcontroller, combining an 8x8 LED matrix for dynamic patterns and a 16x2 LCD screen for textual display. The PIC simultaneously manages the multiplexing of the LED matrix and the communication with the LCD display. The matrix allows animations and scrolling messages thanks to a fast scanning algorithm, while the LCD displays static information. The whole system is programmed in C under MikroC and simulated with Proteus before implementation. This versatile system finds applications in advertising displays, information panels and interactive devices.

Introduction Générale

Les microcontrôleurs jouent un rôle essentiel dans les systèmes embarqués modernes, offrant une solution compacte, performante et économique pour le contrôle et l'automatisation de divers dispositifs électroniques. Parmi eux, le PIC16F877 se distingue par sa polyvalence, sa facilité d'utilisation et ses fonctionnalités avancées, ce qui en fait un choix privilégié pour de nombreuses applications industrielles et éducatives.

Dans cette section, nous explore en détail le microcontrôleur PIC16F877, en mettant l'accent sur son architecture, ses caractéristiques techniques et ses applications pratiques. Nous abordons également les différentes méthodes d'affichage couramment utilisées avec ce microcontrôleur, telles que les afficheurs 7 segments, les matrices LED et les écrans LCD, en soulignant leurs principes de fonctionnement, leurs avantages et leurs limites.

De plus, nous présentons les outils logiciels indispensables pour le développement et la simulation de projets basés sur le PIC16F877, notamment MikroC pour la programmation et Proteus pour la simulation. Ces outils permettent de concevoir, tester et optimiser des systèmes embarqués de manière efficace avant leur implémentation matérielle.

Enfin, ce chapitre sert de fondement pour les applications pratiques qui seront développées dans les chapitres suivants, où nous mettrons en œuvre ces concepts pour réaliser des projets concrets, tels que l'affichage dynamique de messages ou la régulation de paramètres physiques. À travers cette étude, nous visons à démontrer l'importance des microcontrôleurs dans les systèmes automatisés et à fournir les bases nécessaires pour leur maîtrise.

Chapitre I : généralités sur Les PIC16F877

I.1. Introduction :

Chaque dispositif, jouet électrique, mobile, climatiseur ou ordinateur a une certaine carte électronique qui comprend généralement un dispositif programmable dit microcontrôleur, il s'agit d'un microprocesseur spécial avec des périphériques, des ports d'entrée / sortie selon le volume et la qualité de l'appareil dont il est utilisé.

Plus de deux milliards de microcontrôleurs sont produits chaque année. Presque tout le monde les utilise d'une manière quotidienne, aussi bien dans les pays développés que dans les pays en voie de développement ne peuvent penser à une journée sans utiliser des microcontrôleurs.

De nos jours, les microcontrôleurs sont devenus des composants électroniques clé pour tous systèmes qui sont moins coûteux et facilement disponibles pour les amateurs, avec de grandes communautés en ligne autour de certains processeurs

Généralités sur les microcontrôleurs

Un microprocesseur est un composant électronique qui traite une information d'une manière séquentielle et logique .Il est constitué d'une UCT (unité centrale de traitement) qui contient une UAL et des registres. En ajoutant une mémoire centrale et des périphériques (compteur, convertisseurs analogiques/numériques les timers, les ports E/S parallèles ou séries, bus de données et d'adresses). Tout c'est composant sont intégré dans un seul circuit nommé microcontrôleur automatique [1].

I.2. Microcontrôleur :

I.2.1. Qu'est-ce qu'un microcontrôleur ?

Un microcontrôleur est un circuit intégré rassemblant dans un même boitier un microprocesseur, plusieurs types de mémoires et des périphériques de communication (Entrées-Sorties) **[2]**. Il est donc composé en plus de l'unité centrale de traitement, d'une mémoire (mémoire vive RAM et mémoire morte ROM), une (ou plusieurs) interface de communication avec l'extérieur matérialisé par les ports d'entrée/sortie.

En plus de cette configuration minimale, les microcontrôleurs sont dotés d'autres circuits d'interface qui vont dépendre du microcontrôleur choisi à savoir les systèmes de comptage

(TIMER), les convertisseur analogique/numérique (CAN) intégré, gestion d'une liaison série ou parallèle, un Watchdog (surveillance du programme), une sortie PWM (modulation d'impulsion)…**[3]**

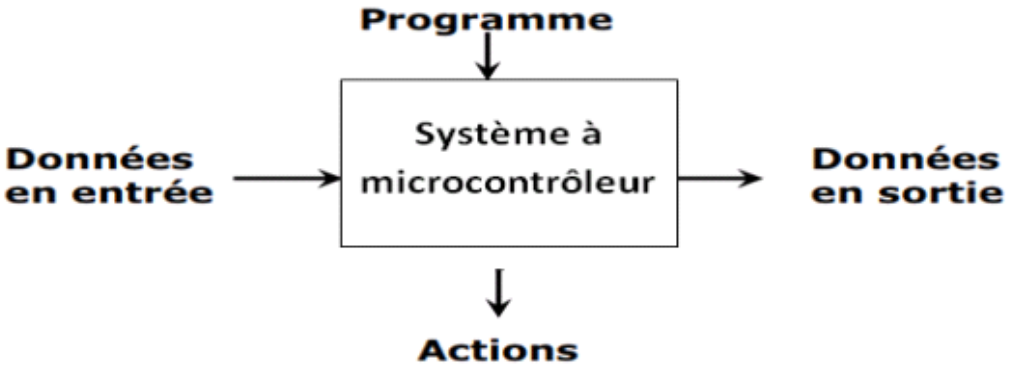


Figure I.1 : Schéma d'un système à base d'un microcontrôleur **[1]**

I.2.2. Histoire des Microcontrôleurs :

Le premier microprocesseur, l'Intel 4004 (4 bits), a été commercialisé en 1972, suivi de l'Intel 8008 et d'autres modèles plus performants. Cependant, ces processeurs nécessitaient des composants externes, rendant leur utilisation coûteuse.

En 1971, Gary Boone et Michael Cochran (Texas Instruments) ont créé le premier microcontrôleur, le TMS 1000, disponible en 1974. Il intégrait mémoire ROM/RAM, processeur et horloge sur une seule puce, facilitant les systèmes embarqués. Pour concurrencer le TMS 1000, Intel a conçu en 1977 le 8048, combinant RAM et ROM sur la même puce. Ce microcontrôleur a connu un immense succès, notamment dans les claviers de PC.

Figure I. 2 : TMS 1000 4-Bit microcontrôleurs, TI, 1976 **[4]**

Dans les années suivantes, différents types de mémoire ont été utilisés : EPROM (effaçable aux UV), PROM OTP (programmable une seule fois) et EPROM effaçable, plus coûteuse à cause de son boîtier en céramique.

En 1993, l'arrivée de la mémoire EEPROM a permis une réécriture rapide sans boîtier coûteux. Atmel a lancé le premier microcontrôleur à mémoire flash, facilitant la programmation et la production en série.

Aujourd'hui, les microcontrôleurs sont abordables et accessibles, avec de grandes communautés en ligne soutenant leur développement [5].

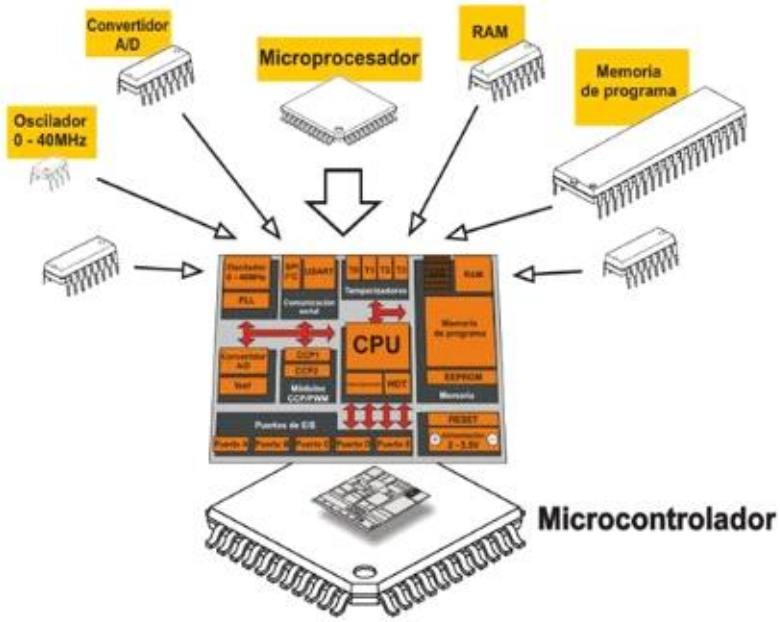


Figure I.3 : Contenu type d'un microcontrôleur **[6]**

Il existe plusieurs familles de microcontrôleurs dont les plus connues sont : **Atmel :** AT : familles AT89Sxxxx, AT90xxxx, …

Motorolla : famille 68HCxxx, …

Microship : PIC : familles 12Cxxx, 16Cxxx, 16Fxxx, 18Fxxx, …

Intel : famille 80C186XX.

STMicroelectronics : famille STX.

Analog Devices : famille ADuC .

Nous allons nous intéresser dans le cadre à la famille Microchip **PIC** (**P**rogrammable **I**ntegrated **C**ircuit) **[3].**

I.2.3. Choix d'un Microcontrôleur :

Il existe plusieurs microcontrôleurs fabriqués par INTEL, MOTOROLA, HITACHI, NEC TEXAS instrument, MICROCHIP etc.… Le choix d'un microcontrôleur dépend de plusieurs critères de sélection dont le développeur doit tenir compte comme :

- Nombre d'entrées/sorties.
- Liaison d'entrées/sorties.
- Conversion analogique numérique et numérique analogique.
- Entrées/sorties rapides, sorties spéciales (M.L.I, horodaté etc…).
- Mémoire RAM, ROM, EPROM interne ou externe, sa taille.
- Vitesse de l'horloge, temps d'exécution d'une multiplication, d'une division.
- Bus de données 8bits /16bits.
- Les logiciels de programmation (assembleur, c, micro c etc…).
- Les émulateurs pour la mise au point des applications.
- Les évolutions prévisibles du composant, son prix, les sources **[7].**

I.2.4. Les avantage des microcontrôleurs :

L'utilisation des microcontrôleurs pour les circuits programmables à plusieurs points forts et réels. Il suffit pour s'en persuader, d'examiner la spectaculaire évolution de l'offre des fabricants de circuits intégrés en ce domaine depuis quelques années.

- Tout d'abord, un microcontrôleur intégré dans un seul et même boitier nécessite une dizaine d'éléments séparés. Il résulte donc une diminution évidente de l'encombrement de matériel et de circuit imprimé.

- Cette intégration a aussi comme conséquence immédiate de simplifier le tracé du circuit imprimé puisqu'il n'est plus nécessaire de véhiculer des bus d'adresse et de donnée d'un composant à un autre.
- Le microcontrôleur contribue à réduire les couts à plusieurs niveaux :
  - Moins cher que les autres composants qu'il remplace.
  - Diminuer les couts de main d'œuvre.
  - Réalisation des applications non réalisables avec d'autres composants.
  - Investissement dans les outils de développement [**7].**

I.3. Définition d'un PIC :

Un PIC (Programmable Interface Controller) est un type de microprocesseur doté de périphériques intégrés, ce qui simplifie sa connexion avec d'autres dispositifs sans nécessiter de composants supplémentaires.

Les PICs appartiennent à la famille des processeurs RISC (Reduced Instruction Set Computer), c'est-à-dire qu'ils utilisent un jeu d'instructions simplifié pour améliorer leur efficacité. Ces microcontrôleurs sont présents dans de nombreux appareils du quotidien tels que les téléphones portables, machines à laver, téléviseurs et lecteurs vidéo, facilitant ainsi leur automatisation et leur fonctionnement. [3]

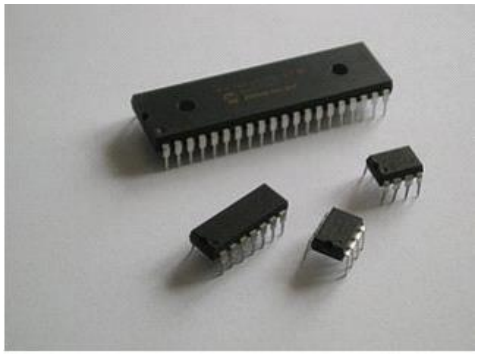


Figure I.4 : Les PICs microcontrôleurs **[8]**

I.4. Les différentes familles de PIC :

- La famille **Base Line**, qui utilise des mots d'instructions de 12 bits.
- La famille **Mid-Range**, qui utilise des mots de 14 bits (et dont font partie la 16F84 et 16F876).
- La famille **High-End**, qui utilise des mots de 16 bits.

Tous les microcontrôleurs PIC **Mid-Range**, y compris le **PIC 16F877,** disposent d'un ensemble de **35** instructions. Chaque instruction est enregistrée dans un seul mot mémoire et s'exécute en un seul cycle d'horloge, à l'exception des instructions de saut. Cette conception permet d'atteindre des vitesses élevées, tout en assurant une exécution rapide et efficace des commandes. **[9]**

I.5. Identification des PIC :

Pour identifier un microcontrôleur PIC, on utilise simplement son numéro. Les 2 premiers chiffres indiquent la catégorie du microcontrôleur PIC, **16** indique une **PIC Mid-Range**. Vient ensuite parfois une lettre :

L : Celle-ci indique que le microcontrôleur PIC peut fonctionner avec une plage de tension plus tolérante.

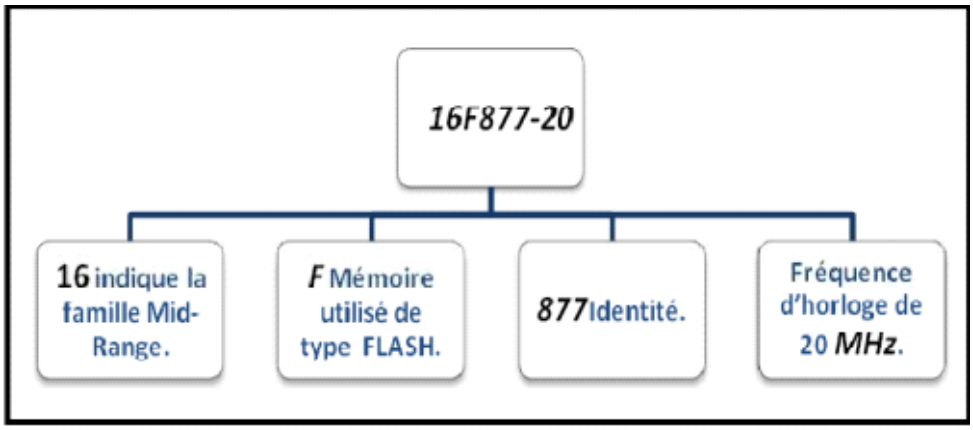
C : indique que la mémoire programme est une **EPROM** ou plus rarement une **EEPROM**.

CR : pour indiquer une mémoire de type **ROM**.

F : pour indiquer une mémoire de type **FLASH**.

Exemple

Un 16F877-20 est un **PIC Mid-range (16)** ou la mémoire programme est de Type FLASH (F) et réinscriptible de type 877 et capable d'accepter une fréquence d'horloge de 20 MHz (figure I.5) **[10].**



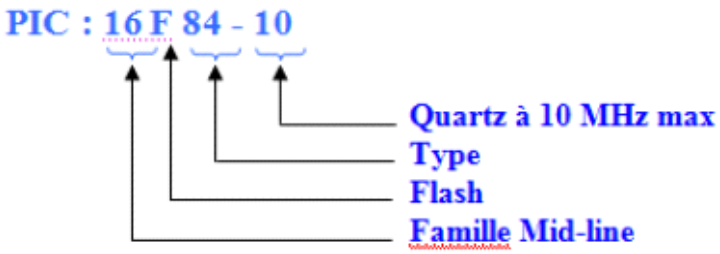


Figure I.5 : Exemple de l'indentification pic **[3][10]**

Les microcontrôleurs PICS sont des composants STATIQUES, c'est à dire que la fréquence d'horloge peut être abaissée jusqu'à l'arrêt complet sans perte de données et sans dysfonctionnement. Ceci par opposition aux composants DYNAMIQUES, donc la fréquence d'horloge doit rester dans des limites précises **[10].**

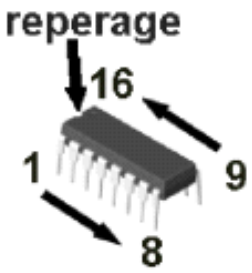


Figure I. 6 : Méthode des lecteurs et Repérage de broches **[3]**

Les microcontrôleurs PIC sont présentés en boîtier DIL (Dual In Line). Un point ou une encoche donne un repérage de la broche 1, ensuite il faut se déplacer vers la droite pour avoir les autres broches. Ont fait le tour du circuit dans le sens trigonométrique.

I.6. Les choix d'un PIC :

Le choix d'un PIC est directement lié à l'application envisagée :

Il faut dans un premier temps déterminer le nombre d'entrées/sorties nécessaires pour l'application. Ce nombre d'entrées/sorties nous donne une idée sur la famille du PIC.

Il faut ensuite déterminer si l'application nécessite un convertisseur Analogique/ Numérique ce qui va centrer un peu plus vers le choix du PIC.

La rapidité d'exécution est un élément important, il faut consulter les DATA-BOOK pour vérifier la compatibilité entre la vitesse maximale du PIC choisi et la vitesse max nécessaire au montage.

La taille de la RAM interne et la présence ou nom d'une EEPROM pour mémoriser des données est également important pour l'application souhaitée.

La longueur de programme de l'application détermine la taille de la mémoire programme du PIC recherché.

