République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur Et de La Recherche Scientifique

N°d'ordre :

N° de série:

Université de Ghardaïa

Faculté des Sciences et Technologie

Département des Sciences et Technologie

Projet de fin d'étude présenté en vue de l'obtention du diplôme de

LICENCE

Domaine : Science et Technologie

Filière : Génie électrique

Spécialité : Maintenance en instrumentation industrielle

THEME:

Architecture et Programmation du microcontrôleur «PIC16F876»

PAR :

BENZEKRI EL rabia

DOUDOU Hammou

KECHAR Smail

Jury:

M^{r}: BENCHAABANE Achour	Maitre Assistant A Univ. Ghardaia	**Encadreur**
M^{r}: ……	Maitre Assistant A Univ. Ghardaia	**Examinateur**

ANNEE UNIVERSITAIRE: 2013/2014

Remerciements

Nous remercions avant tout ALLAH le tout puissant qui m'a donné les capacités physiques et intellectuelles nécessaires à la réalisation de ce projet de fin d'études.

Nous tenons à exprimer nos remerciements les plus sincères envers
Monsieur : Benchaban Achour,

Nos enseignants, pour leur encadrement, leur disponibilité, leur compréhension et leur aide précieuse.

Nous adressons aussi notre reconnaissance à tous ceux qui nous ont apporté leur aide précieuse.

Nous remercions le président et membres du jury pour bien vouloir évaluer notre travail

(Projet de fin de l'étude).

Merci infiniment

Hammou , Rabie et smail

Résume :

A microcontroller is an object capable of processing, storing and returning of information.
It is in particular constituted by a microprocessor. This allows the central portion the information processing.
A microprocessor is an integrated VLSI (**V**ery **L**arge **S**cale **I**ntegrated circuit) circuit.
It includes thousands or millions of transistors. They perform the following logical functions:

- Arithmetic and logical
- Storing
- Communication interface

the objective of the Project is to be understood through the microcontroller PIC16F876 microcontroller operation in general. For this, the general architecture of the microcontroller will first be addressed (to enumerate and identify individual components of a microcontroller). then the architecture of the PIC 16F876 will be detailed. Finally, the assembler programming (low level language close to machine language) and c' sera studied and therefore a simulated example Protuse ISIS.

Résumé :

Un microcontrôleur est un objet capable de traiter, de stocker et de restituer de l'information.

Il est en particulier constitué d'un microprocesseur. C'est la partie centrale qui permet

Le traitement de l'information.

Un microprocesseur est un circuit intégré VLSI (**V**ery **L**arge **S**cale **I**ntegrated circuit).

Il comprend des milliers voir des millions de transistors. Ils réalisent les fonctions logiques suivantes :

- Calculs arithmétiques et logiques
- Mémorisation
- Interface de communication

L'objectif de se Projet est de comprendre à travers le microcontrôleur PIC16F876 le fonctionnement des microcontrôleurs en général. Pour cela, l'architecture générale de microcontrôleur sera d'abord abordée (pour énumérer et définir les différents constituants d'un microcontrôleur). Ensuite l'architecture du PIC 16F876 sera détaillée. Enfin, la programmation en assembleur (langage bas niveau, proche du langage machine) et en 'c'sera étudiée et donc la simulation d'un exemple avec Porteuse ISIS.

ملخص :

المتحكم هو كائن قادر على معالجة وتخزين وإعادة المعلومات و على وجه الخصوص تشكيلها من قبل المعالج و هو الجزء المركزي إذ يسمح لنا بالمعالجة.

المعالج هو الدارة المتكاملة VLSI والمتكونة بداخل المتحكم (الدارات المتكاملة على نطاق واسع للغاية.)

فإنه يشمل الآلاف أو الملايين من الترانزستورات ويقوم بوظائف المنطق التالية :

- الحسابية والمنطقية
- تخزين
- واجهة الاتصالات

الهدف من هذا المشروع هو أن نفهم المتحكم PIC16F876 من خلال عمله بشكل عام و ذالك بدراسة الهندسة الداخلية وهذا بالطبع(لكي نوضح ونعرف المكونات المختلفة بداخل المتحكمات عامة) ثم على وجه التحديد الهندسة الداخلية للمتحكم PIC16F876 وفي النهاية, البرمجة تكون باستعمال اللغة Assembleur وهي لغة القاعدة البيانية وهي تشبه إلى حد قريب لغة الآلات أو باستعمال لغة البرمجة C ,وإذا نقوم باستعمال برنامج المحاكاة الشهير ب:SISI eProteus و هذا لتقديم صورة شبه حقيقية لمثال معين .

Liste des Figures :

Liste des tableaux :

Sommaire

Chapitre III : Programmation de microcontrôleur [pic 16F876

Introduction Générale

Les progrès technologiques continus dans le domaine des circuits intégrés ont permis la réduction des coûts et de la consommation. Les circuits intégrés spécifiques ont permis une réduction de la taille des systèmes numériques ainsi que la réalisation de circuits de plus en plus complexes, tout en améliorant leurs performances et leur fiabilité. Aujourd'hui les techniques de traitement numérique occupent une place majeure dans tous les systèmes électroniques modernes grand public, professionnels ou militaires. De plus, les techniques de réalisation de circuits spécifiques, tant dans les aspects matériels (composants reprogrammables, circuits pré caractérisés et bibliothèques de macro fonctions) que dans les aspects logiciels (placement-routage,synthèse logique) font désormais de la microélectronique une des bases indispensables pour la réalisation de systèmes numériques performants. Elle impose néanmoins une méthodologie de développement très structurée.

Les circuits PICs sont certainement les circuits reprogrammables ayant les plus succès. Ce sont des circuits entièrement configurables par programmation qui permettent d'implanter physiquement, par simple programmation, n'importe quelle fonction logique. De plus, ils ne sont pas limités à un mode de traitement séquentiel de l'information comme avec les microprocesseurs et en cas d'erreur, ils sont reprogrammables électriquement sans avoir à extraire le composant de son environnement.

Chapitre I

Les Circuits Programmables

1.1. Introduction :

L'histoire des circuits programmables commence, à peu de chose près, avec la décennie quatre-vingt. A la fin des années soixante-dix le monde des circuits numériques se répartit schématiquement en quatre grands groupes :

- Les fonctions standard sont utilisées pour les applications réalisées en logique câblée. Les catalogues TTL et CMOS présentent plusieurs centaines de fonctions d'usage général, sous forme de circuits intégrés à grande échelle.
- Les microprocesseurs, désormais d'usage courant, et sont omniprésents dans les applications industrielles.
- Dans des applications trop complexes pour être raisonnablement traitées en logique câblée traditionnelle, et trop rapides pour avoir une solution à base de microprocesseurs, on utilise des séquenceurs micro programmés.
- Quand les volumes de production importants le justifient, les circuits intégrés spécifiques (ASICS) offrent une alternative aux cartes câblées classiques.

Le développement des microprocesseurs stimule une évolution rapide des technologies de réalisation des mémoires à semi-conducteurs ; les circuits logiques programmables ont hérité directement des mémoires pour ce qui concerne les aspects technologiques. Leurs architectures internes sont, en revanche, très différentes. Il n'est donc pas surprenant que le premier fabriquant de circuits programmables ait été un fabriquant de mémoires (MMI, monolithic memories inc). Notons, pour la petite histoire, que cette société a « fusionné1» avec l'un des leaders des fabriquants de processeurs rapides, processeurs micro programmés, processeurs spécialisés dans le traitement de signal, processeurs RISCs, AMD (advanced micro devices).

Ce texte est consacré à un tour d'horizon général au cours duquel nous tenterons de donner au lecteur une vision d'ensemble de ce que sont les membres de la famille (nombreuse) des circuits programmables.

Les aspects structurels internes ne seront abordés que dans la mesure où ils éclairent la compréhension du fonctionnement. L'utilisateur d'un circuit a principalement besoin de bien dominer son architecture, les technologies de fabrication appartiennent au domaine du fondeur.