Afin de choisir un PIC adéquat à notre projet, nous avons pensé à l'utilisation du PIC16F877 **[11].**

I.7. Les caractéristiques d'un PIC 16F877 :

Nous avons utilisé le microcontrôleur PIC 16F877A pour contrôler l'affichage lumineux, ce microcontrôleur c'est un circuit intégré à 40 broches dont les caractéristiques générales peuvent être résumé comme suit : Vitesse d'exécution allant jusqu'à 20MHz permettant l'exécution rapide du programme, une instruction en cycle d'horloge de 200 ns. Une mémoire vive « RAM » de 368 octets, répartie sur 4 banques 0, 1, 2, 3 « Mémoire de données ». Une mémoire EEPROM pour

sauver des paramètres de 256 octets « mémoire de données ». Une mémoire morte « ROM » mémoire programme de type FLASH de 8 ko « 1mot=14bit ».

- 33 Entrées/Sorties programmables multiplexes.
- 8 entrées « canaux » du module de conversion analogique numérique 10 bits.
- 2 Comparateurs analogiques programmables avec une référence de tension, programmable également.
- Conservation des informations en mémoire vive jusqu'à 1.5V.
- Une plage d'alimentation de 2.0V à 5V.
- Module de capture PWM programmable permettant de capturer un signal « Logique » en fonction du temps, ou d'offrir un signal PWM : Module de Largeur d'Impulsion.
- De nouvelles fonctionnalités, comme les gestions de ports « Série ».

Le PIC 16F877 parmi les plus puissants microcontrôleurs de la gamme « Mid- range » chez Micro-chip**[12].**

I.8 Architecture d'un PIC 16F877 :

I.8.1 Architecture extrene :

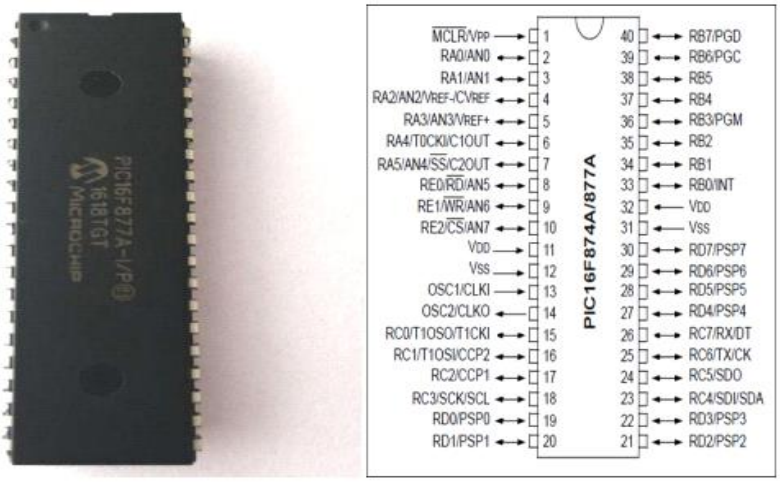


Figure I.7 : boitier du PIC 16F877**[13]**

Le microcontrôleur PIC 16F877 fait partie de la sous famille des 16F87X. Cette branche fait partie intégrant de la famille des Pics Mid-Range, Le pic 16f877 se présente sous la forme d'un boitier PDIP 40 broches, le« XX » qui Donne la fréquence d'horloge maximum du composant. A l'heure actuelle, la version la plus courante est la version 20. Donc la fréquence maximale sera de 20MHz [13]. Par exemple: un PIC 16F877-20 signifie que sa fréquence maximale est 20 MHz.

PIN	Utilisation
RA0 à RA5	Lignes d'entrées/sorties du port A
RB0 à RB7	Lignes d'entrées/sorties du port B
RC0 à RC7	Lignes d'entrées/sorties du port C
RD0 à RD7	Lignes d'entrées/sorties du port D
RE0 à RE3	Lignes d'entrées/sorties du port E
VDD	Broche d'alimentation +5V
VSS	Broche du 0V
MCLR	Reset du circuit
OSC1 et OSC2	Broch recevant quartz externe

Tableau I.1: Broches du PIC 16F877**[13]**

Certaine broche possède plusieurs fonctions par exemple la broche 25 notée RC6/TX/CK :

RC6 : entrée ou sortie bit 6 du port C

TX : Transmission de l'interface série de programmation

CX : horloge (Clock) de l'interface série

Les ports sont les liens physiques de votre programme avec l'environnement extérieur (capteur, afficheur, moteur… etc.). Les ports sont connectés à la circuiterie interne du PIC par l'intermédiaire d'un bus système de 8 bits **[7].**

I.8.1.1 Les broches d'alimentation :

Le 16F877 a des broches d'alimentation : 2 pour le 0V (broches 11 et 32) et 2 pour le +5V (broches 12 et 31). Il suffit de connecter une de chaque à l'alimentation pour que le PIC fonctionne.

Les courants circulés dans le PIC sont loin d'être négligeables du fait des nombreuses entrées/sorties disponibles (Figure 10) **[7].**

I.8.1.2 Les broches de quartz :

La synchronisation du microcontrôleur doit être présente pour le pilotage de ce dernier, elle se faite à l'aide d'un quartz varié de 4 à 20 Mhz, mis avec deux condensateurs de filtrage **[7].**

Les pattes concernées par le cortège du pilotage à l'aide d'un 20 Mhz quartz sont présentes dans le PIC aux numéros 13 et 14 (Figure 10).

Figure I.8 : Le Quartz Crystal **[16]**

I.8.1.3 Circuit d'horloge :

Un signal d'horloge est nécessaire pour piloter le circuit de contrôle et gérer les séquences du microcontrôleur. La fréquence de ce signal subit une division par quatre, fournissant ainsi un signal d'horloge interne, c'est cette base de temps qui est utilisée pour donner le temps d'un cycle

de T est donné par : 4/Fosc. Dans cette application, nous avons utilisé un quartz de 20 MHZ, nous aurons donc T=0.2µs, ce qui donne 5 millions cycles par seconde.

Le PIC16F877 exécute pratiquement (sauf les sauts) une instruction par le cycle, ce qui nous donne une puissance de l'ordre de 5MIPS (5 Million d'instructions par seconde). Comme la montre la figure I.9, Le quartz est branché sur les broches OSC1/CLKIN et OSC2/CLKOUT, avec de capacité de 15pF recommander pour la stabilité de l'oscillateur.

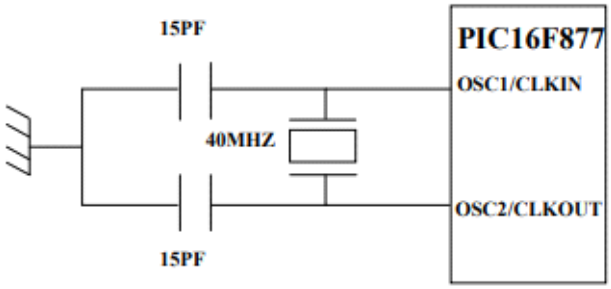


Figure I.9 : Circuit d'horloge du PIC 16F877 **[14]**

I.8.1.4 Circuit d'initialisation :

Le circuit de reset externe est nécessaire lors de la mise sous tension, en effet, à ce moment-là, le microcontrôleur n'est pas prêt à exécuter des instructions instantanément. L'entrée MCLR (Broche1) permet d'initialiser le microcontrôleur après la mise sous tension. Une réinitialisation est correctement effectuée si l'entrée MCLR est mise au niveau bas. Le circuit RC empêche la tension sur la broche 1 d'attendre la valeur de la tension d'alimentation pendant un temps correspondant au temps de charge de la capacité, afin que le circuit interne de reset automatique détecte la tension d'alimentation. Soit le temps de charge τ=1ms.

Le bouton poussoir permet le reset du pic pendant son fonctionnement. En fixant la valeur de la capacité C à 0.1µF (filtrer les rebonds du bouton poussoir, **éliminer les bruits parasites**, et **stabiliser le signal** envoyé au PIC 16F877), on aura R=10KΩ.

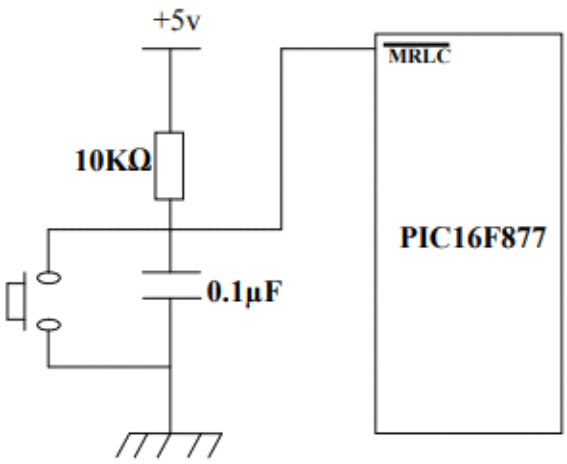


Figure I.10 : Circuit de reset du microcontrôleur **[14]**

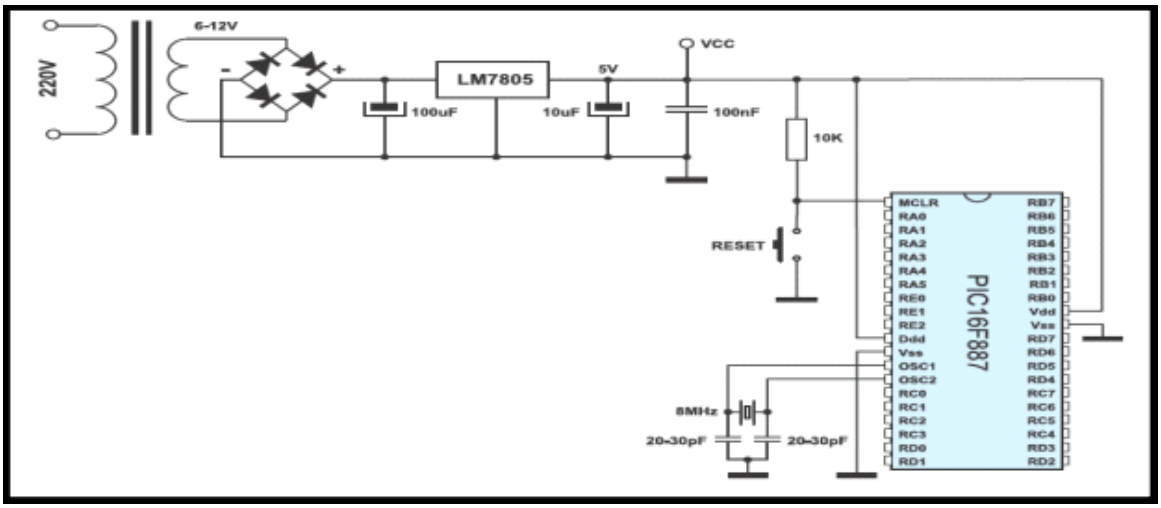


Figure I.11 : boitier du PIC 16F877**[7]**

I.8.2 Architecteur interne :

Sa mémoire programme est composée de trois type de mémoire :

- Une mémoire programmable (Flash), de 8k mots de 14 bits.
- Une mémoire EEPROM de 256 octets.
- Une mémoire vive (RAM) de 386 octets.

Il possède 40 broches, 33 d'entre elles sont des entrées/sorties (PORTA, PORTB, PORTC, PORTD, PORTE), dont 8 peuvent être utilisées comme entées analogiques (RA0, RA1, RA2, RA3, R5, RE0, RE1, RE2) **[14].**

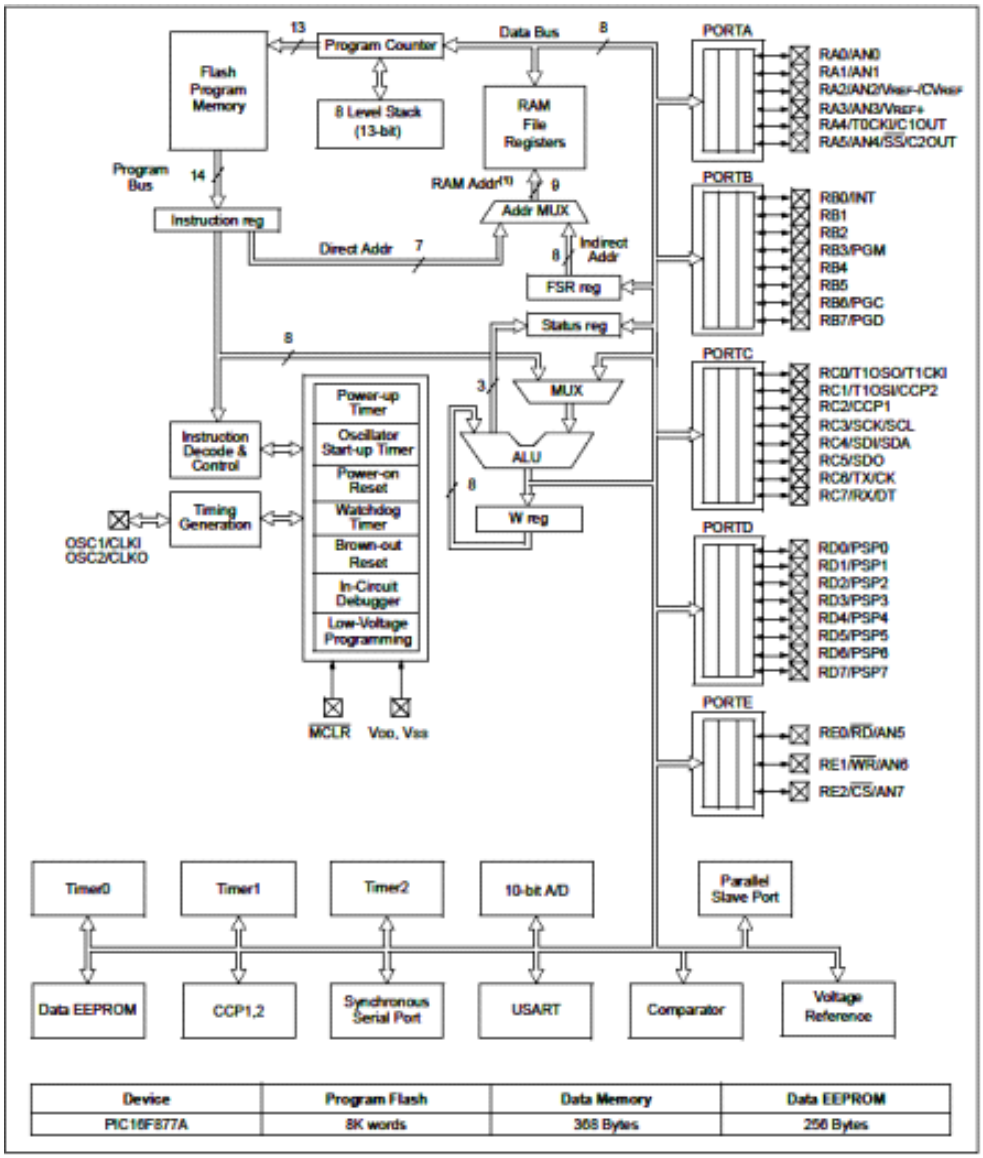


Figure I.12 : Architecteur interne du PIC 16F877 **[14]**

Le CAN est un périphérique intégré destiné à mesurer un signal analogique (une tension électrique) et le convertir en nombre binaire équivalent qui pourra, être utilisé par un programme. Ce module est constitué d'un convertisseur Analogique Numérique 10 bits dont l'entrée analogique peut être connectée sur l'une des 8 (5 pour 16F876) entrées analogiques externes. On

dit qu'on a un CAN à 8 canaux. Les entrées analogiques doivent être configurées en entrée à l'aide des registres TRISA et/ou TRISE. L'échantillonneur bloqueur est intègre, il est constitué d'un Interrupteur d'échantillonnage et d'une capacité de blocage de 120 pF. Les tensions de références permettant de fixer la dynamique du convertisseur. Elles peuvent être choisies parmi V_{dd}, V_{ss}, V_{rf+} ou V_{rf-} **[11].**

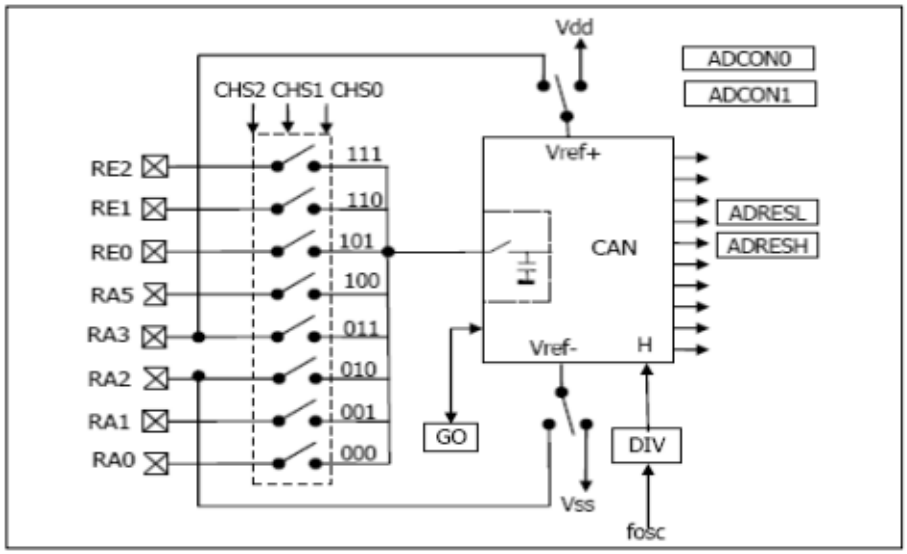


Figure I.13 : Module de conversion analogique/numérique du PIC 16F877 **[14].**

Le microcontrôleur 16F877 travaille avec un convertisseur analogique/numérique qui permet un échantillonnage sur 10 bits. Le signal numérique peut donc prendre 1024 valeurs possibles. On sait que pour pouvoir numériser une grandeur, nous devons connaître la valeur minimale qu'elle peut prendre, ainsi que sa valeur maximale, les PICs considèrent par défaut que la valeur minimale correspond à leur V_{ss} d'alimentation, tandis que la valeur maximale correspond à la tension positive d'alimentation V_{dd}. Le PIC connecte le pin sur laquelle se trouve la tension à mesurer à un condensateur interne, qui va se charger via une résistance interne jusqu'à la tension appliquée. Le pin est déconnecté du condensateur, et ce dernier est connecté sue le convertisseur analogique/numérique interne **[11].**

I.8.2.1 Mémoires internes :

Le PIC 16F877 dispose trois types de mémoire :

- **Mémoire morte FLASH :**

C'et la mémoire programme proprement dite, Chaque mémoire unitaire fait 14 bits, La mémoire FLASH est un type de mémoire stable réinscriptible à volonté, c'est ce nouveau type de mémoire qui fait le succès de microcontrôleur PIC. Dans le cas du 16F877, cette mémoire fait 8k. Lorsque l'on programme en assembleur, on écrit le programme directement dans cette mémoire.