1.2. Les microcontrôleurs :

1.2.1. Définition :

Les microcontrôleurs sont des circuits intégrés programmables. Ils regroupent, au sein d'une même puce, les différents éléments que l'on trouve habituellement dans l'unité centrale d'un ordinateur personnel. En particulier, ils contiennent une unité de calcul, différentes mémoires volatiles et non volatiles, ainsi qu'un ensemble « d'interfaces internes » facilitant la communication avec le monde extérieur. Ils sont conçus pour être programmés par un ordinateur, puis placés dans un circuit électronique dans lequel ils effectuent un travail plus ou moins complexe. Après programmation, ils peuvent fonctionner indépendamment de tout ordinateur.
Leurs principaux domaines d'application sont l'industrie pour la fabrication de machines outils, la domotique (machines à café, appareils électroménagers, etc.), et l'électronique grand public.

1.2.2. Utilisation d'un microcontrôleur

Toutes les solutions à base de composants programmables ont pour but de réduire le nombre de composants sur le circuit électronique et donc fiabiliser le circuit.

Le microcontrôleur est en concurrence avec d'autres technologies .

Suivants les applications : 3 types de technologies

- Logique câblée
  - très rapide, fonctions réalisées par une voie matérielle
  - non programmable, peu économique quand l'application est complexe peu de souplesse : durée d'étude prohibitif et circuit difficilement modifiable
- Réseaux de logique programmables (PAL, LCA,..)
  - rapide, adapté au traitement de signaux complexes
  - prix élevé et langage de programmation non standard
- Les µ processeurs
  - Grande souplesse : fonctions sont réalisées par voie logicielle puissance de calcul, langage évolué.
  - nombre important de composant à réunir, solution onéreuse à retenir

Si la fonction à réaliser est simple ⇒ une logique câblée.
Si le nombre d'unités à réaliser est très important ⇒ circuits intégrés dédié en logique câblée pour les fonctions simples.

1.2.3. Les avantage et les inconvénients des microcontrôleurs :

1.2.3.1. Les avantage :

- Diminution de l'encombrement du matériel et du circuit imprimé
- Simplification du tracé du circuit imprimé (plus besoin de tracer de bus !)

- Augmentation de la fiabilité du système :
  - nombre de composants
  - connexions composants/supports et composant circuit imprimé
- Intégration en technologie MOS, CMOS, ou HCMOS diminution de la consommation.
- Le microcontrôleur contribue à réduire les coûts à plusieurs niveaux:
  - moins cher que les composants qu'il remplace
  - Diminution des coûts de main d'œuvre (conception et montage)
- Environnement de programmation et de simulation évolués

1.2.3.2. Les inconvénients :

- le microcontrôleur est souvent surdimensionné devant les besoins de l'application
- Investissement dans les outils de développement
- Écrire les programmes, les tester et tester leur mise en place sur le matériel qui entoure le microcontrôleur.
- Incompatibilité possible des outils de développement pour des microcontrôleurs de même marque.
- Les microcontrôleurs les plus intégrés et les moins coûteux sont ceux disposant de ROM programmables par masque

Fabrication uniquement en grande série >1000

Défaut relatif car il existe maintenant systématique des versions OTPROM un peu plus chère **[01]**.

1.3. Les types des circuits programmables :

1.3.1. Les PICs de Microchip :

Les PICs sont des microcontrôleurs à architecture RISC (Reduce Instructions Construction Set), ou encore composant à jeu d'instructions réduit. L'avantage est que plus on réduit le nombre d'instructions, plus leur décodage sera rapide ce qui augmente la vitesse de fonctionnement du microcontrôleur.

La famille des PICs est subdivisée en 3 grandes familles : 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), et la famille High-End, qui utilise des mots de 16 bits. Les PICs sont des composants STATIQUES, Ils peuvent fonctionner avec des fréquences d'horloge allant du continu jusqu'à une fréquence max spécifique à chaque circuit. Un PIC16F876-04 peut fonctionner avec une horloge allant du continu jusqu'à 4 MHz. Nous nous limiterons dans ce

document à la famille Mid-Range et particulièrement au PIC 16F876/877, sachant que si on a tout assimilé, on pourra facilement passer à une autre famille, et même à un autre microcontrôleur.

Microchip : c'est une société Américaine technologique à mis au point dans les années 90 un microcontrôleur CMOS: le PIC, ce composant encore très utilisé à l'heure actuelle, est un compromis entre simplicité d'emploi, rapidité et prix de revient.

1.3.2. caractéristques des PICs 16F87X-BB :

Que signifie 16F87X-BB :

16 : le circuit appartient à la série Mid-Line.

F : la mémoire programme est de type FLASH.

87X : type.

BB : Quartz à BBMHz. **[02]**

1.4. DSP (Digital Signal Processor) :

1.4.1. Introduction Aux DSP Orientés Applications Industrielles :

Depuis plusieurs années, le traitement numérique du signal est une technique en plein essor. Cette technique s'appuie sur plusieurs disciplines, citons les principales :

- L'électronique analogique et numérique (préparations, conditionnements des signaux, conversions numériques ↔ analogiques).
- Les microprocesseurs (classiques ou dédiés au traitement du signal).
- L'informatique (algorithmes, systèmes de développements, exploitations),
- Les mathématiques du signal (traitements du signal).

Parmi ces disciplines, ce chapitre est plus précisément une description des processeurs de traitements des signaux, plus communément désignés par l'acronyme Anglais DSP (Digital Signal Processor).

Les domaines d'applications du traitement numérique du signal sont nombreux et variés (traitements du son, de l'image, synthèse et reconnaissance vocale, analyse, compression de données, télécommunications, automatisme, etc.). Chacun de ces domaines nécessite un système de traitement numérique, dont le cœur est un (parfois plusieurs) DSP ayant une puissance de traitement adaptée, pour un coût économique approprié.

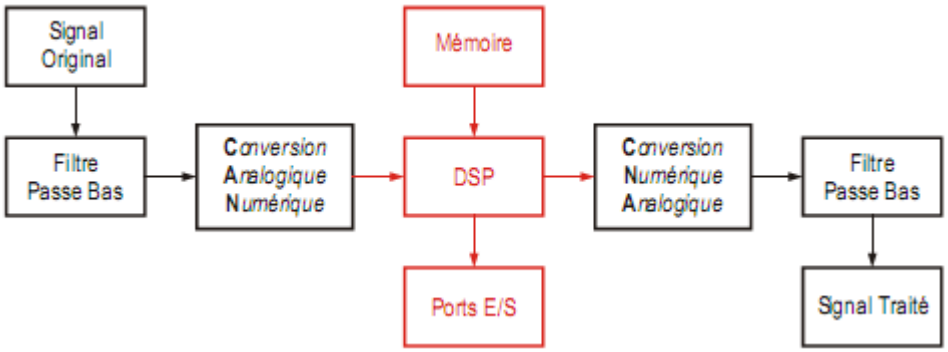

Figure I .1 : Chaîne complète typique d'un système de traitement numérique du signal

Un DSP est un type particulier de microprocesseur. Il se caractérise par le fait qu'il intègre un ensemble de fonctions spéciales. Ces fonctions sont destinées à le rendre particulièrement performant dans le domaine du traitement numérique du signal. Comme un microprocesseur classique, un DSP est mis en œuvre en lui associant de la mémoire (RAM, ROM) et des périphériques. Un DSP typique a plutôt vocation à servir dans des systèmes de traitements autonomes. Il se présente donc généralement sous la forme d'un microcontrôleur intégrant, selon les marques et les gammes des constructeurs, de la mémoire, des timers, des ports sérient synchrones rapides, des contrôleurs DMA, des ports d'E/S divers.

1.4.2. Approche technologique :

La Figure 1-2 montre la vitesse d'un composant en fonction de sa « performance », c'est-à-dire de l'adaptation à des besoins spécifiques de l'électronique. On voit la place privilégiée du DSP par opposition à celle du microprocesseur, d'usage plus général.

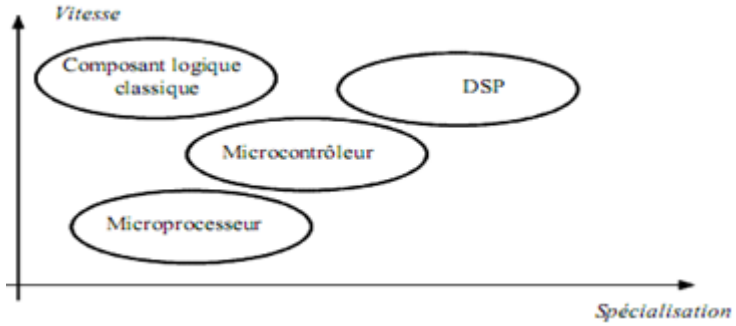

Figure 02 : Place du DSP vis-à-vis des autres processeurs

1.4.3. Classification des DSP :

Il est impossible d'effectuer une classification « définitive » des DSP, car chaque constructeur met sur le marché tous les ans un nouveau composant qui surclasse les anciens ou les concurrents par la puissance de calcul, la rapidité (gestion du pipeline et fréquence d'Horloge), le nombre de registres, de Timers, de ports série…

1.4.4. Puissance de calcul d'un DSP :

C'est un autre critère de classification des DSP. Cette puissance de calcul dépend de la rapidité de l'exécution des instructions, et donc de l'horloge. Dans un DSP, le MAC (multiplicateur et accumulateur) calcule le produit de deux entrées codées sur N bits, dans un temps « record » de 7ns à 150ns. Un cycle d'horloge !. La multiplication est obtenue de manière asynchrone. Le résultat est chargé dans un accumulateur à 2 x N bits. L'utilisateur choisit de garder seulement les N bits de poids fort en simple précision, et effectue alors une troncature ou alors l'ensemble du résultat en double précision. Bien entendu, si le DSP est à virgule flottante, l'effet de la troncature est moins gênant.

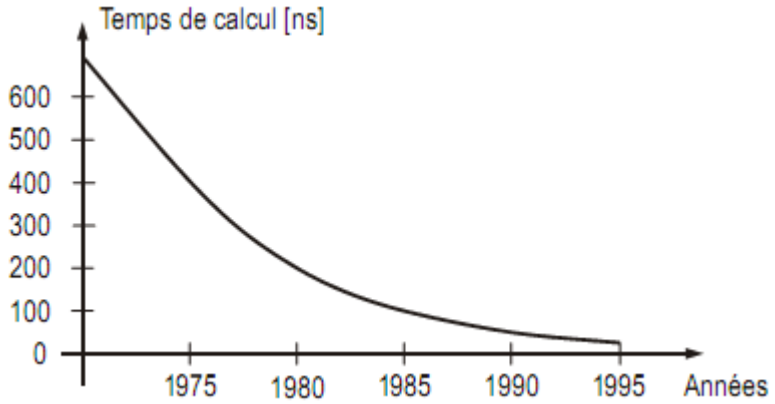

Figure I .3 : Evolution du temps d'exécution d'une opération MAC selon Texas Instruments

1.4.5. Structure logicielle de développement :

Les deux principales méthodes pour écrire un programme DSP consistent à utiliser un assembleur dédié ou un langage de haut niveau. Un langage de haut niveau comme le langage C présente l'avantage d'être connu par la plupart des ingénieurs amenés à travailler dans le domaine du traitement numérique du signal. Un programme DSP écrit en langage C peut donc être compris relativement facilement par un grand nombre de personnes, sans qu'elles aient besoin de connaître précisément le DSP cible. De plus, la portabilité du langage C permet de d'utiliser un programme sur des DSP fabriqués par différents constructeurs.

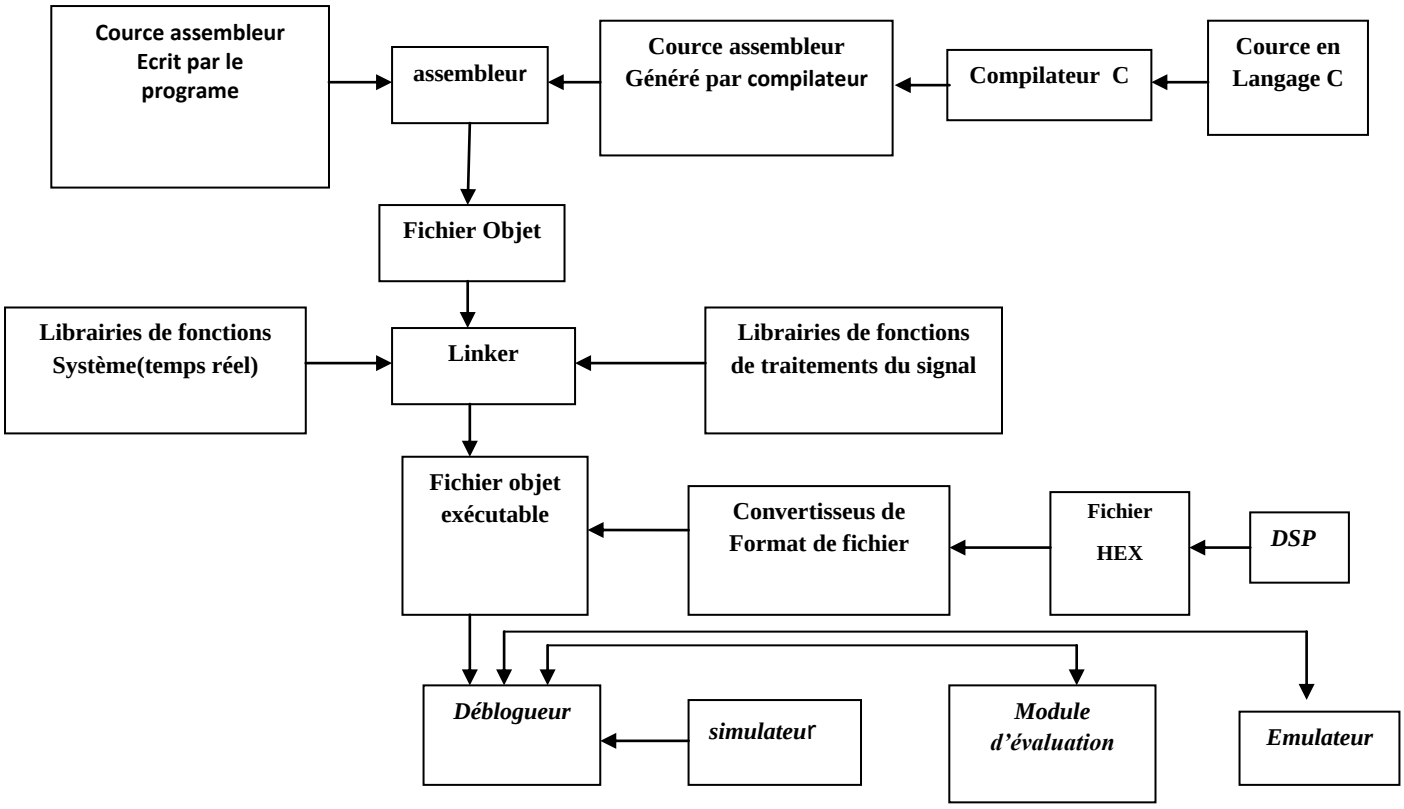

Figure I.4 : Organigramme d'un système de développement de logiciel pour DSP

L'utilisation d'un assembleur pour programmer un DSP n'en demeure pas moins intéressante. plus que pour un microprocesseur classique, les performances de traitement sont cruciales, et l'assembleur est le seul langage permettant d'utiliser totalement les possibilités spécifiques de tel ou tel DSP. Par exemple, un algorithme de codage LPC est ainsi estimé 1,5 fois plus rapide quand il est implémenté en assembleur plutôt qu'en C. Même lorsqu'ils sont Spécialement optimisés pour le DSP cible, les compilateurs C ne permettent pas de générer un programme ayant les performances d'un code bien écrit en assembleur par un développeur confirmé.

La souplesse du système de développement permet d'ajouter facilement des portions de programmes assembleurs (assemblage « en ligne », ou librairie de fonctions assembleurs) à un logiciel écrit en C. Ce mélange est recommandé par les constructeurs, ainsi la plupart des exemples de traitements numériques sont donnés en assembleur, alors que les exemples de mise en œuvre du DSP sont donnés en C. Les outils de débogage acceptent indifféremment l'un, l'autre, ou le mélange des deux langages.