- **Mémoire RAM :**

C'est la mémoire d'accès rapide mais labile (c-ta-dire qu'elle s'efface lorsqu'elle n'est plus sous tension), cette mémoire contient les registres de configuration de PIC ainsi que les différents registres de données, elle contient également les variables utilisées par le programme. Cette mémoire RAM disponible sur le PIC 16F877A est de 368 octets

- **Mémoire EEPROM :**

Cette mémoire programme est de 256 octets, elle est électriquement effaçable, réinscriptible et stable, ce type de mémoire est d'accès plus lent, elle est utilisée pour conservés les paramètres après une coupure de courant.

- **Timer :**

Le timer est un registre de 8 ou 16 bits, qui est incrémenté avec un rythme fixé avec :

- Les impulsions d'une horloge (fonctionnement timer).
- Changement d'états d'un port (fonctionnement compteur).

Dans le cas de fonctionnement timer , le timer est utilisé comme une base de temps.

Le WDT (watchdog Timer) ou COP (Compteur Operating Properce), est un timer qui vérifie le déroulement de programme, si le programme plante, il génère un reset, pour cela il faut initialiser le WDT périodiquement.

I.8.2.2 Les registres :

Ils sont des cases mémoires réserves qui gèrent la manière d'exécution des programmes et indiquent les résultats de certaines opérations.

Le nombre et les noms des registres varient d'un microcontrôleur à l'autre, mais certains registres basiques sont présents dans tous les microcontrôleurs

- **Le registre accumulateur :** Il maintient le résultat d'une opération calculé par le CPU.
- **Le registre d'index :** Il est utilisé afin de spécifier une adresse dans le cas d'un adressage indirecte **[11]**.

I.8.2.3 Les convertisseurs analogiques / numériques :

Pour utiliser un convertisseur analogique / numérique, il existe trois registres internes nommés ADCON0, ADCON1 et ADCON2, qui permettront de configurer et de sélectionner les différentes entrées analogiques (ADC pour Analog to Digital Converter) du PIC utilisé. Le paramétrage de ces trois registres conduit à choisir la façon dont se comporteront le ou les convertisseurs utilisés. Les entrées du PIC en relation avec le convertisseur analogique/ digital Interne sont les broches AN0 à AN12. Les entrées analogiques (RA0 à RA5) transitent via le PORT A du PIC, les entrées AN5 à AN7 font partie du PORT E et les entrées AN8 à AN12 font partie du port B.

Comme vous le voyez sur le schéma général, les bits CHS0 à CHS3 du registre ADCON0 permettent de sélectionner un canal (AN0 à AN12) qui sera en relation avec le convertisseur. Les bits VCFG0 et VCFG1 du registre ADCON1 permettent à eux de définir les références de tension (AN2 ou V_{ss} pour V_{ref-} et AN3 ou V_{dd} pour V_{ref+}). Les bits PCFG0 à PCFG3 du même registre permettent de configurer chaque broche ANO à AN12 soit en entrée analogique soit en entrée / sortie logique. Le troisième registre ADCON2 permettra de sélectionner une horloge pour le convertisseur analogique (bits ADCS0 à ADCS2), ainsi qu'un temps d'acquisition avant conversion (temps nommé TAD : bits ACQT0 à ACQT2), allant de 0 TAD à 20 TAD (soit 0 top d'horloge à 20 tops d'horloge), puis indiquera grâce au bit ADFM le format de la valeur de la conversion en sortie.

Lorsqu'une conversion est lancée, un bit nommé GO/DONE du registre ADCON0 est forcée à «1». Dès que la conversion en cours est terminée, ce bit repasse à « 0 ». On réalise donc une boucle d'attente tant que la conversion n'est pas échue. Notez également que le bit ADON du

même registre permet d'alimenter le convertisseur en 5 V : si ce bit est à « 0 », alors le convertisseur ne consomme aucun courant. Le résultat de la conversion sera stocké dans deux registres 8 bits nommes ADRESH et ADRESL. Le résultat étant sur 10 bits, il faudra choisir (ajustement à droite ou ajustement à gauche) les 10 bits parmi les 16 contenus dans ces deux registres, le bit ADFM du registre de configuration ADCON2 aura ce rôle **[11].**

1.8.2.4 Control du module de conversion analogique/numérique

Le control du module se fait par les deux registres ADCON0, ADCON1.

- **ADCON0 :**

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	--------	ADON

ADCS1-ADCS0 : Choix de l'horloge de conversion donc du temps de conversion.

00 : Fosc/2

01 : Fosc/8

10 : Fosc/32

11 : Oscillateur RC dédié au CAN.

CHS2-CHS0 : choix de l'entrée analogique.

000 = Channel 0, (RA0)

001 = Channel 1, (RA1)

010 = Channel 2, (RA2)

011 = Channel 3, (RA3)

100 = Channel 4, (RA5)

101 = Channel 5, (RE0)

110 = Channel 6, (RE1)

111 = Channel 7, (RE2)

GO/DONE : une conversion démarre quand on place ce bit à 1. A la fin de la conversion, il est remis automatiquement à zéro.

ADON : Ce bit permet de mettre le module AN en service

- **ADCON1 :**

ADFM	------	------	------	PCFG3	PCFG2	PCFG1	PCFG0

ADFM : justification à droite ou à gauche du résultat dans le registre ADRESH et ADRESL.

	ADRESH	ADRESL
1 : justifié à droite	000000XX	XXXXXXXX.
0 : justifié à gauche	XXXXXXXX	XX000000.

PCFG3-PCFG0 : configuration des E/S et des tensions de références. Les 5 broches de PORTA et les 3 de PORTE peuvent être configurés soit en E/S digitales, soit en entrées analogiques. RA2 et RA3 peuvent aussi être configurées en entrée de référence **[14].**

I.9. Conclusion :

Dans ce chapitre on a présenté le microcontrôleur PIC16F877, un composant clé dans les systèmes embarqués modernes. Il a détaillé l'évolution des microcontrôleurs, en soulignant leur importance croissante dans divers appareils électroniques. Le choix d'un microcontrôleur repose sur plusieurs critères, tels que le nombre d'entrées/sorties, la mémoire, la vitesse de traitement et les logiciels compatibles.

Le PIC16F877 se distingue par sa flexibilité et ses performances, notamment grâce à son architecture RISC optimisée, sa mémoire flash réinscriptible et ses nombreux ports de communication. Son architecture interne intègre plusieurs fonctionnalités avancées, comme un convertisseur analogique/numérique et un module PWM, permettant une large gamme d'applications.

En conclusion, le PIC16F877 est un choix idéal pour les projets nécessitant un microcontrôleur puissant, polyvalent et abordable, adapté aux systèmes embarqués et à l'automatisation.

Chapitre II : techniques d'affichage

II.1. Introduction

Les dispositifs d'affichage (Display device) sont souvent appelés systèmes d'affichage jouent un rôle très important dans la vie quotidienne spécifiquement dans le domaine de l'information et de la communication. Ils sont utilisés dans de nombreux domaines, allant du simple affichage de texte sur un ordinateur à la projection d'images complexes, sur des murs ou des surfaces. Voici quelques exemples de dispositifs d'affichage :

II.2. Affichage 7 segments :

II.2.1. Présentation :

L'afficheur 7 segments est un dispositif électronique utilisé pour afficher des chiffres de 0 à 9 et quelques caractères alphanumériques. Il est largement employé dans les systèmes embarqués en raison de sa simplicité et de sa lisibilité**[1].**

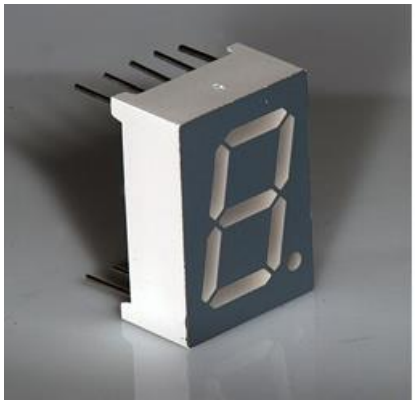


Figure II.1 : Afficheur LED 7 segments avec point décimal, boîtier DIP-10 large.**[17]**

Ce type d'affichage repose sur 7 segments lumineux, généralement des diodes électroluminescentes (LED), disposées de manière à former une représentation du chiffre affiché.

Un huitième segment, souvent désigné par "dp" (decimal point), est ajouté pour afficher un point ou une virgule décimale. Les afficheurs 7 segments sont très présents dans les calculatrices, horloges numériques, compteurs électroniques, thermomètres et divers dispositifs industriels nécessitant un affichage numérique simple et efficace.

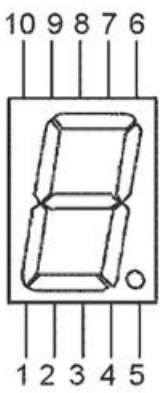
II.2.2. Brochage

Un afficheur 7 segments est constitué de 10 broches :

- 8 broches pour les 7 segments et le point décimal (DP).
- 2 broches pour la connexion commune (anode commune ou cathode commune).

A	B	C	D	E	F	G	DP
7	6	4	2	1	9	10	5

❖ Remarque : La disposition des broches peut varier selon le modèle de l'afficheur. Il est donc important de consulter sa data-sheet pour une utilisation correcte.



PIN NO.	E MAN6960
1	Cathode E
2	Cathode D
3	Com. Anode
4	Cathode C
5	Cathode D.P.
6	Cathode B
7	Cathode A
8	Com. Anode
9	Cathode F
10	Cathode G

Figure II.2 : Brochage de l'afficheur 7 segment.**[17]**

II.2.3. Principe de Fonctionnement :

Le fonctionnement d'un afficheur 7 segments repose sur l'activation sélective des LEDs internes pour former le chiffre souhaité.

- Chaque segment est une LED qui s'illumine lorsqu'un courant le traverse.

- L'affichage d'un chiffre est réalisé en allumant certains segments et en éteignant les autres.

- Le PIC16F877 ou un autre microcontrôleur envoie des signaux électriques aux segments correspondants pour afficher le chiffre désiré **[18].**

II.2.4. Types d'afficheurs 7 segments :

Les afficheurs 7 segments existent en deux configurations principales :

II.2.4.1. Anode commune (CA - Common Anode) :

- Toutes les anodes des LEDs sont reliées ensemble.
- Chaque segment est activé en mettant sa cathode à 0V (LOW).
- Adapté aux circuits nécessitant un pilotage par masse (courant sinking)

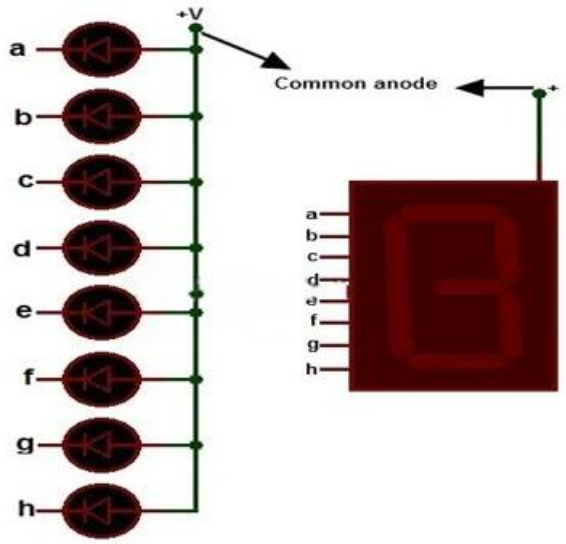


Figure II.3 : un afficheur 7 segment anode commune.[19]

Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	
0	0	0	0	0	0	1	0
1	0	0	1	1	1	1	1
0	0	1	0	0	1	0	2
0	0	0	0	1	1	0	3
1	0	0	1	1	0	0	4
0	1	0	0	1	0	0	5
0	1	0	0	0	0	0	6
0	0	0	1	1	1	1	7
0	0	0	0	0	0	0	8
0	0	0	0	1	1	0	9

Tableau II.1 : un afficheur 7 segment anode commune [19].

II.2.4.2. Cathode commune (CC - Common Cathode) :

- Toutes les cathodes des LEDs sont reliées ensemble.
- Chaque segment est activé en mettant son anode à 5V (HIGH).
- Plus simple à utiliser avec un microcontrôleur.

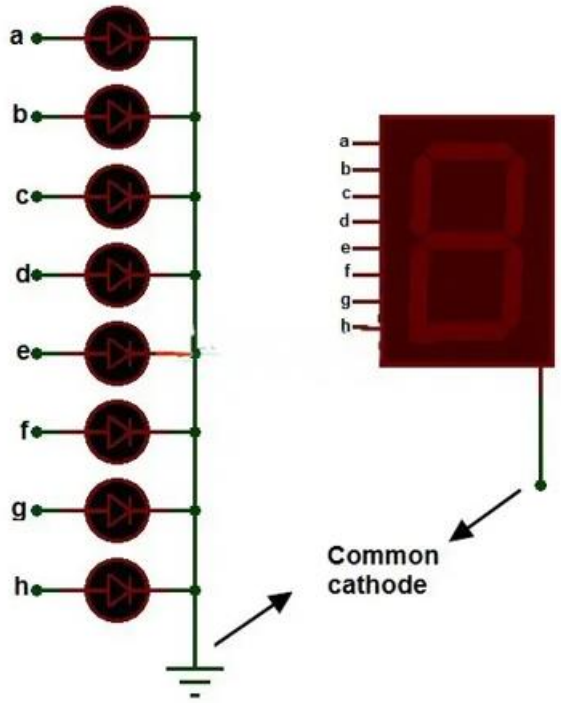


Figure II.4 : un afficheur 7 segment cathode commune [19].

Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	
1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	2
1	1	1	1	0	0	1	3
0	1	1	0	0	1	1	4
1	0	1	1	0	1	1	5
1	0	1	1	1	1	1	6
1	1	1	0	0	0	0	7
1	1	1	1	1	1	1	8
1	1	1	1	0	0	1	9

Tableau II.2 : un afficheur 7 segment cathode commune[19].

- Différence clé :

  - Anode Commune → Pilotage actif bas (LOW).

  - Cathode Commune → Pilotage actif haut (HIGH)

✅ Avantages	❌ Inconvénients
Simplicité : Facile à comprendre et à interfacer avec un microcontrôleur comme le PIC16F877.	Affichage limité : Ne peut afficher que des chiffres et quelques lettres (A, b, C, d, E, F…).
Faible coût : Moins cher que d'autres technologies d'affichage comme les LCD ou OLED.	Nombre de broches élevé : Chaque afficheur nécessite 7 à 8 broches, ce qui peut être un problème si plusieurs afficheurs sont utilisés.
Bonne visibilité : Lisible même en lumière ambiante élevée.	Pas de couleurs variées : Généralement rouge, vert ou jaune, contrairement aux écrans LCD ou OLED.
Consommation modérée : Plus économique que les écrans LCD en termes de puissance.	Multiplexage nécessaire : Pour afficher plusieurs chiffres, une gestion dynamique des affichages est requise.

Tableau II.3 : Avantages et inconvenients de affichage 7 segment

II.3. Technologie des LEDs :

La LED (Light Emitting Diode – Diode électroluminescente) est un composant optoélectronique qui convertit directement l'énergie électrique en lumière grâce au phénomène d'électroluminescence. Ces composants, devenus omniprésents dans les systèmes d'éclairage, les dispositifs d'affichage et les communications optiques, offrent une alternative efficace, durable et fiable aux sources lumineuses traditionnelles.

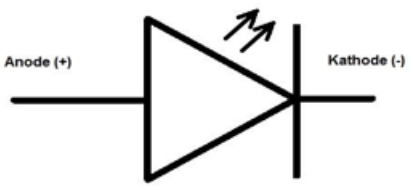


Figure II.5 : symbole de la LED [20].

II.3.1. Principe de fonctionnement :

Une LED est un composant électronique qui émet de la lumière lorsqu'un courant la traverse dans le sens direct. Ce phénomène est appelé électroluminescence : lors du passage du courant, les électrons traversent la jonction PN et libèrent de l'énergie sous forme de photons (lumière). Contrairement aux sources lumineuses classiques, les LEDs ne contiennent ni filament, ni gaz : elles sont donc plus robustes et plus fiables.

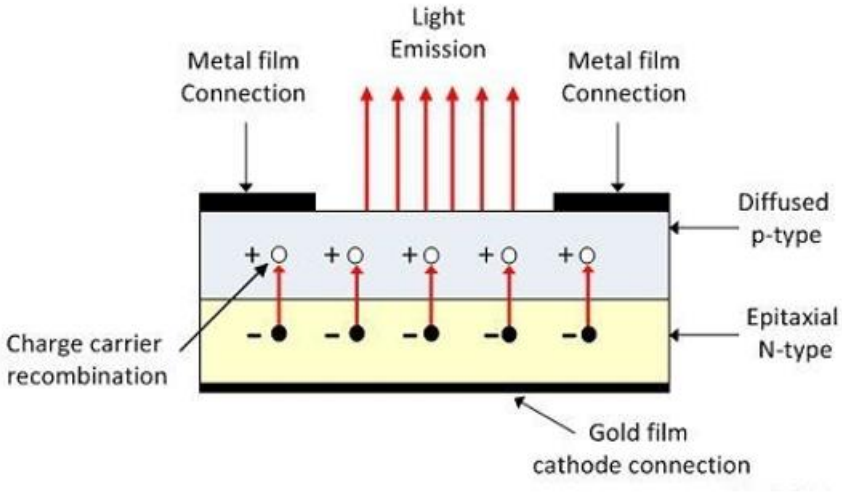


Figure II.6 : Structure de base d'une diode électroluminescente [21].

II.3.2. Caractéristiques électriques d'une LED :

- **Tension de seuil :** Elle est généralement plus élevée que celle d'une diode classique. Cette tension dépend de la couleur de la LED, c'est-à-dire de la longueur d'onde de la lumière qu'elle émet. Par exemple, une LED rouge fonctionnera autour de 2V, tandis qu'une LED bleue ou blanche nécessitera entre 3V et 3,6V.
- **Courant direct moyen** : Il varie entre 5 mA et 20 mA pour une LED standard. Il est important de respecter cette valeur pour éviter d'endommager le composant.

- **Courant de crête (en impulsion)** : Dans certaines applications, une LED peut être brièvement soumise à un courant plus élevé, à condition que la durée soit très courte et que la fréquence soit maîtrisée.
- **Tension inverse maximale** : Elle est généralement limitée à 5 V. Les LEDs sont très sensibles à une polarisation inverse et peuvent être rapidement endommagées si cette limite est dépassée.

II.3.3. Caractéristiques optiques d'une LED :

- **Couleur et longueur d'onde** : La lumière émise par une LED dépend directement du matériau utilisé. Chaque type de LED produit une couleur spécifique correspondant à une longueur d'onde dominante (exprimée en nanomètres). Par exemple, une LED rouge émettra vers 620 nm, une verte vers 530 nm, et une bleue vers 470 nm.
- **Longueur d'onde de pic** : C'est la valeur précise à laquelle l'intensité lumineuse de la LED est maximale.
- **Largeur spectrale** : Contrairement aux sources lumineuses classiques, le spectre d'émission d'une LED est relativement étroit, ce qui permet une couleur plus "pure".
- **Angle de rayonnement** : La lumière n'est pas émise uniformément dans toutes les directions. La forme de la LED (dôme, lentille, surface plane, etc.) détermine l'angle d'émission, mesuré à mi-intensité (lorsque la lumière chute à 50 % de son maximum). On parle alors de LED "directives" (angle étroit) ou "diffuses" (angle large).
- **Intensité lumineuse**: L'intensité lumineuse (mesurée en candelas) est la quantité de lumière émise dans une certaine direction à 1 mètre de distance.[22]

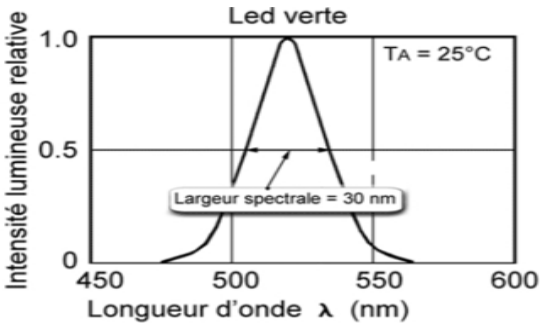


Figure II.7 : La langueur d'onde du spectre [22].

II.4. Les matrices à LEDs :

Une matrice LED est un affichage composé de plusieurs diodes électroluminescentes (LED) disposées en lignes et colonnes. Par exemple, une matrice 8x8 contient 64 LED, organisées en 8 lignes et 8 colonnes. Chaque point lumineux (pixel) est à l'intersection d'une ligne et d'une colonne.

Figure II.8 : matrice LED [23].

II.4.1. Principe de fonctionnement d'une matrice LED :

Dans une matrice LED, chaque LED est connectée à une ligne (anode ou cathode) et une colonne (cathode ou anode). Pour allumer une LED spécifique, il faut :

- Activer sa ligne (mettre à un niveau logique approprié)

- Activer sa colonne (mettre l'autre niveau logique)

Cependant, toutes les LED ne sont pas allumées en même temps : on utilise un principe de multiplixage

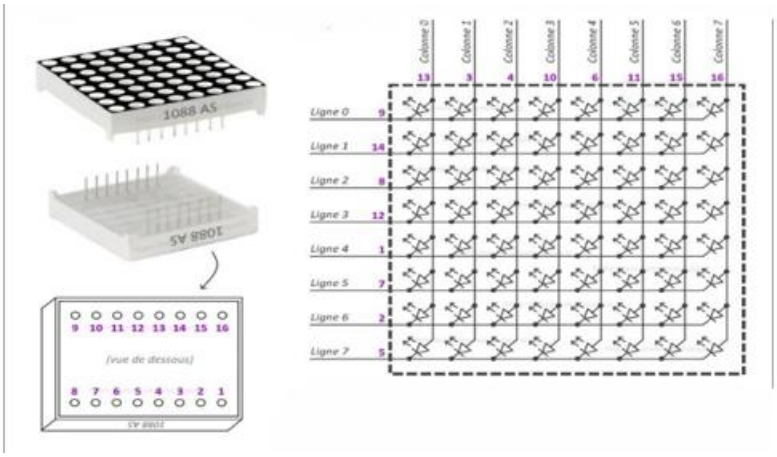


Figure II.9 : structure d'une matrice LED [24].

Le multiplexage consiste à activer une seule ligne ou colonne à la fois, pendant un court instant, tout en affichant les LED souhaitées sur cette ligne ou colonne. On répète cette opération très rapidement (plusieurs centaines de fois par seconde), ce qui donne l'illusion que toutes les LED sont allumées en permanence.

Avantage :

- Réduction du nombre de broches nécessaires
- Contrôle dynamique et flexible
- Moindre consommation instantanée

II.4.2. Circuit de pilotage :

Pour contrôler une matrice LED, on utilise généralement un microcontrôleur en combinaison avec un circuit intégré tel que le 74LS05. Ce dernier joue un rôle essentiel dans la gestion des colonnes ou des lignes en fonction de la configuration choisie.

Le 74LS05 est un circuit logique TTL comportant 6 inverseurs à sorties en collecteur ouvert (open collector). Cela signifie que chaque sortie peut uniquement tirer le signal vers le niveau bas (0 logique), et nécessite donc une résistance pull-up pour obtenir un niveau haut.

Ce type de sortie est idéal pour commander des LED, car il permet de les connecter directement à la masse lorsque le signal est activé.

Schéma typique :

- Les colonnes sont connectées aux sorties du 74LS05.
- Les lignes sont connectées aux sorties du microcontrôleur.

II.5. Journal lumineux :

Comme l'afficheur alphanumérique est un traducteur électro-optique qui transforme une énergie électrique en une autre lumineuse ou plus généralement en une information visible, l'affichage est réalisé en allumant en même temps ou à des instants différents et périodiques les point lumineux nécessaires du tableau pour communiquer l'information désirée, en ce sens le tableau d'affichage est constitué de points lumineux couramment appelés pixels.

Le journal lumineux dépend naturellement de la densité des points par unité de surface et la surface totale de la matrice est elle-même liée à la distance d'où elle doit être lue. Ainsi, pour un affichage sur un panneau géant, ces matrices peuvent avoir des dimensions assez grandes pour afficher un ou quelques caractères. On peut aussi réaliser de nombreux effet lumineux qui se succèdent automatiquement comme par exemples : défilement de gauche à droite ou de haut vers le bas ou bien affichage des messages ou des signes [25].

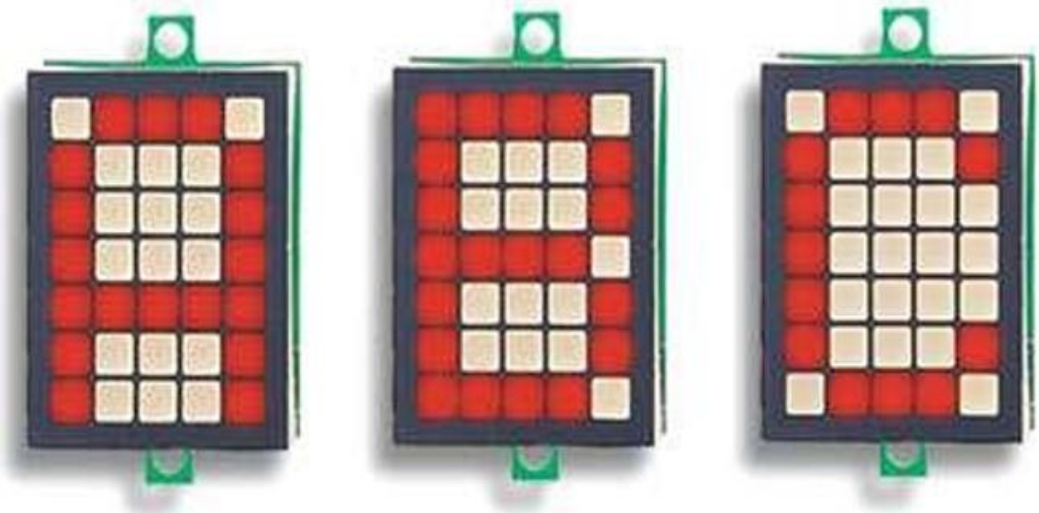


Figure II.10 : affichage sur une matrice à LED.

II.6. Introduction aux écrans LCD :

Un écran LCD (*Liquid Crystal Display*) est un dispositif d'affichage à faible consommation énergétique qui utilise la modulation de la lumière à travers des cristaux liquides pour produire des caractères ou des images. Ces écrans sont largement utilisés dans les appareils électroniques modernes, allant des montres numériques aux systèmes embarqués.

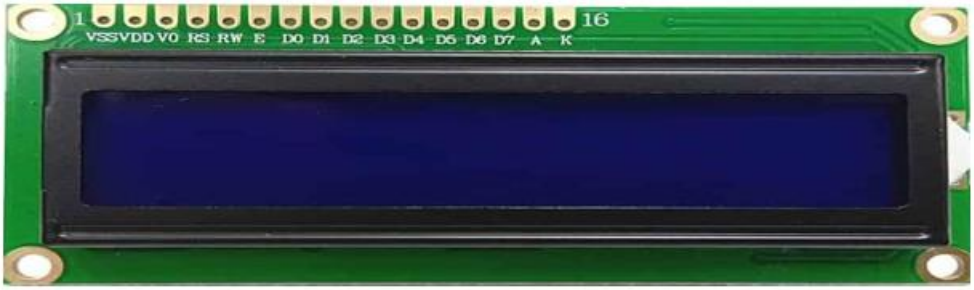


Figure II.11 : un afficheur LCD.

II.6.1. Principe de fonctionnement d'un écran LCD :

Un écran LCD est constitué de deux plaques de verre entre lesquelles sont enfermés des cristaux liquides. Chaque plaque est recouverte d'un filtre polarisant, les deux étant orientés perpendiculairement. Le fonctionnement repose sur les étapes suivantes :

- **Sans tension appliquée**, les cristaux liquides tournent le plan de polarisation de la lumière, permettant à celle-ci de traverser les deux filtres et rendant le pixel visible (clair).
- **Avec tension appliquée**, les cristaux s'alignent dans le champ électrique et n'altèrent plus la polarisation de la lumière, bloquant ainsi sa transmission. Le pixel apparaît alors sombre.
- Ce mécanisme, combiné à un système de **rétroéclairage** (souvent à LED), permet l'affichage lisible de caractères dans diverses conditions de luminosité.

Le contrôle de l'affichage est assuré par un **circuit intégré** qui pilote individuellement les segments selon les données reçues du microcontrôleur.

II.6.2. Exemple d'implémentation : le module LCD 1602 :

Le **LCD 1602** est un module d'affichage alphanumérique largement utilisé dans les systèmes électroniques embarqués pour son coût réduit, sa facilité d'intégration et sa compatibilité avec de nombreux microcontrôleurs (comme l'Arduino).

II.6.3. Caractéristiques techniques du LCD 1602 :

- **Affichage** : 2 lignes de 16 caractères, soit 32 caractères au total.
- **Contrôleur intégré** : HD44780, standard de l'industrie.
- **Interface de communication** : Parallèle (4 ou 8 bits) ou I²C avec adaptateur.
- **Tension de fonctionnement** : 5 V (certains modèles acceptent le 3,3 V).
- **Consommation** : faible (1 à 5 mA sans rétroéclairage).
- **Rétroéclairage** : LED interne (généralement bleue ou verte).
- **Contraste** : ajustable par un potentiomètre connecté à la broche V0.
- **Temps de réponse** : de quelques millisecondes, variable selon la température.[26]

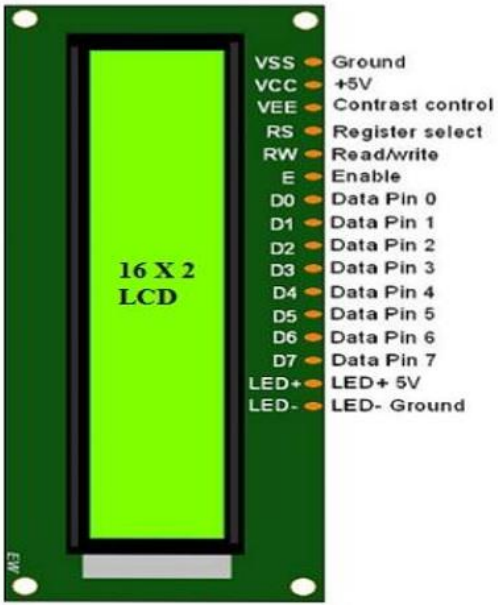


Figure II.12 : LCD 1602 module [27].

II.6.4. Fonctionnement dans un système embarqué :

- Le microcontrôleur envoie des **commandes** (ex. : effacer l'écran, positionner le curseur) et des **données** (caractères à afficher) au contrôleur HD44780.
- L'écran peut être piloté en **mode 8 bits** (plus rapide, mais plus de broches nécessaires) ou en **mode 4 bits** (plus économique en broches).
- Un adaptateur **I²C** est souvent utilisé pour simplifier le câblage, réduisant le nombre de broches à deux (SDA et SCL).

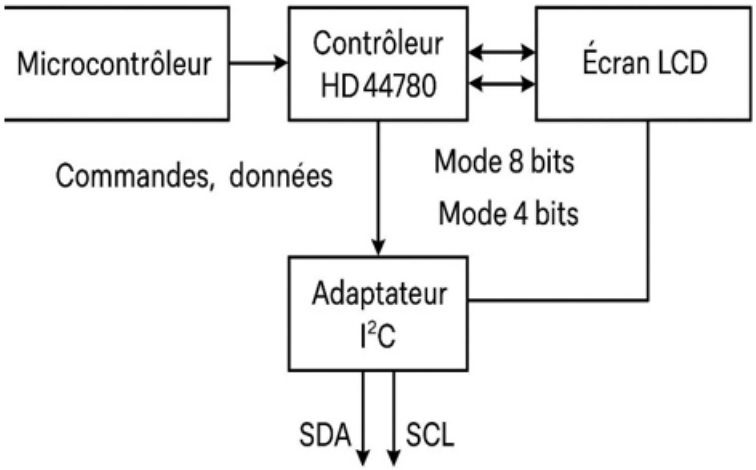


Figure II.13 : Schéma de fonctionnement d'un écran LCD HD44780 piloté par un microcontrôleur avec adaptateur I²C.

Ce schéma illustre le fonctionnement d'un système embarqué utilisant un écran LCD contrôlé par un contrôleur HD44780, souvent via un adaptateur I²C. Voici l'explication des éléments et du flux :

- ❖ Microcontrôleur : C'est le cerveau du système. Il envoie : des commandes (comme "effacer l'écran", "placer le curseur") des données (caractères à afficher).
- ❖ Contrôleur HD44780.

Ce composant reçoit les instructions du microcontrôleur et contrôle l'écran LCD. II peut fonctionner :

En mode 8 bits (transfert rapide mais utilise 8 lignes de données)

Ou en mode 4 bits (transfert plus lent mais avec seulement 4 lignes de données)

- Écran LCD : Affiche les caractères ou informations selon ce que le contrôleur HD44780 lui envoie.
- Adaptateur I^2C : Pour simplifier les connexions, on peut intercaler un adaptateur I^2C entre le microcontrôleur et le contrôleur HD44780. Il convertit les signaux I^2C (SDA/SCL) en commandes compréhensibles par le HD44780.
- SDA: ligne de données
- SCL: ligne d'horloges

II.7.Conclusion :

Dans ce chapitre nous avons montré le principales technologies d'affichage, et nous avons étudié beaucoup sur les afficheurs 7 segments et notamment les matrices à LED et les afficheurs à cristaux liquides. Cette étude sera l'élément principal de tout ce qui suivra.

Chapitre III :
Les logiciels utilisés

III.1. Introduction :

La programmation des microcontrôleurs PIC peut être réalisée à l'aide de plusieurs langages et environnements, tels que **MikroC for PIC,** MPLAB**,** MikroBasic PRO for PIC, HI-TECH C for PIC **et** Flowcode. Pour notre projet, nous avons choisi **le compilateur MikroC**, qui est basé sur le langage **C**, un langage évolué permettant également l'intégration de certaines routines en **assembleur**. Ce choix s'explique à la fois par des préférences personnelles et des considérations technologiques, car **le langage C** est à la fois **accessible, flexible et efficace** pour le développement des microcontrôleurs.