Certaines sociétés indépendantes des constructeurs fournissent des librairies de fonctions de traitements numériques du signal prêtes à l'emploi. Le langage C facilite l'intégration des librairies, voire même la mise en œuvre de véritables systèmes d'exploitation temps réels et multitâche, par exemple le système SPOX de Spectron Microsystems. Bien qu'étant le plus répandu, le langage C n'est pas le seul utilisable pour programmer un DSP, il existe quelques rares compilateurs ADA, FORTRAN et PASCAL pour DSP. **[03]**

1.5. FPGA (Field Programmable Gate Array):

Les FPGAs (Field Programmable Gate Array) sont des circuits intégrés qui contiennent des blocs de logique configurables (programmables), ainsi que des interconnexions configurables entre ces blocs .La configuration du circuit, c'est-à-dire la programmation de sa fonction, se fait sur place (Field Programmable), sans envoi chez un fabricant . Les FPGA se placent à mi-chemin entre lesPLD et les ASIC. Les avantages par rapport aux ASIC sont:

- coût de développement plus bas: les dépenses en NRE (nonrecurring engineering) sont moins importantes
- modifications plus simples à réaliser
- time-to-market plus court

cibles						
processeurs				circuits		
généralistes		spécifiques		programmables		dédiés ASIC
classiques	μ-contrôleurs	DSP	ASIP	FPGA	ARD	ASIC
Pentium PowerPC Alpha MIPS . . .	68705 68HC11 80C51 AVR . . .	ADSP-21xx TMS320xx DSP56xx . . .		Xilinx Altera Actel . . .		*full custom* *standard cell* *gate array* . . .

Tableau 01: Les Cibles technologiques.

• **ARD:** architecture reconfigurable dynamiquement

• **FPGA:** field programmable gate array_ Réseau de portes programmables

• **DSP:** digital signal processor_ Processeur orienté vers le traitement du signal

• **ASIC:** application specific integrated circuit_ Circuit intégré conçu à la demande

• **ASIP:** application specific integrated processor_processeur intégré conçu à la demande. **[04]**

5.1. Différents domaines d'applications des FPGA :

- Informatique : Périphériques spécialisés
- Machinerie industrielle : Contrôleur pour machines outils, balances, grues, presses, etc.

  Traitement d'images
- Télécommunications : Filtrage et Unité centrale (PABX)
- Instrumentation : Équipement médical
- Transport : Contrôle "Fly-by-wire"

  Prototypage.**[05]**

1.5.2. Circuits programmables et applications particulières :

Compagnie	Produits	Technologie	Description
Altera	SAM EPS4000	EPROM	Microséquenceur[1], Générateur d'interruptions synchrone (32 I/O)
Cypress	CY7C300	EPROM	MSA programmable, MSA à haute cadence (26 I/O)
Philip-Signetics	PLUS405	Fusible	Séquenceur logique programmable (24 I/O, 55 MHz)
Aptix, I-Cube	AX1024 IQ160	SRAM	Module d'interconnexions programmable, FPCB, carte programmable
Université de Toronto	FPAA	CMOS 1.2µ (Transconducteurs MOS)	Matrice programmable analogue (Field Programmable Analog Array)

Tableau I.2 : Les déférents types des Circuits programmables FPGA

1.5.3. Les éléments programmables :

Élément programmable	RAM statique	EPROM / EEPROM[1]	Antifusible	
			Diélectrique[2]	ViaLink[3]
Historique	Xilinx	Altera et al.	Actel > TI	QuickLogic
R_{on} (typ.)	~1-2 kohms	~1000 ohms	~500 ohms	~50 ohms
C_{off} (typ.)	~30-50 fF	~15 fF	~5-10 fF	~1 fF
Délai	(appréciable)	Moyen	Court	Court
Dimensions	Très grande (6 T)	Grande	Moyenne (1 T)	Petite (1 T)
Densité				
Programmation	RAM	Charge > grille Déch. par UV / Élec.	12 V, 10 mA, 1 ms > PNP	11 V, 10 mA, 1 ms > M-M

Tableau I.3 : Les caractéristiques des Circuits programmables FPGA [05].

1.5.4. Les Avantages de FPGA:

- Forte modularité
- Rapidité

1.5.5. Les Inconvénients de FPGA :

- Mise en œuvre plus complexe
- Coûts de développement élevé. **[06]**

Conclusion

- Circuits programmables caractérisés
  - Architectures
  - Flot de conception
    - Outils de description et de programmation
- Comment décrire le comportement d'un système électronique en vue de l'implémenter sur un circuit programmable ?
  - Langage de description matériel

Chapitre II
Architecture de microcontrôleur (µC) PIC 16F876

II.1 Introduction :

Les microcontrôleurs PIC utilisent un jeu d'instructions réduit, d'où leur nom d'architecture : RISC (Reduced Instructions Set Computer). Les instructions sont codées sur un nombre réduit de bits, ce qui accélère l'exécution du programme (1 cycle machine par instruction sauf pour les sauts qui requirent 2 cycles). En revanche, le nombre limité d'instructions oblige à se restreindre à des instructions basiques, contrairement aux systèmes d'architecture CISC (Complete Instructions Set Computer) qui proposent plus d'instructions, donc codées sur plus de bits, mais réalisant des traitements plus complexes.

Il existe trois familles de PIC :

- Base-Line : les instructions sont codées sur 12 bits.
- Mid-Line : les instructions sont codées sur 14 bits.
- High-End : les instructions sont codées sur 16 bits.

Ils ne communiquent avec l'extérieur que par des ports d'entrées/sorties (ils ne possèdent pas de bus d'adresses, de bus de données et de bus de contrôle comme la plupart des microprocesseurs).

II.2 L'architecteur :

Les microcontrôleurs ont été conçus sur une architecture dite HARVARD (RISC) et non sur un modèle VON NEUMANN (COMPLEX).

II.2.1 L'architecteur de VON NEUMANN et HARVARD :

• L'architecture VON NEUMANN employée par la plupart des microcontrôleurs actuels (INTEL80XX, Motorola HC05, HC08 et HC11, ou ZILOG Z80) est basée sur un bus de données unique. Celui-ci véhicule les instructions et les données.

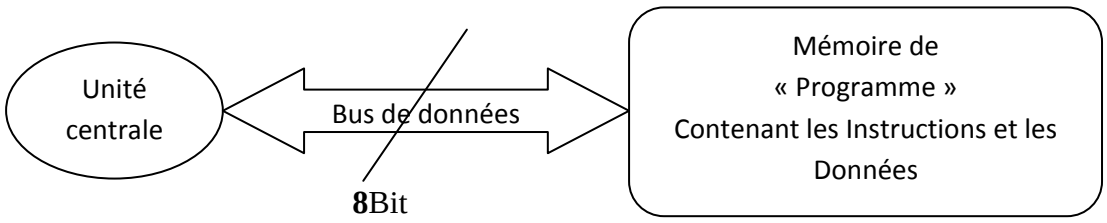

Figure II.1 : Architecture de Von Neumann

• L'architecture HARVARD utilisée par les microcontrôleurs PICS est basée sur deux bus de données. Un bus est utilisé pour les données et un autre pour les instructions.

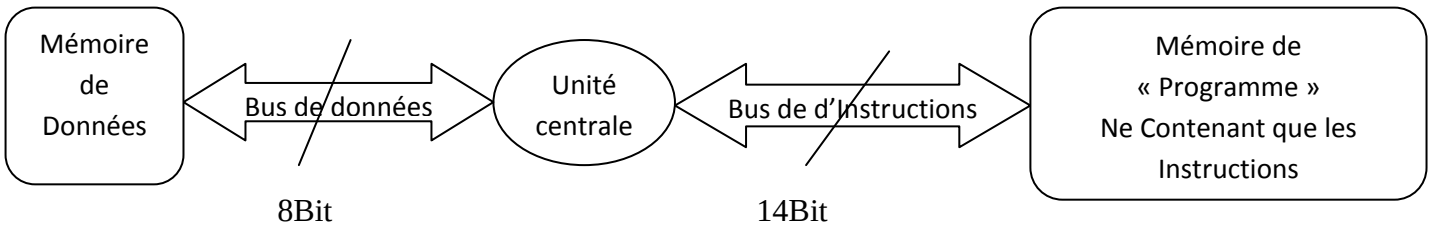

Figure II.2 : Architecture de Harvard

II.2.2. Les avantages et les inconvénients de L'architecture (VON NEUMANN et HARVARD) :

	Architecture VON NEUMANN (MOTOROLA, INTEL, ZILOG, ..)	**Architecture HARVARD (RISC) (MICROCHIP PICs)**
Avantages	- Jeu d'instructions riches. - Accès à la mémoire facile.	- Jeu d'instructions pauvre, mais facile à mémoriser. - Le codage des instructions est facile, chaque instruction est codée sur un mot et dure un cycle machine. - Le code est plus compact.
Inconvénients	- Le temps pour exécuter une instruction est variable. - Le codage des instructions se fait sur plusieurs octets.	- Le jeu d'instruction est très pauvre, par exemple pour effectuer une comparaison il faut faire une soustraction. - Les accès aux registres internes et la mémoire sont très délicats.