III.2. Etapes de développement du programme :

L'élaboration d'un programme est un travail qui se fait en plusieurs étapes (**Figure. III.1**) :

La première étape : L'algorithme

La deuxième étape : Ecriture du programme

La troisième étape : Simulation du programme

La quatrième étape : Transfert du programme vers PIC **[11].**

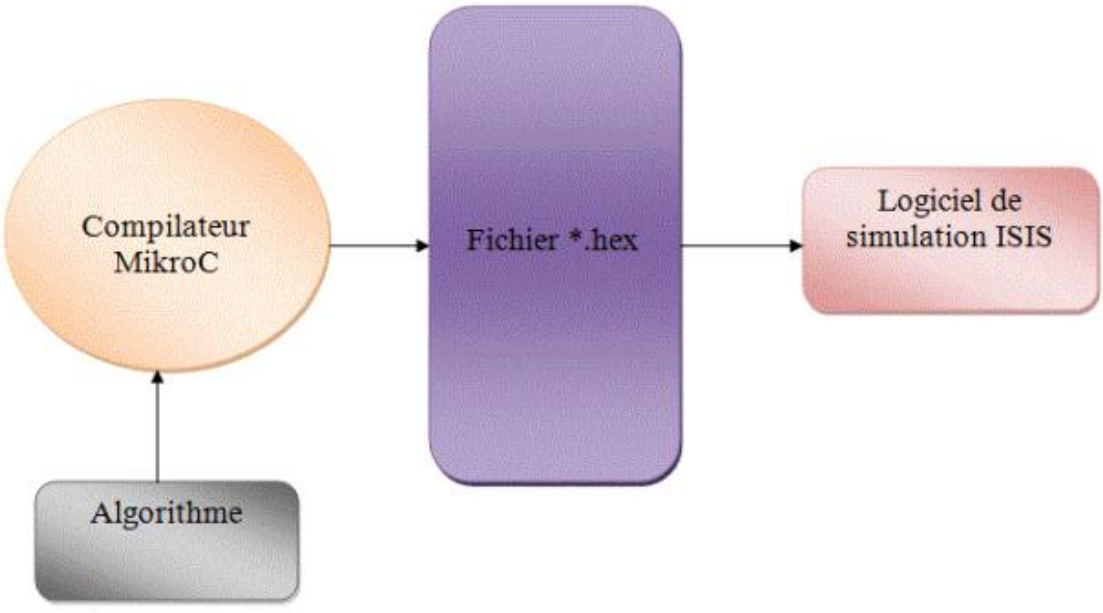


Figure III.1: Etape dévolution d'un programme **[28]**

III.3. Présentation du MikroC :

Le MikroC est un puissant outil pour les microcontrôleurs PIC. Il est conçu pour fournir au Programmeur la solution la plus simple possible pour développer des applications pour les systèmes embarqués, sans compromettre les performances ou le contrôle.

Le MikroC permet de développer et déployer des applications complexes :

- Écrire un code source C en utilisant le très avancé éditeur de code.
- Utiliser les bibliothèques du MikroC fait accélérer considérablement le développement (Acquisition de données, la mémoire, affichage, les conversions, les communications.....).
- Surveiller la structure du programme variables et fonctions dans l'explorateur de code.
- Inspecter le déroulement du programme et déboguer la logique d'exécution avec l'intégration debugger, obtenez des rapports détaillés et des graphiques sur les statistiques du Code.

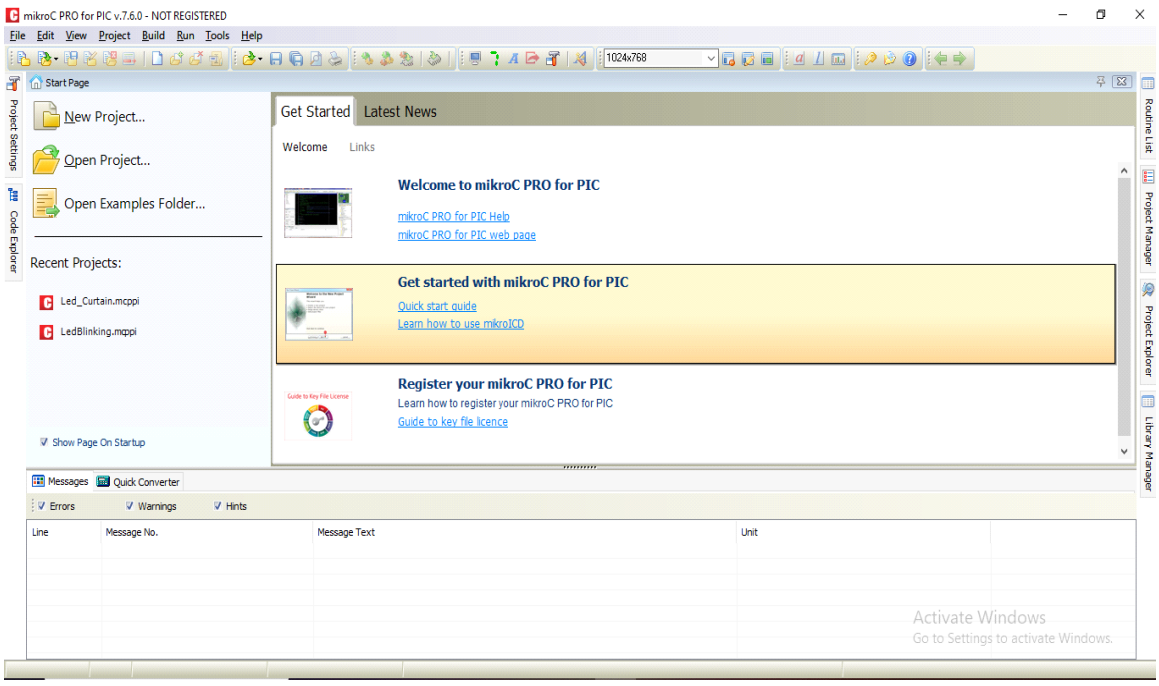


Figure III.2 : L'ouverture du mickroC

Nouvelle version appelée mikroC PRO dispose de très nombreuses améliorations du compilateur mikroC : nouvelles variables utilisables, nouvelle interface IDE, amélioration des

performances du linker et de l'optimisateur, cycle de compilation plus rapide, code machine généré plus compact (jusqu'à 40 % suivant les cas), nouveaux PIC supportés, environnement de développement encore plus ergonomique, nouveaux exemples d'applications, etc..**[13]**

III.3.1 Création d'un projet :

Le mikroC PRO pour PIC organise des applications dans des projets, composé d'un seul fichier de projet (extension. mcppi) et un ou plusieurs fichiers sources (extension). Les fichiers source peuvent être compilés que si elles font partie d'un projet. Le fichier projet contient les informations suivantes :

- Nom du projet et une description facultative.
- Composant cible.
- Option du composant.
- Fréquence d'horloge du composant.
- La liste des fichiers source du projet avec les chemins.
- Fichiers d'image.
- Fichiers binaires (* mcl.).
- D'autres fichiers **[28].**

La méthode la plus efficace pour créer un projet consiste à utiliser l'Assistant Nouveau Projet, accessible via le **(**menu **Project > New Project)** ou en cliquant sur l'icône **Nouveau Projet** dans la barre d'outils du projet.

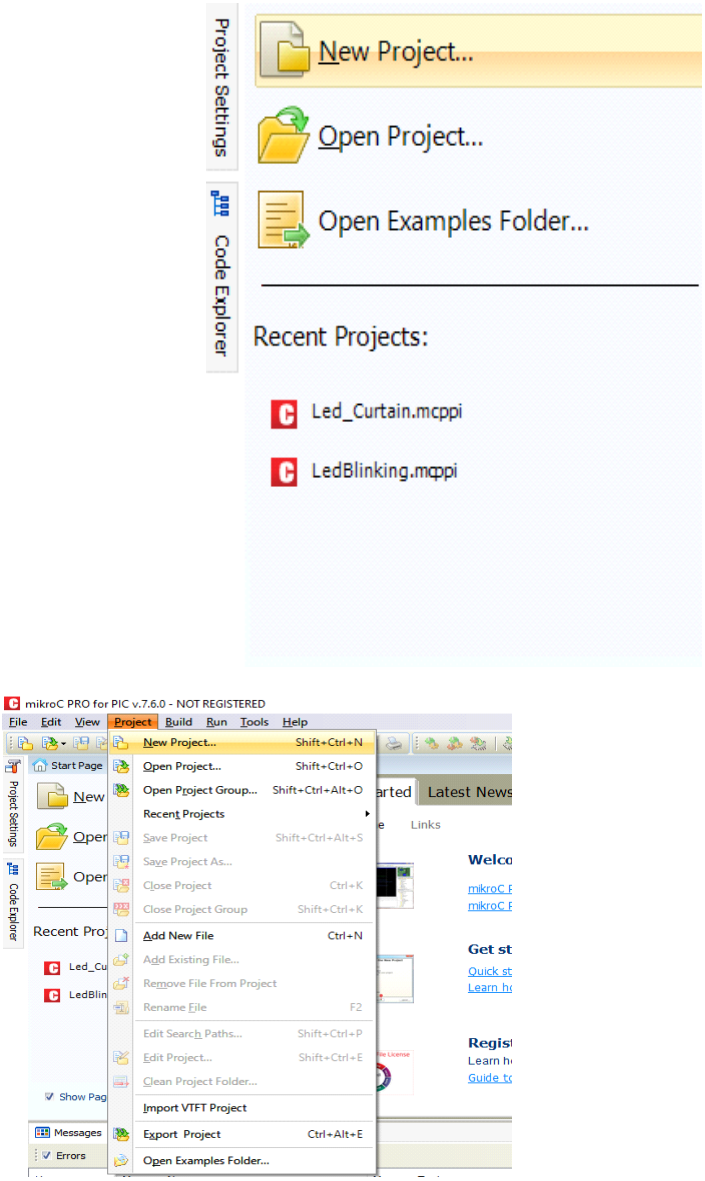


Figure III.3 : Création d'un projet en MikroC

Après la 1ère étape on a une nouvelle fenêtre apparaîtra, il y a plusieurs champs à renseigner comme le nom du projet, l'emplacement du projet, sa description, l'horloge et les options du composant. Comme indiqué sur (Figure III.4).

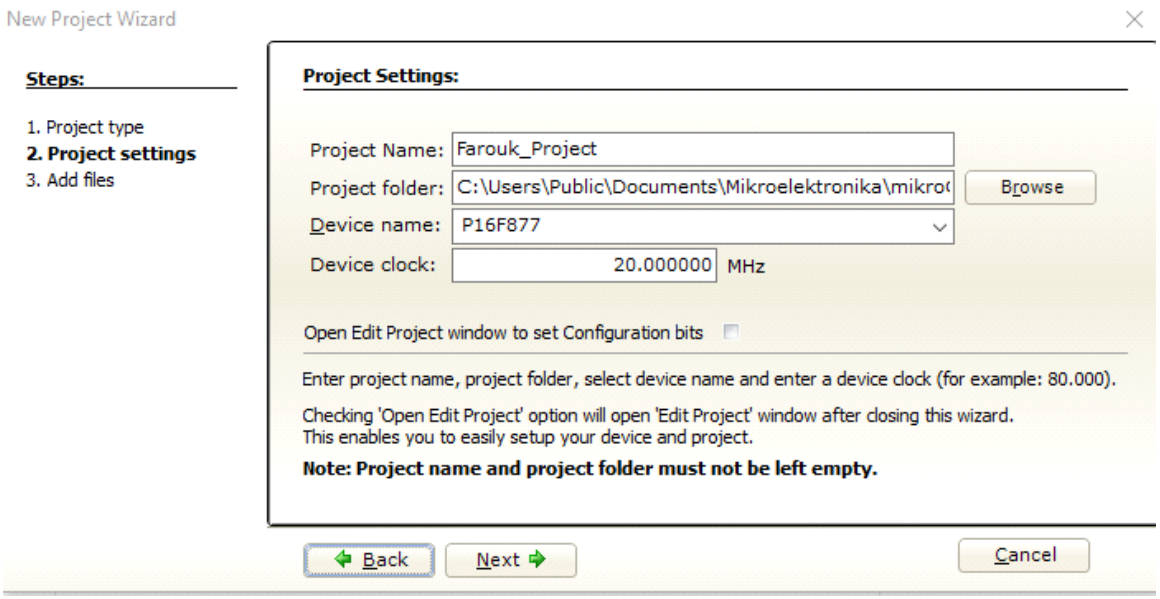


Figure III.4 : Les configurations de projet

Après ça, une nouvelle fenêtre (Figure III.5) vide s'affiche pour écrire notre programme

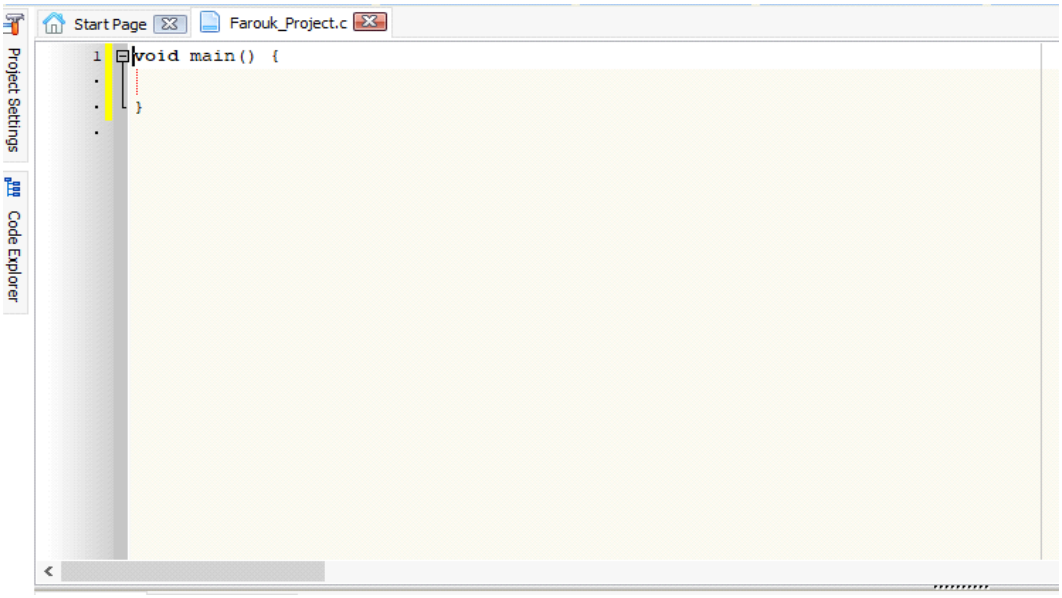


Figure III.5 : Les programmes

III.3.2 Compilation :

Une fois le projet créé et le code source rédigé, il ne reste plus qu'à le compiler. Pour cela, vous pouvez choisir **Project Build** dans le menu déroulant ou simplement cliquer sur l'icône **Build** dans la barre d'outils du projet.

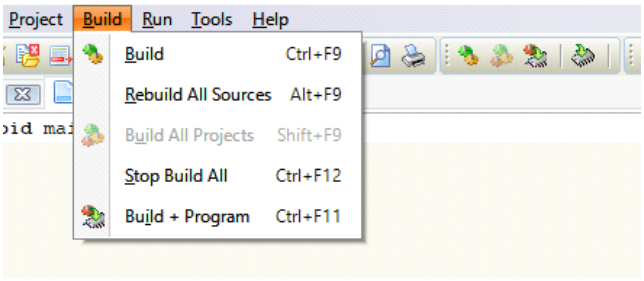


Figure III.6 : La compulsation de programme.

III.3.3 Les commentaires :

Les commentaires sont utilisés pour préciser le fonctionnement du programme et pour une annotation du programme. Les lignes de commentaires sont ignorées et non compile par le compilateur.

En mikroC les commentaires de programmes peuvent être de deux types : de longs commentaires (s'étendant sur plusieurs lignes) et de courts commentaires (occupant une seule ligne). Un commentaire au début d'un programme peut décrire brièvement le fonctionnement du programme et fournir le nom de l'auteur, le nom de fichier de programme, la date à laquelle le programme a été écrit, et une liste des numéros de version, ainsi que la modification dans chaque version **[17].**

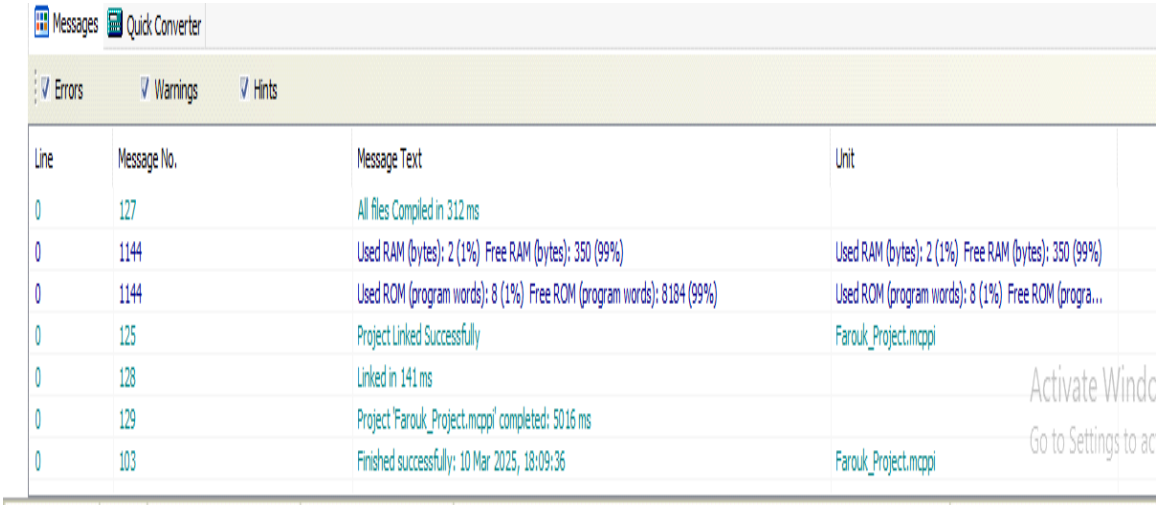


Figure III.7 : Affichage des résultats.

Après la compilation réussie, le compilateur mikroC PRO pour PIC génère des fichiers de sortie dans le dossier du projet (dossier qui contient le fichier projet. mcppi). Les fichiers de sortie sont résumés dans le tableau ci-dessous :

Format	Description	Type de fichier
Intel HEX	Code hexadécimal dans le format Intel. Fichier est utilisé pour programmer PIC	*.hex*
Binary	Fichier compilé pour la bibliothèque *mikroC*. Les distributions binaires sont des routines qui susceptibles	*.mcl*

Tableau III.1: Les fichiers de sortie **[28].**

III.3.4 Sauvegarder un fichier :

Pour cela vérifiez que la fenêtre du fichier à sauvegarder est bien active. Ensuite, choisissez Save dans le menu File, appuyez sur **Ctrl+S**, ou cliquez sur l'icône Save as dans la barre d'outils File (de l'extension .C).