Tableau II.1 Les avantages et les inconvénients de L'architecture (VON NEUMANN et HARVARD) :

II.3 Le Choix d'un Microcontrôleur PIC 16F876 :

Pour apprendre, la meilleurs solution est de se faire la main sur du concret. On va donc étudier un vrai microcontrôleur, sachant que ce qui aura été vu sera assez facilement transposable à d'autres PIC. **Le 16F876** est un PIC de la série « Midrange » qui se prête particulièrement bien à la programmation en **C**. **Les PIC** de la série inférieure sont un peu justes en performance et en capacité mémoire pour accueillir un programme issu d'un compilateur C ; mieux vaut les programmes en assembleur. Les gammes supérieures (16 ou 32 bits) supportent sans problème la programmation en C, mais comme se sont des circuits plus complexes (et plus chers), commençons par quelque chose de plus simple et de plus didactique. Le 16F876 (F comme « Flash ») convient

parfaitement : mémoire programme de taille suffisante (8K), nombreux périphériques intégrés, fréquence de fonctionnement jusqu'à 20 MHz.

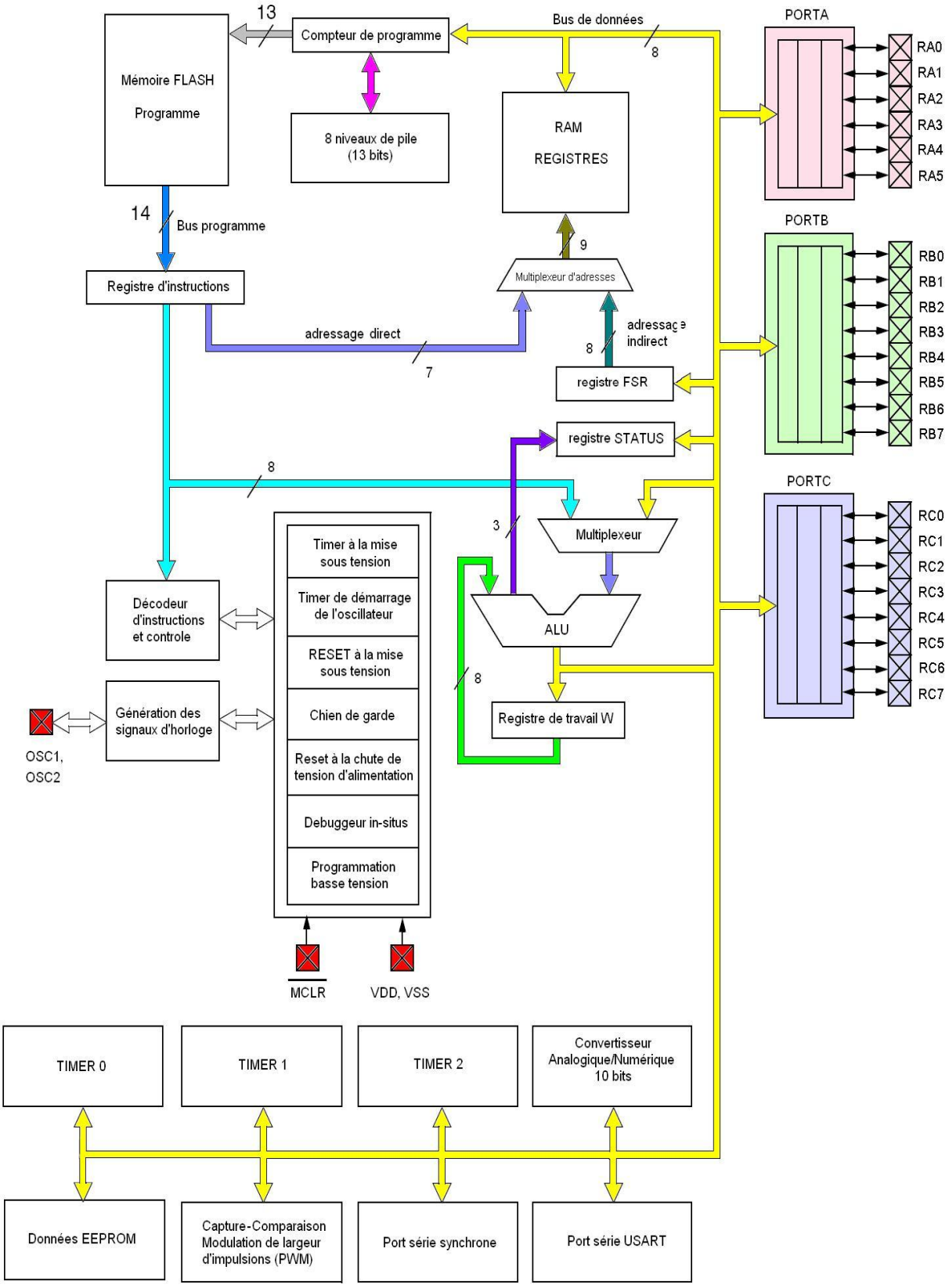

Figure II.3 : Structure interne d'un microcontrôleur Pic 16F876 [07].

II.4. Architecture interne du 16F876 :

II.4.1. Unité Arithmétique et logique ALU :

Comme son nom l'indique, l'unité arithmétique et logique (ALU) effectue au sein du microcontrôleur PIC toutes les opérations arithmétiques et logiques dans un format de données 8 bits (addition, soustraction, comparaison logique, ET logique, etc…).Ces opérations sont effectuées entre le registre de travail W et une opérande Provenant d'un registre de la RAM. Les trois bits de poids faible du registre STATUS sont affectés par le résultat de l'opération effectuée par l'ALU.

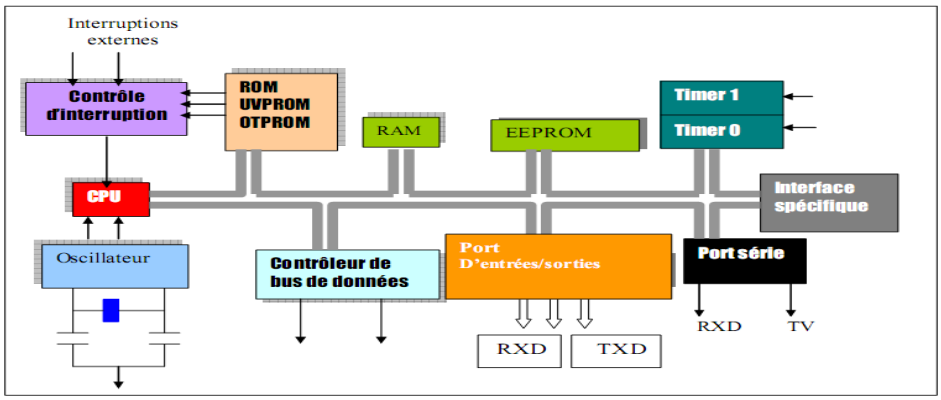

Figure II.4 : Structure interne d'un microcontrôleur

II.4.2. Description des différentes broches du pic 16F876 :