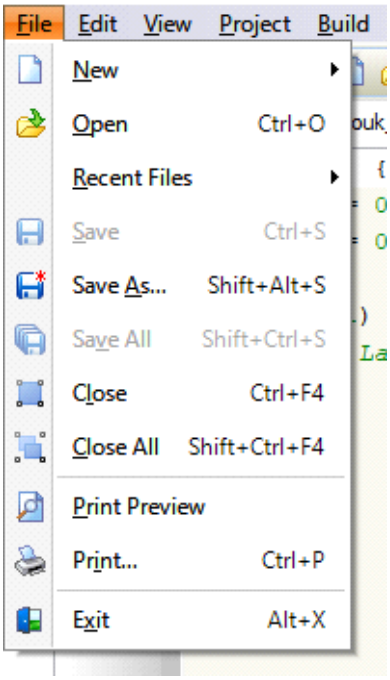


Figure III.8: Sauvegarder un fichier.

III.3.5 Quelques notions de programmation en C sous MikroC:

La saisie d'un programme en 'C' répond pratiquement toujours à la même structure. On peut noter que :

- Le symbole "#" est suivi d'une directive.
- Le symbole "//" est suivi d'un commentaire, ou bien "/*" indiquant le début d'un commentaire et "*/" indiquant sa fin.
- Chaque ligne d'instruction se termine par un " ; ".
- Le début d'une séquence est précédé du symbole " { ".
- La fin d'une séquence est suivie du symbole " } ".
- La notation des nombres peut se faire en décimal de façon normale ou en hexadécimal avec le préfixe "0x" ou encore en binaire avec le préfixe"0b" **[29].**

III.3.6 Déclaration des variables en C :

Les variables de base dans ce compilateur spécifique sont :

- Bit.
- Char.
- Short.
- Int.
- Long.
- Float.
- Double.

Dans le tableau ci-dessous, on peut voir les caractéristiques des variables.

Type de variable	*Taille en octets*	*Valeurs*
Bit	*1*	*0 ou 1*
Char	*1*	*-127 à 127*
Short	*1*	*-127 à 127*
Int	*2*	*- 32767 à 32767*
Long	*4*	*- 2147483647 à 2147483647*
Float	*4*	*-1.5x10^45 à 3.4x10^38*
Double	*4*	*-1.5x10^45 à 3.4x10^38*
unsigned char	*1*	*0 à 255*
unsigned short	*1*	*0 à 255*
unsigned int	*2*	*0 à 65535*
unsigned long	*4*	*0 à 4294967295*

Tableau III.2: Les caractéristiques des variables **[29].**

La déclaration des variables s'effectue en indiquant le type de la variable suivi d'un nom que le développeur attribue arbitrairement à la variable. Au moment de la déclaration d'une variable, il est possible de lui donner une valeur initiale, mais ceci n'est pas strictement nécessaire. Enfin, la déclaration doit se terminer par un point-virgule (;).

Exemple :

char *CARACTERE*; // Déclaration d'une variable type char.

III.3.7 Les instructions de MikroC :

On va identifier uniquement les fonctions et les instructions les plus importantes que nous avons utilisées dans notre programme.

A. La boucle ‘for’

Elle permet de contrôler le nombre de fois à exécuter un bloc d'instructions. Dans la syntaxe de cette boucle on trouve : la valeur initiale du compteur, ça valeur finale est la valeur avec laquelle il doit être incrémenté (ou décrémenté) chaque fois que le bloc d'instructions est exécuté **[11].**

```
void main() {
    unsigned short i;  // Déclaration correcte pour MikroC

    TRISB = 0x00;  // Configure le PORTB en sortie
    PORTB = 0x00;  // Initialise PORTB à 0

    while(1) {  // Boucle infinie
        for(i = 0; i < 10; i++) {  // Boucle de 0 à 9
            PORTB.F0 = 1;  // Allumer la LED sur RB0
            Delay_ms(500); // Attendre 500ms
            PORTB.F0 = 0;  // Éteindre la LED
            Delay_ms(500); // Attendre 500ms
        }
    }
}
```

Figure III.9: Programme utilise boucle for.

B. Opération de condition 'if/else'

Cet opérateur permet d9exécuter une partie du programme sous certaines conditions. La syntaxe de cet opérateur est la suivante.

```
void main() {
    TRISB = 0x00;  // Configure le PORTB en sortie
    PORTB = 0x00;  // Initialise PORTB à 0

    while(1) {  // Boucle infinie
        if(PORTB.F0 == 0) {  // Vérifie si la LED est éteinte
            PORTB.F0 = 1;  // Allume la LED
            Delay_ms(500); // Attendre 500ms
        } else {
            PORTB.F0 = 0;  // Éteint la LED
            Delay_ms(500); // Attendre 500ms
        }
    }
}
```

Figure III.10: Programme utilise condition (if/else).

III.4. Présentation du logiciel PROTEUS :

Le PROTEUS Professionnel est un logiciel qui permet le développement et la simulation des applications dans le domaine électronique via un environnement graphique simple et interactif utilisé dans les Pic de gamme MICROSHIP. C'est un éditeur de schémas qui intègre un simulateur analogique et logique ou parfois mixte.

Toutes les opérations se passent dans cet environnement, aussi bien la configuration de différentes sources que le placement des sondes et le tracé des courbes. Il est également capable de simuler le comportement d'un microcontrôleur (Atmel, 8051, ARM, HC11...) et leurs interactions avec les composants qui l'entourent et il est largement utilisé comme outil pédagogique ces dernières années, en particulier pour sa simplicité.

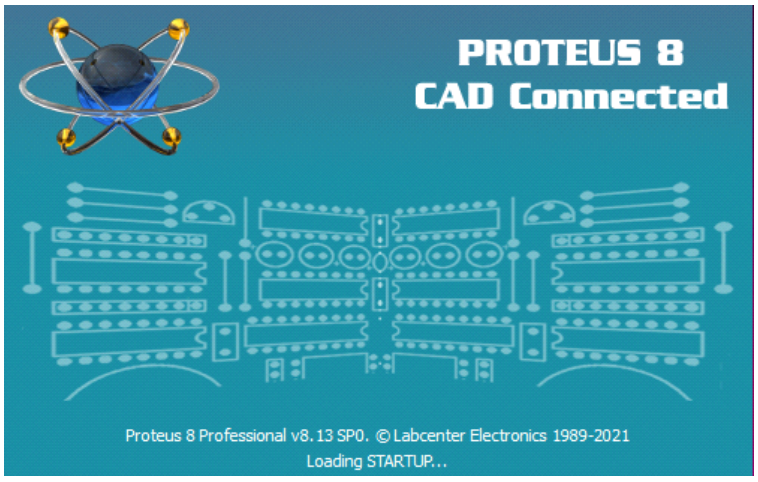


Figure III.11: Page d'accueil du logiciel PROTEUS Professionnel v8.13.

Le logiciel PROTEUS est une suite logicielle permettant la CAO électronique (Construction Assistée par Ordinateur), éditée par la société Labcenter Electronics qui s'est positionnée pour plus de 10 ans comme l'un des outils les plus utiles pour simuler les microcontrôleurs PIC.

Il est composé de deux principaux logiciels :

- ISIS qui permet entre autres la création des schémas et la simulation électrique.
- ARES dédié à la création des circuits imprimés grâce à des modules additionnels.

Outre la popularité de l'outil, PROTEUS possède d'autres avantages tels que :

- Le PROTEUS est un pack contenant des logiciels faciles et rapides à comprendre et à utiliser.
- Le PROTEUS possède un support technique est performant.
- Le PROTEUS est un outil de création de prototype virtuel permettant de réduire les coûts matériels et logiciels lors de la conception d'un projet **[1].**

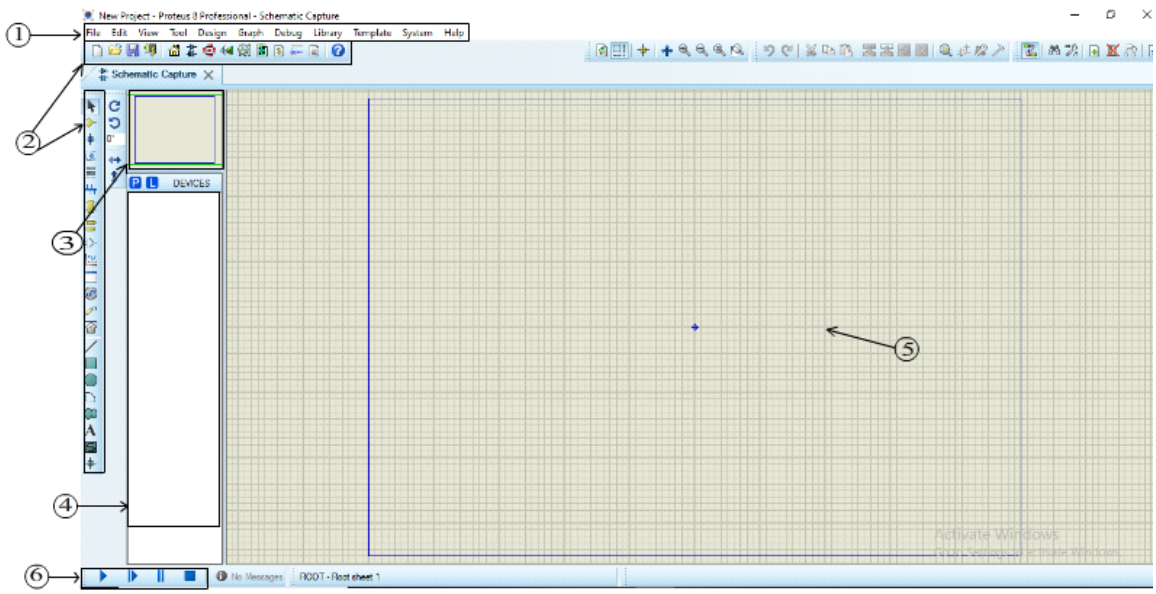


Figure III.12 : Représentation de l'interface ISIS du logiciel PROTEUS

- **Barre de Menu :** Contient les différentes fonctionnalités du logiciel (File, View, Edit…).
- **Barre d'Outils :** Contient les icônes permettant l'accès direct aux fonctions du logiciel.
- **Zone d'Observation :** Affiche une vue d'ensemble du circuit en cours de réalisation.
- **Zone des Objets :** Contient la liste des objets sélectionnés précédemment.
- **Espace de Travail :** Espace réservé à la réalisation des circuits électroniques.
- **Boutons de simulation** : Permettant le démarrage/Arrêt de la simulation.

III. 5 Conclusion :

Dans ce chapitre, nous avons exploré l'utilisation des logiciels MikroCPRO for PIC et Proteus 8 Professional pour la programmation et la simulation de microcontrôleurs PIC. MikroC PRO a été choisi pour sa flexibilité et sa richesse en bibliothèques facilitant le développement, tandis que Proteus a permis de simuler efficacement les circuits électroniques et d'optimiser le prototypage.

Grâce à ces outils, nous avons pu structurer notre démarche en plusieurs étapes, allant de la rédaction du programme en langage C à sa simulation et sa validation sur Proteus. Cette approche nous a offert un environnement fiable pour tester et corriger notre code avant son implémentation finale sur le microcontrôleur.

Ainsi, l'utilisation combinée de MikroC et Proteus constitue une méthode efficace pour le développement de systèmes embarqués, en minimisant les erreurs et en optimisant le temps de conception.

Chapitre IV
Réalisation et Simulation

IV.1. Introduction :

Après une étude générale des différents éléments constituants notre carte électronique, on passe maintenant à la simulation et la réalisation physique de notre projet. Dans cette partie, la programmation du PIC est très importante qui nécessite à comprendre les langages de programmation en utilisant les outils de développement software suivants :

- Le MikroC pro pour la programmation (déjà explique dans chapitre 3).
- Proteus 8 pour faire des tests et la simulation (existant dans chapitre 3).
- PicKit 3 pour le transfert des files (Hex file).

IV.2. Logiciel PicKit 3 :

Le programmateur PicKit est un environnement de développement intégré fourni par Microchip. Ce logiciel permet aux développeurs de compiler le programme et téléchargez le code sous forme de fichier hexadécimal et transférez-le vers PIC (" **joue le rôle d'intermédiaire entre la programmation et l'appareil cible** ").

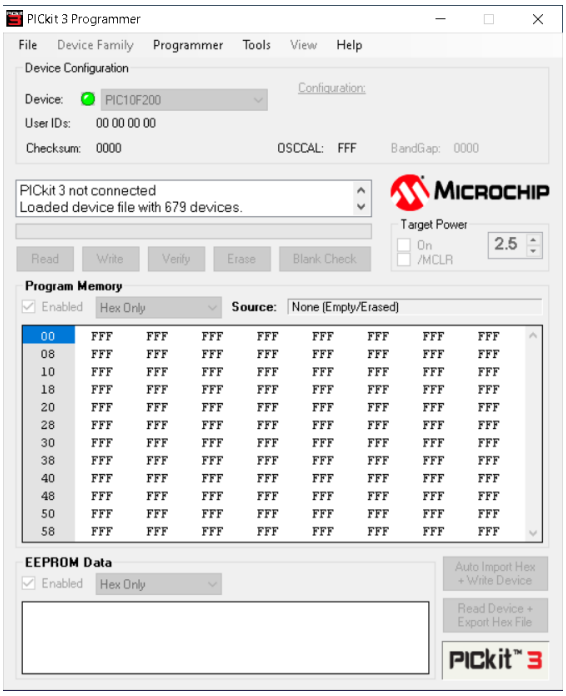


Figure IV.1 : l'interface de PicKit Programmer.

- Ce programme comprend plusieurs fonctionnalités (Apres la liaison de PicKit coté hardware), dont les plus importantes sont :

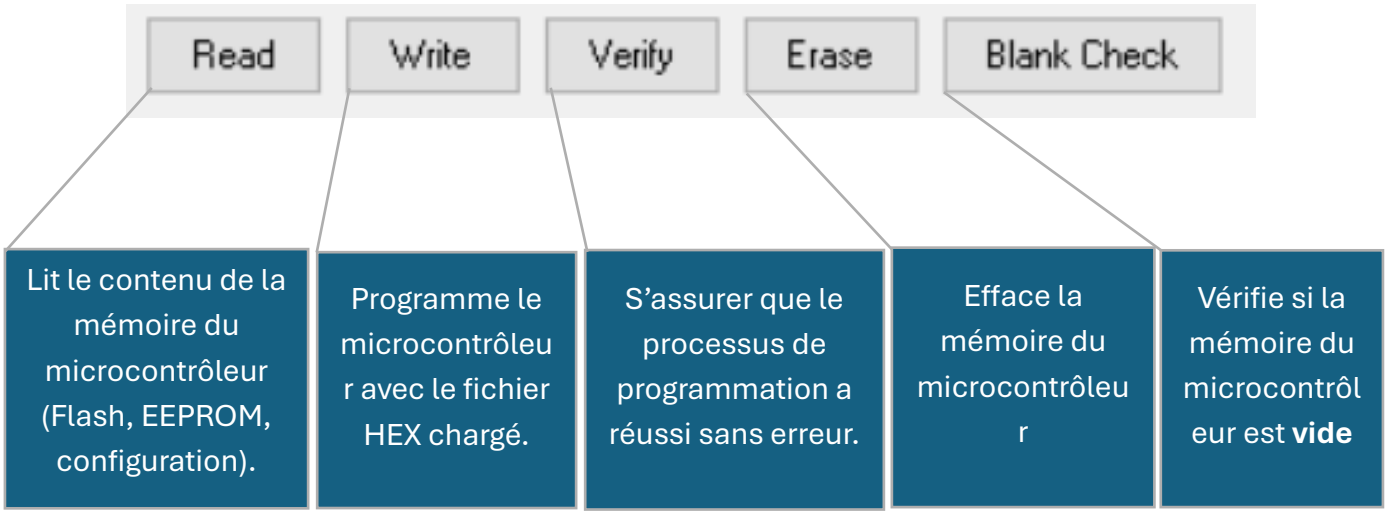
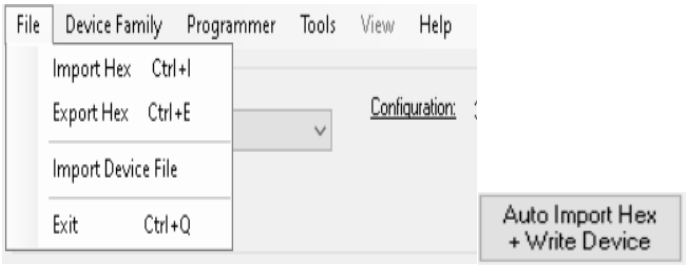


Figure IV.2 : Utilisation du PicKit 3 programmer.

Pour importer un fichier HEX dans le programme PICkit 3, il existe deux méthodes principales. On peut soit utiliser le bouton **"Import HEX"**, ou bien **"Auto Import"**, soit glisser-déposer directement le fichier dans l'interface du logiciel :



IV.2.1. Programmation d'un Pic 16F877 :

1. Consiste à choisir le microcontrôleur PIC utilisé afin que le logiciel puisse configurer correctement la programmation.

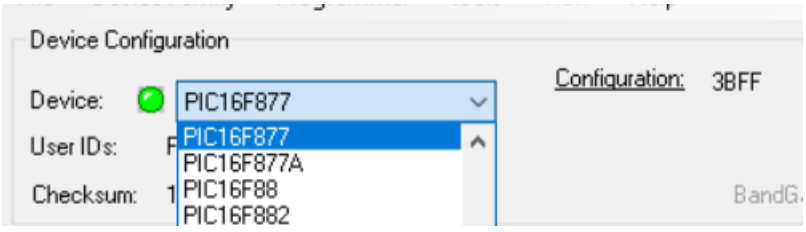


Figure IV.3 : Choix du Pic 16F877.

2. L'importation du programme réalisé pour la carte se fait en chargeant le fichier **HEX** dans le logiciel PICkit 3 afin de le transférer vers le microcontrôleur.