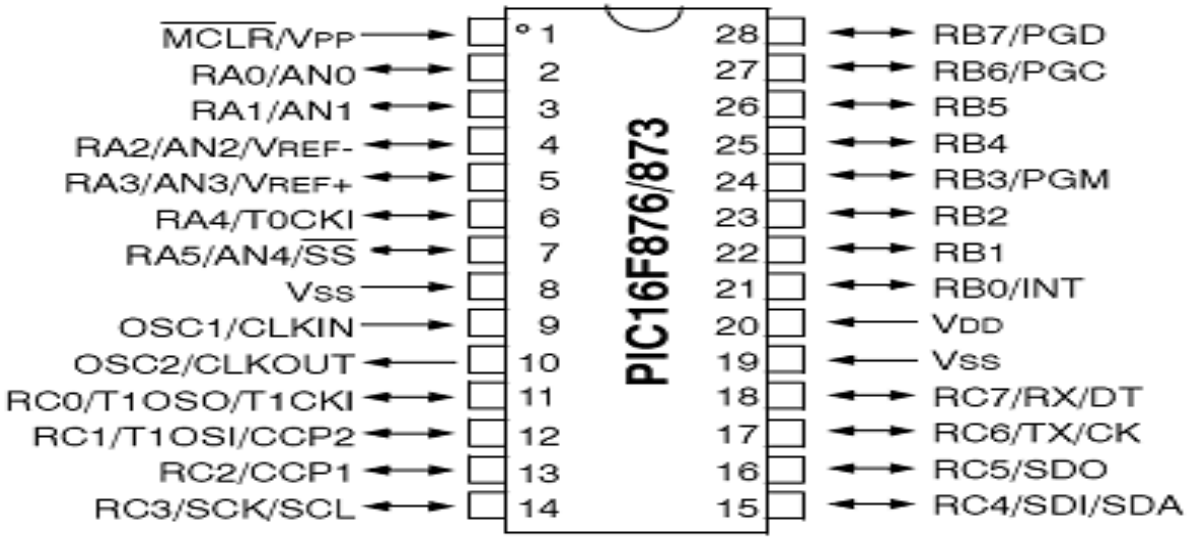

Figure II.5 : Les déférents types des Pins d'un microcontrôleur PIC 16F876

a - MCLR : cette broche sert à initialiser le microcontrôleur PIC :

- à la mise sous tension par un front montant (min 72 ms, max 72 ms +1024 x Tosc) .Cette initialisation est appelée POR (POWER ON RESET) Cette broche peut être simplement reliée à l'alimentation VDD si on n'a pas besoin de RESET externe.

b - OSC1 et OSC2 : ces broches permettent de faire fonctionner l'oscillateur interne du Microcontrôleur PIC de trois façons différentes.

Un quartz ou résonateur céramique : permet d'obtenir une fréquence de fonctionnement très précise (voir document constructeur pour les valeurs des condensateurs C1, C2 et de la résistance RS).

Un oscillateur externe : permet une Synchronisation avec un autre circuit.

- Un simple réseau RC : peut suffire, l'oscillateur est peu précis mais économique.

c - VDD et VSS : broches d'alimentation du circuit (la tension VDD peut être comprise entre 4V à 5,5V).

Les ports du microcontrôleur 16F876 sont couplés à différents modules (Timers (Ports A et C), convertisseur analogique numérique (port A), plusieurs types de liaisons séries (port C). C'est pour cette raison que le nom de ces broches comportent plusieurs désignations (ex : RA0/AN0, broches RA0 couplée au convertisseur A/N).

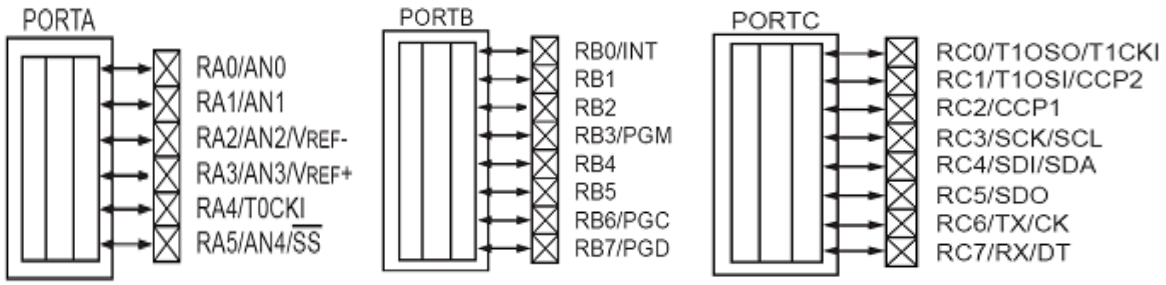

Figure II.6 : Les Trois types des PORTs de microcontrôleur PIC16F876

d - RA0 à RA5 : 6 broches du port A. Chaque ligne peut être configurée individuellement en entrée ou en sortie.

e - RB0 à RB7 : 8 broches du port B. Chaque ligne peut être configurée individuellement en entrée ou en sortie. Les broches RB6/PGC (ProGramming Clock) et RB7/PGD (ProGramming Data) permettent la connexion à un module ICD (In Circuit Debugger, mise au point de programme sur site).

f - RC0 à RC7 : 8 broches du port C.

Chaque ligne peut être configurée individuellement en entrée ou en sortie.

Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	6	I/O	ST	RA4 can also be the clock input to the Timer0 module. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	21	21	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	25	25	I/O	TTL	Interrupt-on-change pin.
RB5	26	26	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	27	27	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	28	28	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
					PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	11	I/O	ST	RC0 can also be the Timer1 oscillator output or Timer1 clock input.
RC1/T1OSI/CCP2	12	12	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	13	13	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	14	14	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	16	16	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	17	17	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	18	18	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
VSS	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	20	P	—	Positive supply for logic and I/O pins.

Tableau II.1: Extrait du document constructeur décrivant le rôle des broches

II.5. Mémoires et registres :

Comme les microcontrôleurs PIC utilisent un bus pour les instructions et un bus pour les données, il faut considérer deux plans mémoire l'un pour les instructions et l'autre pour les données ainsi que les registres internes.

II.5.1. Plan Mémoire pour les instructions (code programme) :

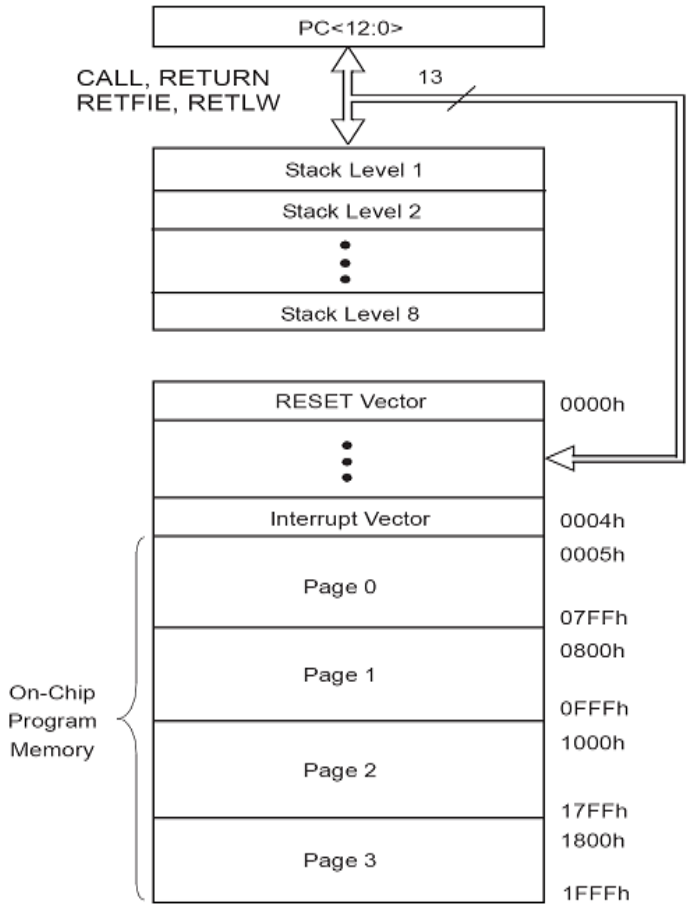

Figure II.7 : Le plan mémoire est linéaire.

Les adresses vont de **$0000** à **$1FFF (8 k mots de 14 bits),** par page de **2 K mots.** On peut remarquer, le vecteur d'initialisation (de reset) est figé à l'adresse **$0000.** Les microcontrôleurs PIC n'ont qu'un seul vecteur d'interruption à l'adresse **$0004.** Lors d'une demande d'interruption, le sous programme associé doit déterminer quel périphérique a demandé cette interruption. Les piles utilisées par les sous programmes ne sont pas implantées en mémoire de données comme dans le cas d'un microcontrôleur classique, mais dans la mémoire programme. Elles sont utilisées lors d'appels de sous programmes, on ne peut pas imbriquer plus de **8** sous programmes (ce qui est déjà beaucoup).

II.5.2. Plan Mémoire des données et des registres internes :

Le plan mémoire des données et des registres internes est découpé en 4 zones ou Banks de 128 octets. Pour accéder à une zone, il faut positionner les bits RP0 (bit n°5) et RP1 (bit n°6) du registre STATUS.

RP1 : RP0	Zone sélectionnée (BANK)
0 0	De 00h à 7Fh : BANK 0
0 1	De 80h à FFh : BANK 1
1 0	De 100h à 17Fh : BANK 2
1 1	De 180h à 1FFh : BANK 3

Tableau II.3: Les déférant BANKs de registre interne

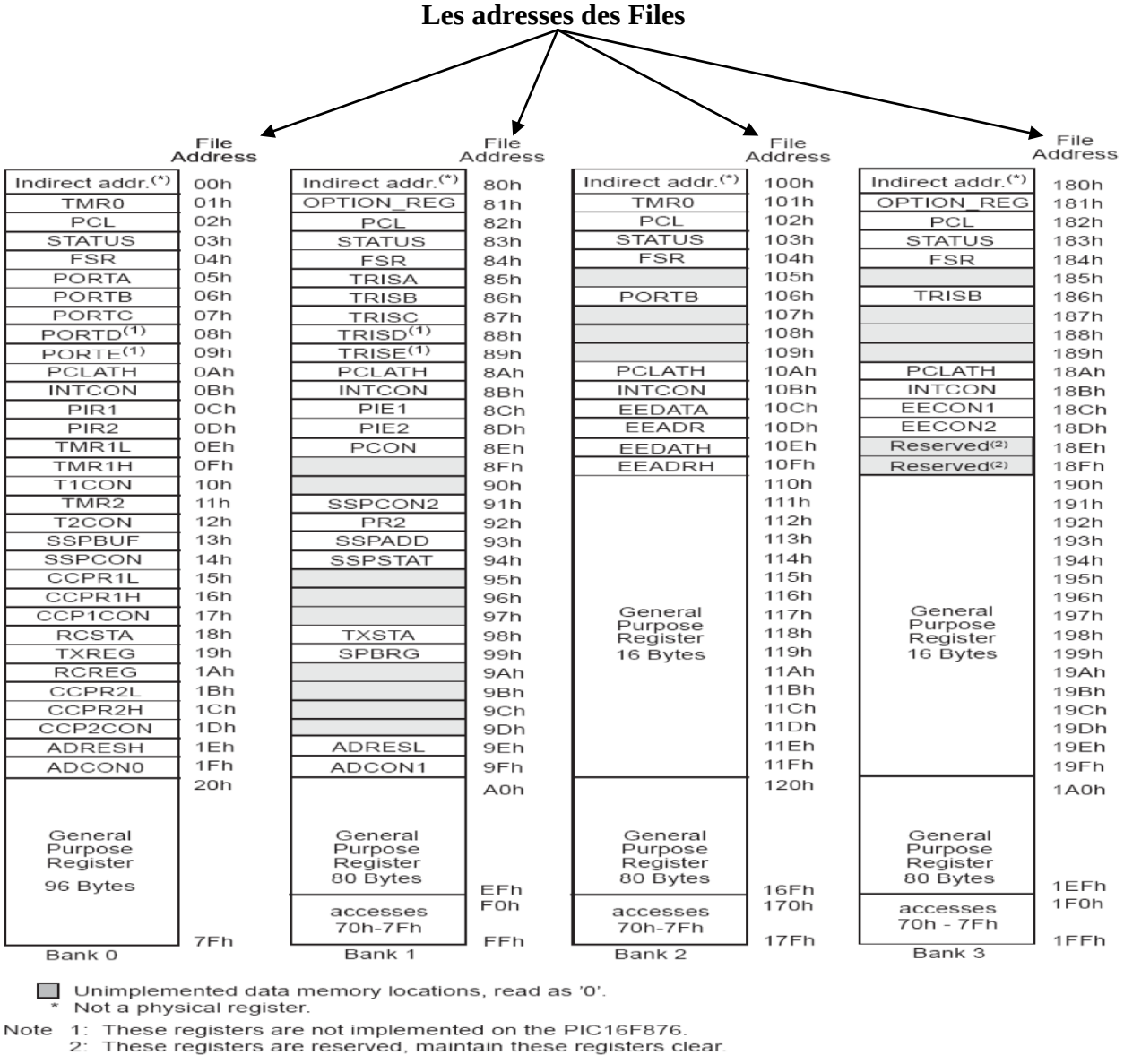

Figure II.8: Plan Mémoire des données et des registres internes[07]

II .6. Conclusion :

Avec ce deuxième chapitre nous avons vu simplement la plupart des constituants internes d'un **PIC**. Le fonctionnement exact de certain bloc tel que le TIMER, la mémoire **EPROM**, le port d'entrées - sorties. Le temps est donc venu de s'intéresser un peu au " hard " du PIC 16F876, nous verrons dans la prochaine partie le brochage ainsi que les conditions de RESET et l'horloge système. Les ports d'Entrées-Sorties un peu plus en détail ainsi que certaines instructions permettant de configurer le microcontrôleur. Nous allons aborder dans la prochaine leçon les trente cinq instructions de base du PIC 16F876 avant de commencer à écrire un premier programme.

Chapitre III

Programmation de microcontrôleur [PIC 16F876]

III.1 Introduction :

La programmation des PIC se fait par le langage assembleur qui est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain. Les combinaisons de bits du langage machine sont représentées par des symboles dits « mnémoniques » (du grec mnêmonikos, relatif à la mémoire), c'est-à-dire faciles à retenir.Le programme assembleur convertit ces mnémoniques en langage machine en vue de créer par exemple un fichier exécutable. Le développement des environnements de programmation, nous a permis de voir naitre de nouveaux compilateurs qui permettent de programmer avec les langages haut niveau tels que le C, PASCAL, BASIC etc.…

Ces environnements comportent aussi des bibliothèques qui permettent de faciliter le développement. Il existe plusieurs outils de développement, les uns sont gratuits, les autres sont payants. Dans notre recherche de l'outil que nous allons utiliser pour programmer notre PIC, nous avons optés pour le langage C. Ce choix est à la fois un choix personnel et un choix technologique. D'une part le langage C est utilisé dans différents systèmes et domaines de développement, ce qui Nous permettra une évolution future, d'autre part le langage C est l'un des langages les plus puissants. **[08].**

III.2 Les deux langages de programmation des µCs PIC 16f876 le plus utilisé :

III.2.1 Langage' ASSEMBLEUR' (ASM):

III.2.1.1. Définition :

Un langage d'assemblage ou langage assembleur est, en programmation informatique, un langage de base neveu qui représente le langage machine sous une forme lisible par un humain. Les combinaisons de bits du langage machine sont représentées par des symboles dits « mnémonique » (du grec *mnêmonikos*, relatif à la mémoire), c'est-à-dire faciles à retenir. Le programme assembleur convertit ces mnémoniques en langage machine en vue de créer par exemple un fichier objet ou un exécutable. Dans la pratique courante, le même terme *assembleur* est utilisé à la fois pour désigner le langage d'assemblage et le programme assembleur qui le traduit. On parle ainsi de « programmation en assembleur ». **[09]**

Le PIC16F876 peut se programmer dans divers langages mais la langue de base, celle qui est la plus performante, est l'assembleur. Il s'agit d'un langage très proche de la langue machine ce qui lui confère une efficacité inégalée. Inconvénient, son utilisation va vous obliger à penser comme un ordinateur ce qui n'est pas évident à priori. **[10]**