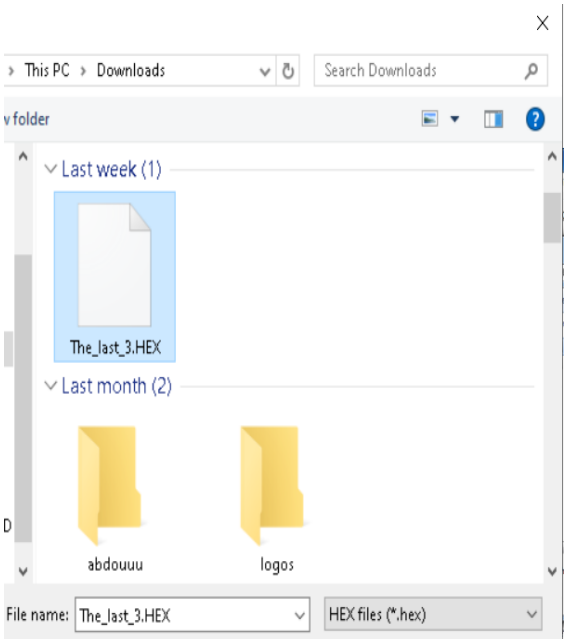


Figure IV.4 : importe Hex file.

3. Ecriture du programme.

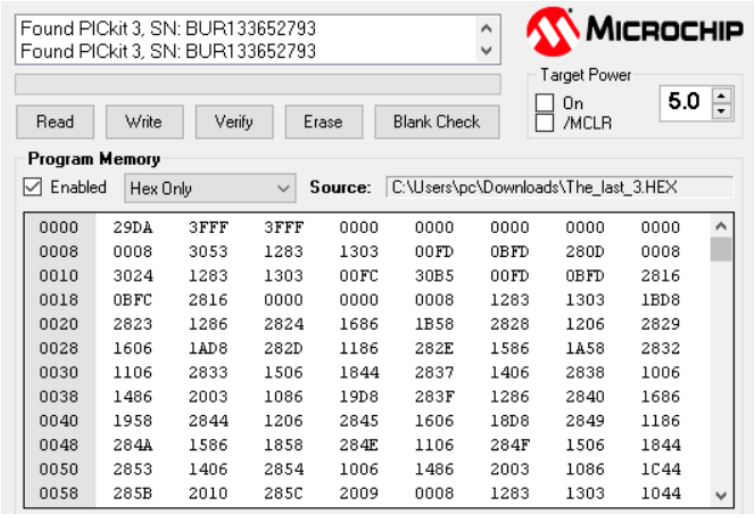


Figure IV.5 : Chargement du fichier HEX en mémoire avant la Programmation.

IV.3. PicKit 3 Hardware :

Figure IV.6 : les composants du PicKit.

Le PICkit 3 est un outil polyvalent qui permet aux développeurs de programmer et de déboguer une large gamme de microcontrôleurs PIC 8 bits et PIC 16 bits de Microchip. Il offre une interface conviviale, une programmation rapide et des capacités de débogage avancées. Le PICkit 3 se connecte à un ordinateur hôte via une interface USB et communique avec le périphérique cible via une interface ICSP (In-Circuit Serial Programming).

Figure IV.7: PICkit 3 MCU PROGRAMMER/DEBUGGER**[30]**

1. Connexion par cordon (***lamyard connectiom***) : Une connexion pratique par cordon est disponible sur le programmateur.
2. Connexion au port USB (***USB Port Connection***) : Le port USB est un connecteur mini-B. Connectez le PICkit 3 au PC à l'aide du câble USB fourni.
3. Marqueur de broche 1 (***Pin 1 Marker***) : Ce marqueur désigne l'emplacement de la broche 1 pour un alignement correct du connecteur.
4. Connecteur de programmation (**Programming Connector**) : Le connecteur de programmation est un connecteur à 6 broches qui se connecte au périphérique cible.

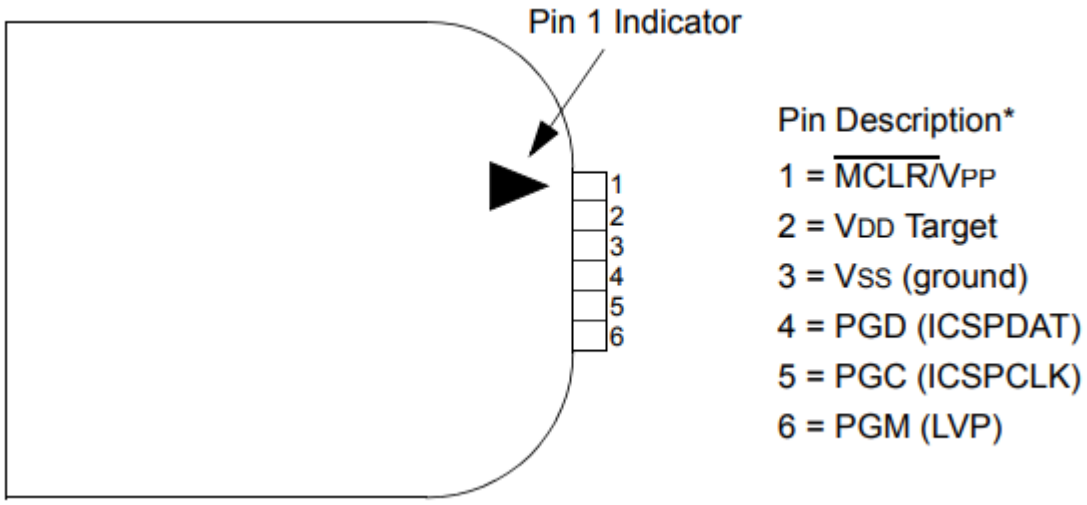


Figure IV.8: PICkit 3 programmer connector pinout **[30]**

5. l'état de LED (**Status leds**) : il contient 4 couleurs :

   Power ( ) : L'alimentation du PICkit 3 est fournie via le port USB.

   Active ( ) : Le PICkit 3 est connecté au port USB du PC et la liaison de communication est active.

   Status :

   Busy( ) : Le PICkit 3 est occupé par une fonction en cours, Comme la programmation.

   Error( ) : Le PICkit 3 a rencontré une erreur.

6. Bouton poussoir **(Push Button) :** Le bouton poussoir est utilisé pour le Programmer-To-Go

IV.3.1. PicKit 3 PinOut :

Le PICkit 3 est équipé d'un connecteur ICSP à 6 broches servant d'interface avec le périphérique cible. Le brochage du connecteur ICSP est le suivant :

Pin Number	Pin Name	Description
1	VPP	Programming Voltage
2	VDD	Target Device Power Supply
3	GND	Ground
4	PGD	Programming Data (ICSPDAT)
5	PGC	Programming Clock (ICSPCLK)
6	LVP	Low Voltage Programming (Optional)

Figure IV.9 : Présentation du brochage du PICKit 3 **[31].**

1. **VPP :** Fournit la haute tension nécessaire pour programmer/effacer la mémoire du microcontrôleur (jusqu'à 14V).
2. **VDD :** Alimente le microcontrôleur (3.3V ou 5V selon le choix).
3. **GND :** Masse commune pour garantir une bonne communication.
4. **PGD :** Transmet les données de programmation (liaison série bidirectionnelle).
5. **PGC :** Fournit l'horloge pour synchroniser les données.
6. **LVP :** Permet la programmation en basse tension (optionnelle selon le PIC) **[31].**

IV.3.2. Avantages de PicKit 3 :

- Ensemble complet pour programmer et déboguer les MCUs de la série PIC.
- Prise en charge de nombreux modèles de microcontrôleurs.
- Fonction de simulation stable et rapide.
- Interface USB pour une connexion facile au PC.

IV.3.3. Limitations de PicKit 3 :

- La connexion à la carte cible doit être effectuée avec précaution pour éviter de brûler le programmateur.
- Pas encore de compatibilité avec les derniers modèles de périphériques PIC **[32].**

IV.4. PicKit 3 Programming Adapter :

La carte universelle de programmation PIC ICD2 / PICkit2 / PICkit3 est un adaptateur compatible avec plusieurs programmateurs Microchip. Elle permet de programmer des microcontrôleurs PIC via un support ZIF (Zero Insertion Force) sans avoir à les souder. La carte

facilite la connexion ICSP avec le programmateur grâce à un brochage standardisé et des cavaliers de configuration. Elle est conçue pour supporter différents modèles de PIC (8 à 40 broches).

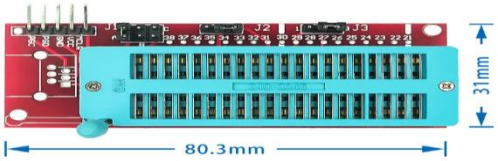


Figure IV.10: PIC ICD2 PICKit2 PICKIT3 Universal Programming Adapter Programmer Board **[33].**

IV.4.1. Avantages:

- **Support ZIF intégré :** insertion/retrait facile des PIC sans endommager les broches.
- **Compatibilité étendue :** prend en charge plusieurs programmateurs (ICD2, PICkit2, PICkit3).
- **Multi-taille** : convient aux PIC DIP de 8 à 40 broches.
- **Programmation hors-circuit** rapide et pratique.

IV.4.2. Inconvénients:

- Coût plus élevé qu'un simple câblage ICSP**.**
- Prend plus de place sur une table de travail.
- Réservé aux boîtiers DIP (pas compatible SMD sans adaptateur).

IV.5. Projet cible : Le projet cible utilise un microcontrôleur PIC16F877 pour contrôler deux matrices LED et un écran LCD. Ce montage permet l'affichage simultané de messages ou symboles sur les matrices, tout en fournissant des informations ou menus via l'écran LCD, idéal pour des projets interactifs ou éducatifs.

IV.5.1. Tableau des composants : Tableau de represantes apect material

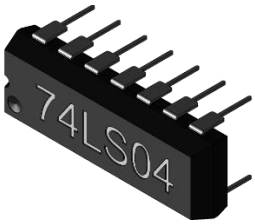
Le Nom	La photo	Le nom	La photo
Matrice 8x8		**Potentiomètre**	
74LS04		**bouton poussoir**	
Condensateur 22PF		**Quartz**	
LCD 1602		**résistance**	
jumper wires cable Male-Male		**Pic16F877**	

Tableau IV.1 : Les composants les plus importants utilisés dans le projet.

IV.6. Programmation et simulation :

IV.6.1. Programmation :

Pour notre carte de commande, on a conçu un algorithme en langage C. Ce dernier, gère la communication du PIC16F877 avec d'autres composants de notre carte, tout en gérant la vitesse d'exécution de chaque action. Il est possible de le défragmenter en plusieurs sous programmes pour mieux commenter son fonctionnement et sa structure. Le code source a été écrit en langage C avec le compilateur MikroC de la société mikro Elektronika. Le code source a été écrit en langage C avec le compilateur MikroC de la société mikro Elektronika

```
#include <built_in.h>    // Inclut les fonctions intégrées de MikroC (comme Delay_ms)
 #include <string.h>     // Inclut les fonctions de manipulation de chaînes (comme strlen)


// LCD configuration
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D4 at RB2_bit;
sbit LCD_D5 at RB3_bit;
sbit LCD_D6 at RB4_bit;
sbit LCD_D7 at RB5_bit;
// Définition des directions des broches (entrée/sortie)
sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D7_Direction at TRISB5_bit;

#define SCROLL_SPEED 100  // Vitesse de défilement (non utilisée ici)
#define LETTER_WIDTH 8    // Largeur de chaque lettre dans la matrice

// Matrix pins(Tableau pour contrôler les lignes de la matrice 8x8)
char dotControl[8] = {1,2,4,8,16,32,64,128};
```

On premier lieux, on doit déclarer toutes les variables avec les quelle nous allons développer notre programme. Mais nous devons déclarer la fonction send_data pour pouvoir l'utiliser à l'envoie des données aux shifts registres.

- **Tableau des caractères**

```
// Font for A-Z and space (uppercase only)
const char font[][8] = {
    {0x10,0x38,0x6C,0xC6,0xFE,0xC6,0xC6,0x00}, // A
    {0xFC,0x66,0x66,0x7C,0x66,0x66,0xFC,0x00}, // B
    {0x3C,0x66,0xC0,0xC0,0xC0,0x66,0x3C,0x00}, // C
    {0xF8,0x6C,0x66,0x66,0x66,0x6C,0xF8,0x00}, // D
    {0xFE,0x62,0x68,0x78,0x68,0x62,0xFE,0x00}, // E
    {0xFE,0x62,0x68,0x78,0x68,0x60,0xF0,0x00}, // F
    {0x3C,0x66,0xC0,0xC0,0xCE,0x66,0x3E,0x00}, // G
    {0xC6,0xC6,0xC6,0xFE,0xC6,0xC6,0xC6,0x00}, // H
    {0x3C,0x18,0x18,0x18,0x18,0x18,0x3C,0x00}, // I
    {0x1E,0x0C,0x0C,0x0C,0xCC,0xCC,0x78,0x00}, // J
    {0xE6,0x66,0x6C,0x78,0x6C,0x66,0xE6,0x00}, // K
    {0xF0,0x60,0x60,0x60,0x62,0x66,0xFE,0x00}, // L
    {0xC6,0xEE,0xFE,0xFE,0xD6,0xC6,0xC6,0x00}, // M
    {0xC6,0xE6,0xF6,0xDE,0xCE,0xC6,0xC6,0x00}, // N
    {0x38,0x6C,0xC6,0xC6,0xC6,0x6C,0x38,0x00}, // O
    {0xFC,0x66,0x66,0x7C,0x60,0x60,0xF0,0x00}, // P
    {0x38,0x6C,0xC6,0xC6,0xD6,0x6C,0x3A,0x00}, // Q
    {0xFC,0x66,0x66,0x7C,0x6C,0x66,0xE6,0x00}, // R
    {0x7C,0xC6,0xE0,0x78,0x1C,0xC6,0x7C,0x00}, // S
    {0xFF,0x99,0x18,0x18,0x18,0x18,0x3C,0x00}, // T
    {0xC6,0xC6,0xC6,0xC6,0xC6,0xC6,0x7C,0x00}, // U
    {0xC6,0xC6,0xC6,0xC6,0xC6,0x6C,0x38,0x00}, // V
    {0xC6,0xC6,0xD6,0xFE,0xFE,0xEE,0xC6,0x00}, // W
    {0xC6,0x6C,0x38,0x10,0x38,0x6C,0xC6,0x00}, // X
    {0xCC,0xCC,0xCC,0x78,0x30,0x30,0x78,0x00}, // Y
    {0xFE,0xC6,0x8C,0x18,0x32,0x66,0xFE,0x00}, // Z
    {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}  // Space
};
```

Figure IV.11: Configuration matérielle et police de caractères pour l'affichage sur matrice LED 8x8 et écran LCD

Chargement des caractères dans la mémoire ROM du PIC, sous forme de code ascii en hexadécimal.

```
// Get letter index: A=0, ..., Z=25, SPACE=26  (Retourne l'indice de la lettre dans le tableau font)
char getLetterIndex(char c) {
    if (c >= 'A' && c <= 'Z') return c - 'A';  // A=0, B=1, ...
    else return 26;  // Si autre chose (comme un espace), retourne l'indice 26
}

// Affiche une colonne complète sur la matrice LED
void displayColumn(char colData[8]) {
    char i;
    for (i = 0; i < 8; i++) {
        PORTD = 0x00;          // Éteindre toutes les lignes
        PORTC = colData[i];    // Envoyer la colonne à afficher
        PORTD = dotControl[i];  // Activer une seule ligne
        Delay_ms(1);            // Petite pause pour visualisation
    }
}

// Scroll text on 8x8 matrix
void scrollText(const char* message) {
    char displayBuf[8];          // Tampon pour afficher 8 colonnes à la fois
    char i, j, k, letterIdx, col;

    // Calcul manuel de la longueur du message
    char msgLen = 0;
    while (message[msgLen] != '\0') msgLen++;

    for (i = 0; i < msgLen; i++) {
        letterIdx = getLetterIndex(message[i]);   // Obtenir l'index de la lettre
```

```
    for (i = 0; i < msgLen; i++) {
        letterIdx = getLetterIndex(message[i]);   // Obtenir l'index de la lettre

        for (col = 0; col < LETTER_WIDTH; col++) {
            // Décalage des colonnes vers la gauche
            for (k = 0; k < 7; k++) {
                displayBuf[k] = displayBuf[k + 1];
            }
            displayBuf[7] = font[letterIdx][col];   // Nouvelle colonne à droite

            // Répéter l'affichage pour rendre visible la colonne
            for (j = 0; j < 15; j++) {
                displayColumn(displayBuf);
            }
        }

        // Ajouter un espace entre les lettres
        for (col = 0; col < 1; col++) {
            for (k = 0; k < 7; k++) {
                displayBuf[k] = displayBuf[k + 1];
            }
            displayBuf[7] = 0x00;  // Colonne vide = séparation

            for (j = 0; j < 15; j++) {
                displayColumn(displayBuf);
            }
        }
    }
}
```

- **Programme principal**

Ensuite, nous avons la partie vitale ou celle responsable des interactions, en se concentrant notamment sur la matrice. La fonction scroll Text traite chaque caractère du message en le convertissant via get Letter Index, puis utilise un tampon (display Buf) pour faire défiler les colonnes de pixels vers la gauche tout en ajoutant les nouvelles colonnes du caractère suivant. La fonction display Column active séquentiellement chaque ligne de la matrice avec un délai de 1ms, créant l'illusion d'un affichage continu grâce au balayage rapide et à la persistance rétinienne.

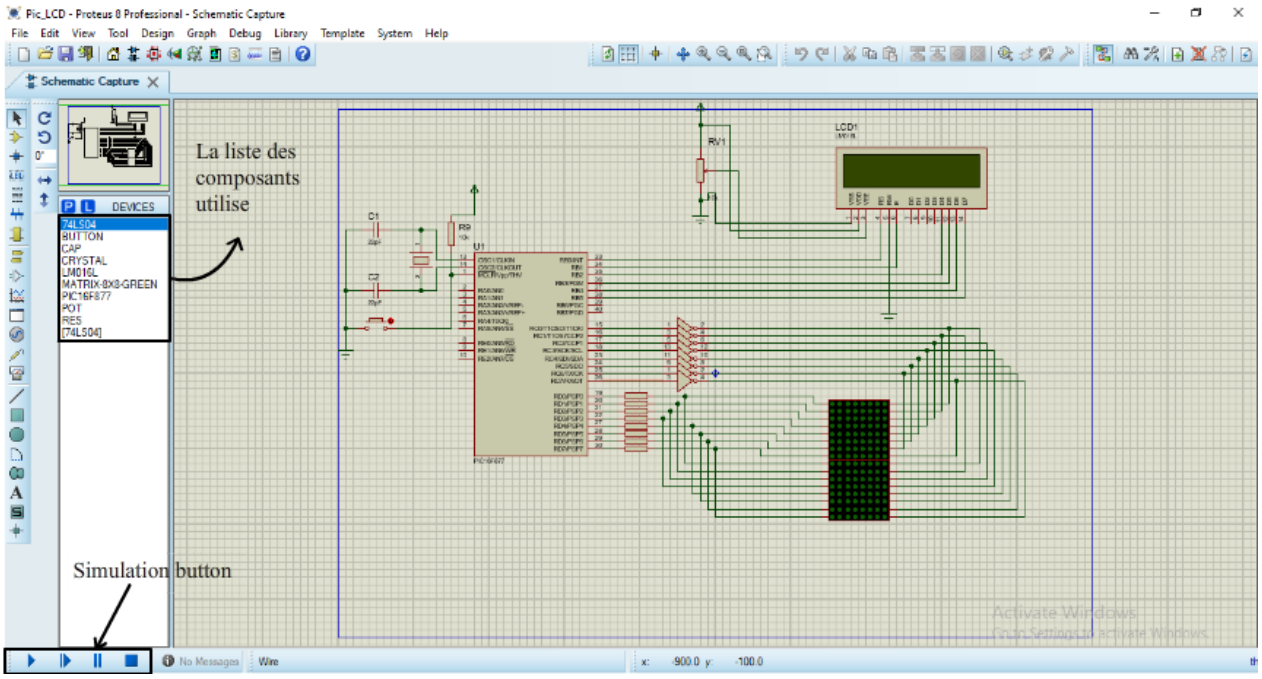
```
void main() {
    TRISB = 0x00; // LCD  (Configurer PORTB comme sortie pour le LCD )
    TRISC = 0x00; // Matrix columns (Configurer PORTC comme sortie pour les colonnes de la matrice)
    TRISD = 0x00; // Matrix rows   (Configurer PORTD comme sortie pour les lignes de la matrice)
    PORTC = 0x00;
    PORTD = 0x00;

    // Initialisation de l'écran LCD
    Lcd_Init();
    Delay_ms(100);
    Lcd_Cmd(_LCD_CLEAR);  // Efface l'écran
    Lcd_Cmd(_LCD_CURSOR_OFF);   // Désactive le curseur
    Lcd_Out(1, 1, "douba farouk");  // Affiche sur la 1ère ligne
    Lcd_Out(2, 1, "fellague ilyes");  // Affiche sur la 2ème ligne

    while (1) {
    // Boucle infinie pour faire défiler le texte sur la matrice
        scrollText("DOUBA FAROUK / FELLAGUE ILYES ");
    }
}
```

Figure IV.12: Configuration et boucle principale du système d'affichage.

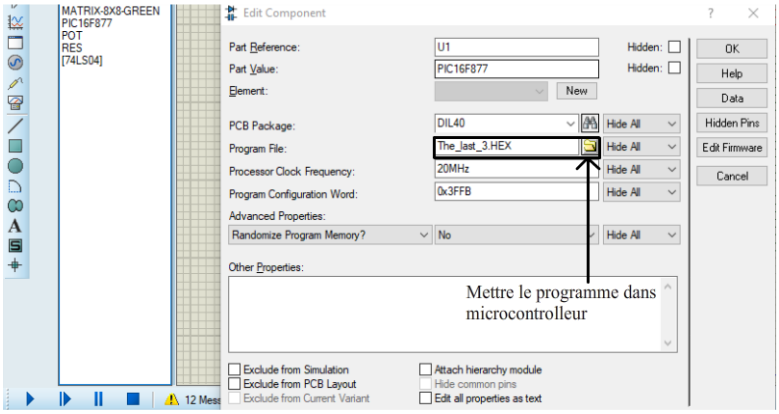
Ce programme permet de contrôler l'éclairage du LCD et de la matrice 8x8 en configurant les ports du PIC16F877 pour envoyer les signaux nécessaires. Il faut donc charger ce code dans le microcontrôleur afin de gérer l'affichage et les animations lumineuses des deux dispositifs.

IV.6.2. Simulation :

Il faut bien évidemment commencer par la saisie du schéma de la carte qu'on souhaite réaliser: Pour saisir le schéma, il faut créer un nouveau projet puis placer les composants qui doivent être sélectionné à partir de la bibliothèque des composants .Dans cette section, nous simulons le travail requis dans un programme de simulation Proteus 8 afin de garantir que le code de programmation fonctionne et de voir aussi les résultats



Apres ca on a va préparer le microcontrôleur par le Hex file dans le but de fonctionnement de système.



La simulation du programme avec MikroC pour le PIC16F877 permet de visualiser le fonctionnement de l'éclairage du LCD et de la matrice 8x8 en temps réel. Après le démarrage, les signaux envoyés aux ports configurés activent correctement les segments du LCD et les LEDs de la matrice, confirmant ainsi le bon fonctionnement du code. La figure 2 illustre clairement les résultats attendus de la simulation.

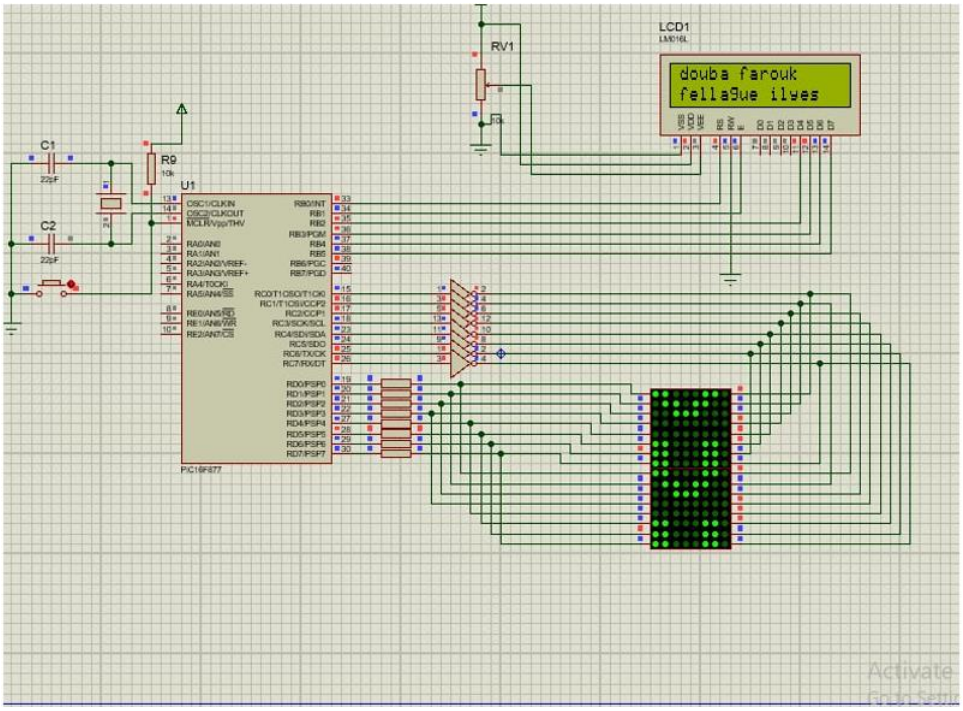


Figure IV.13 : Résultat de simulation du pilotage LCD et matrice 8x8.

IV.7. Résultat de projet :

Le coût de réalisation du projet reste abordable grâce à l'utilisation du microcontrôleur PIC16F877 et des composants standards comme le LCD et la matrice 8x8. Ces éléments sont facilement disponibles sur le marché à des prix raisonnables, ce qui facilite la mise en œuvre du système.

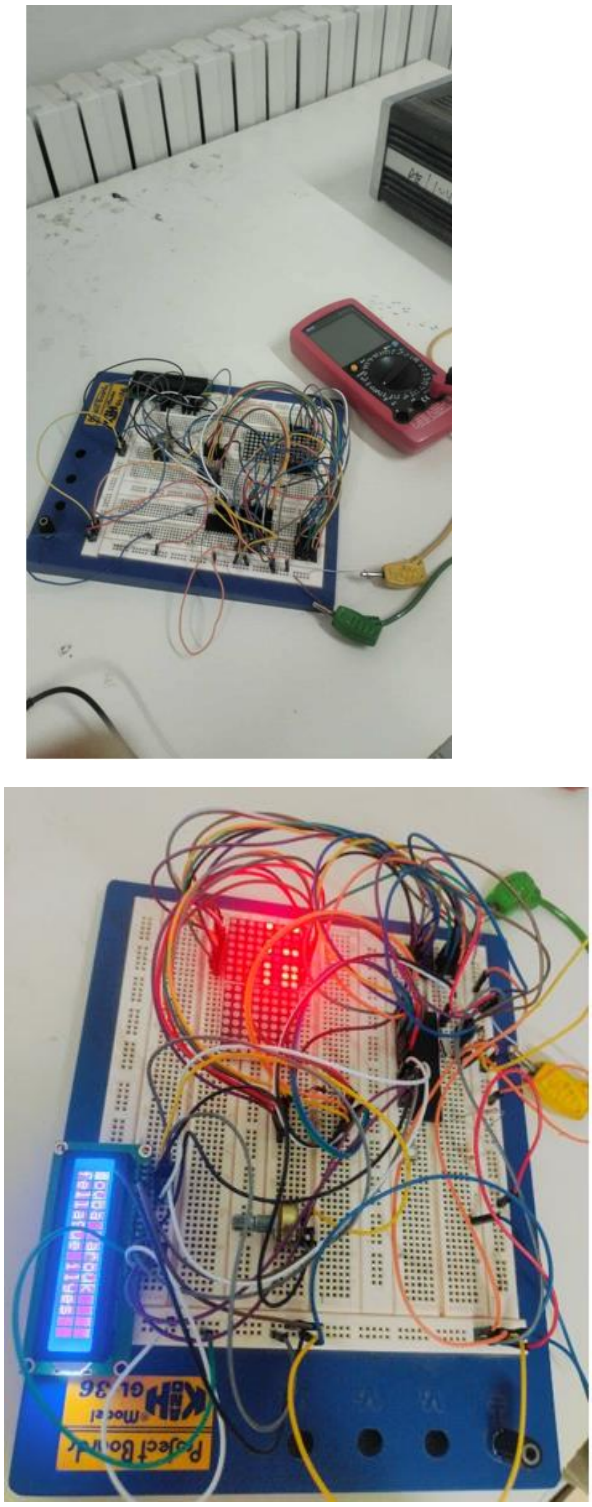


Figure IV.14 : Résultat de la réalisation

IV.8. Conclusion :

Dans ce chapitre, nous avons présenté les différentes étapes de la mise en œuvre pratique de notre projet à l'aide d'un microcontrôleur PIC16F877, en mettant l'accent sur l'utilisation des outils de développement appropriés tels que MikroC Pro, Proteus 8, et PICkit 3. Nous avons d'abord expliqué le fonctionnement du programmateur PICkit 3 et ses différentes fonctionnalités matérielles et logicielles.

Ensuite, nous avons décrit la procédure de programmation et de simulation, en montrant comment le code est transféré vers le microcontrôleur et testé dans l'environnement Proteus.

Par la suite, nous avons détaillé les composants utilisés pour contrôler une matrice LED 8x8 et un écran LCD, illustrant à travers des figures le bon fonctionnement du système après simulation. Enfin, nous avons souligné la faisabilité du projet en termes de coût et de disponibilité des composants, rendant ce projet accessible pour des applications éducatives ou interactives.

Cette phase pratique démontre l'efficacité de l'intégration entre le matériel et le logiciel, validant ainsi les choix technologiques et la logique de programmation mise en place dans les chapitres précédents.

Conclusion Générale

Le but principal de notre travail concernait l'étude et la réalisation d'un affichage lumineux en utilisant un microcontrôleur Pic16F877. Les affichages lumineux sont très utilisés dans nos jours, notamment dans les supermarchés les Voitures……ect

L'idée du format pic est bien connue et répandue à toute les personnes ayant même une connaissance superficielle de cette spécialité, mais le principe de fonctionnement n'est pas connu de tout dans le monde, et pour cette raison nous avons consacré un chapitre (chapitre 1) entier à parler uniquement sur le pic, et plus précisément (pic16F877).

D'après avoir appris le pic, nous passons au deuxième point, qui est l'axe autour duquel s'articule le projet, à savoir l'apprentissage des différents types d'éclairage, dont nous avons parlé dans le deuxième chapitre.

Dans le troisième chapitre, nous avons abordé la partie programmation qui permet de contrôler le pic et de connaître son programme spécial, Mikro c pro, et le programme de simulation, qui joue un rôle fondamental dans de nombreux projets, y compris celui sur lequel nous avons travaillé. Ensuite, dans le quatrième chapitre, nous avons montré les outils utilisés dans le projet et à la fin du chapitre, nous avons montré des photos du projet avant et après la connexion.

En fin de compte, nous disons que tout ce travail a été utile en termes d'informations précieuses. C'est une expérience nouvelle, surtout dans la partie réalisation et les difficultés auxquelles nous avons été confrontés.

Bibliographie

[1].Mémoire de fin d'étude « Microcontrôleurs optimisés pour les enregistreurs de données » (GHALI Mohamed Amine et BENBRAHIM Amel, université IBN KHALDOUN - TIARET)

[2].website :(https://www.techno-science.net/definition/6737.)

[3]. Cours « Principe sur microcontroleurs » (Y. RKHISSI KAMMOUN)/https://isetn.rnu.tn/archives/fr/images/documents/cours/microcontroleurs_yosra.pdf

[4]. website :(https://www.computerhistory.org/revolution/digital-logic/12/284/1564)

[5]. website :(https://fr.panasystech.com/info/microcontroller-history-36102814.html)

[6]. https://slideplayer.es/slide/18064596/

[7]. Mémoire de fin d'étude «Conception d'une carte d'acquisition pour un contrôle automatique de température à base de microcontrôleur PIC 16F877A»BENTAHAR Charaf Eddine/BENAMARA Mohamed El Mahdi 2021/2022 université-Ain-Temouchent- Belhadj Bouchaib

[8]. https://elearning-facsci.univ-annaba.dz/course/view.php?id=784

[9]. Mémoire de fin d'étude « éclairage public et mesure de température a base de pic 16f877»Taboui Mouna 2009/2010 université de Tunis

[10]. Mémoire de fin d'étude « Régulation automatique du fonctionnement d'un store àbase du PIC 16F877a»DOURI Anes/YAMZI Ali 2011/2012 université SAAD DAHLAB de BLIDA

[11]. Mémoire de fin d'étude «Commande de l'éclairage public et mesure de la température à base de PIC18F4550 » Smail Soufiane Université Mohamed Khider Biskra

[12]. S.LAIMACHE. F.MOUHOUBI."Réalisation d'un régulateur numérique PI pour commander un moteur à courant continu à base d'un microcontrôleur PIC16f877a", mémoire master, Université Dr YAHIA FARES de Médéa, 2015/2016.

[13]. Mémoire de fin d'étude « ETUDE ET SIMULATION D'UN REGULATEUR PI NUMERIQUE ET IMPLIMENTATION DANS UN MICROCONTROLEUR PIC16F877A POUR La Commande de vitess D'UN MOTEUR A COURANT CONTINU » (ABD ALLAH ALHADJ HAYET TERRAS NAWEL, université DR. YAHIA FARES DE MEDEA,2017-2018)

[14]. Mémoire de fin d'étude « Etude et réalisation d'une carte de commande PWM, PFM universelle » (GUELLIL Assam, Université Mouloud MAMMERI de Tizi-Ouzou,2009-2010)

[15]. figure 12 dans le website:https://www.researchgate.net/figure/FIGURE-NO-242-Internal-architecture-3-PIN-DIAGRAM-OF-PIC-16F877_fig1_356716973

[16]. figure 8 dans le website :https://embededstudio.com/product/20mhz-crystal-oscillator/

[17]. website (https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-embarquee/743_gestion-des-entrees-sorties/3424_afficheurs-7-segments/)

[18]. Cour de module Composants optoélectroniques (Dr. Makhloufi Mohamed Tahar-University of BATNA 2)

[19]. website (https://www.pcb-hero.com/blogs/lickys-column/7-segment-displays)

[20]. website (https://wiki.3rail.nl/index.php?title=Bestand:LED-symbool.png)

[21]. website (https://circuitglobe.com/difference-between-led-and-lcd.html)

[22]. website (https://www.positron-libre.com/cours/electronique/diode/led/led.php)

[23]. website (https://www.microscale.net/products/dot-matrix-display-with-max7219-driver-ic)

[24]. website (https://passionelectronique.fr/matrice-led-max7219-arduino/)

[25]. website (https://www.robotique.site/tutoriel/afficheur-lcd-1602-avec-i2c/)

[26]. website (https://www.robotique.site/tutoriel/afficheur-lcd-1602-avec-i2c/)

[27]. website (https://duino4projects.com/en/digital-clock-using-arduino-uno-without-rtc-module/)

[28]. Mémoire de fin d'étude «Commande de l'éclairage public et mesure de la température à base de pic 16f877A » AMRANI Ayoub/HARBOUCHE Nour El Houda Université IBN-KHALDOUN DE TIARET

[29]. Site web(https://www.scribd.com/doc/36434413/Programmation-en-C-du-Microcontrolleur-PIC16F877),chapitre 2"Programmation en C et application"

[30].article pdf(https://ww1.microchip.com/downloads/en/DeviceDoc/51795B.pdf)

[31].site web(https://rogers5880.com/2024/04/25/pickit-3-pinout-everything-you-need-to-know)

[32].site web(https://www.raspberry.ma/programmation-universelle-avec-pickit3-et-pic-test-et-avis)

[33].site web (https://www.rytronics.in/product/pic-icd2-pickit2-pickit3-universal-programming-adapter-programmer-board/)