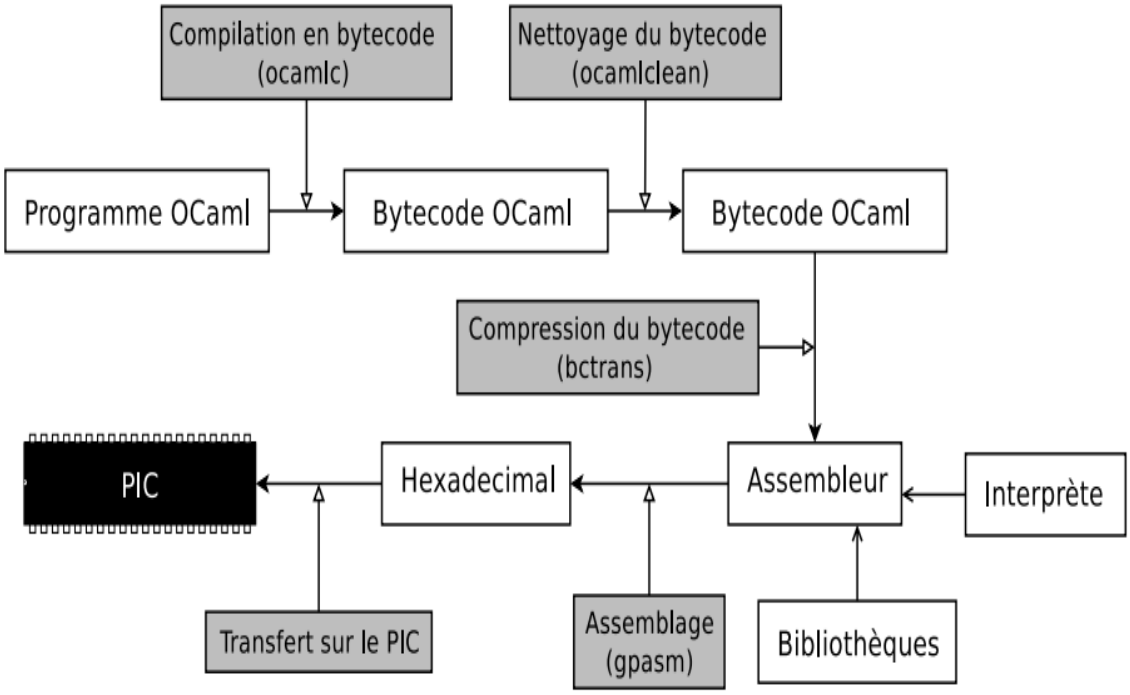

Figure III.1: Chaîne de production.

III.3 Structure d'un programme en assembleur :

III.3. 1 Registres importants :

•W : Le registre W est le registre de travail sur 8 bits pour réaliser des opérations arithmétiques ou logiques.

•STATUS :

- IRP : sélection de la banque (ADR indirect); RP1:RP0 : sélection de la banque (ADR direct)
- TO: time-out bit ; PD: power-down ; Z: zero bit ; DC: digital carry ; C: carry

•OPTION_REG:

- RBPU : PORTB pull-up(on/off) ; INTEDGS : INT edge (montante/descendante)
- T0CS : T0 clock source (RB4/CLK int) ; T0SE : T0 source edge (montante/descendante)
- PSA : pre-scaleron T0 ou WD ; PS2:PS0 : valeur prescaler

•INTCON :

- GIE: Global Interrupt Enable ; PEIE: Peripheral Interrupt Enable ;
- T0IE: TMR0 Over flow Interrupt Enable ; INTE: RB0/INT External Interrupt Enable
- RBIE: RB Port Change InterruptEnable ; T0IF: TMR0 Over flow Interrupt Flag
- INTF: RB0/INT External Interrupt Flag ; RBIF: RB Port Change Interrupt Flag **[11].**

III.3.2 Directives :

•LIST : indique le type de processeur

•**#include :** indique les fichiers où sont regroupées les assignations

•**CONFIG** : fixe le fonctionnement du PIC à plusieurs niveaux (debugging, watchdog, protection du code, oscillateurs,…)

•INC : pour l'utilisation et la définition de ces variables de configuration

•EQU : remplacer un nombre par une chaîne de caractères

•Syntaxe : assignations EQU nombre à remplacer

•#DEFINE : remplacer un texte plus complexe par une chaîne de caractères.

•Syntaxe : définition chaîne a substituer

•MACROS : remplacer des mots de code par une chaîne de caractères.

•Syntaxe : nom de macro MACRO ; endm : Code de macro

•CBLOCK : suivi d'une adresse, permet de définir le début d'une zone de variables.

•Syntaxe : CBLOCK adresse de début de zone, Nom de variable: taille de zone(en bites), ENDC

•ORG : suivi d'une adresse, précise à quelle adresse les instructions qui suivent seront placées dans le PIC.

•END : fin du programme**.[11]**

III.3.3 Instructions :

•Syntaxe : étiquette |_| mnémonique opérande destination

- Etiquette : repère pour le programme
- Tabulation (= |_|) : il est interdit d'écrire autre chose qu'une étiquette en première colonne.
- Mnémonique : symbole littéral = instruction
- Destination : W, F ou numéro de 0 à 7
- Sources : (1 seule à la fois !) EEPROM, T0, RB0/INT, PORTB
- ADR d'INTR unique : 0x04
- Sauvegarde de l'état du système :
- Uniquement le PC est sauvé lors du saut vers la routine d'interruption → le reste du contexte
- (W, STATUS,…) doit être sauvé manuellement **[10], [11].**

III.3. 4 Jeu d'instructions :

Le microcontrôleur PIC 16F876 possède 35 instructions de bases. Chaque instruction ne dure qu'un cycle machine (4/fQ) sauf les instructions de sauts telles que GOTO, CALL, est… **[07].**

BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Tableau III.1 : Les déférents types d'instruction de µC PIC 16F876

III.3.5. Interruptions (INTR) :

- INTR = rupture du déroulement normal par un évènement déclencheur → exécution d'une routine d'INTR

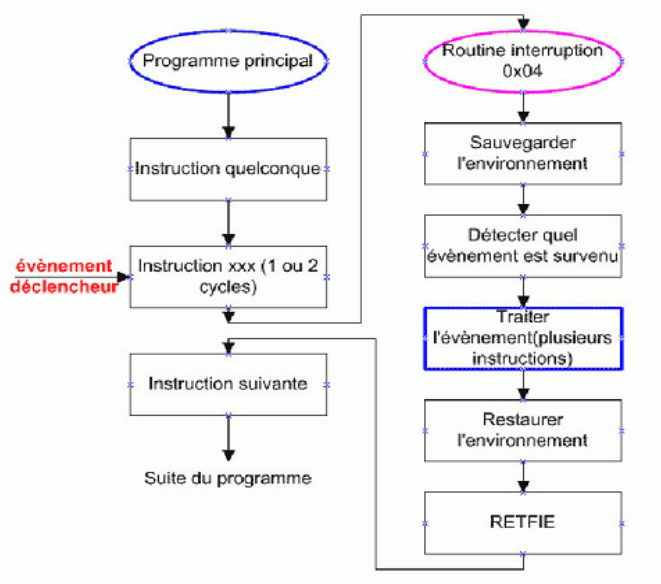

Figure III.2 : Les Interruptions (INTR) du programme interne de µCpic16f876

III.4. Exemple d'un programme simple :

Programmer un microcontrôleur consiste à écrire une suite de 0 et de1 en mémoire programme. Cette suite sera décrite dans un fichier « .hex » généré par **MPLAB** à partir de l'association de mnémoniques et de directive d'assemblage. On trouvera ci-après un programme simple permettant de faire basculer la sortie RB7 du port B d'un microcontrôleur PIC16F876. **[11].**

```
;*****************************************************************
;                              CONFIGURATION                      *
;*****************************************************************

    LIST      p=16F876             ; Définition de processeur
    #include <p16F876 .inc>        ; Définitions de variables

    __CONFIG    _CP_OFF & _DEBUG_OFF & _WRT_OFF & _CPD_OFF & _LVP_OFF & _BODEN_OFF &
    _PWRTE_ON & _WDT_OFF & _HS_OSC

;*****************************************************************
;                              ASSIGNATIONS                       *
;*****************************************************************

OPTIONVAL EQU       H'0088'    ; Valeur registre option
                               ; Résistance pull-up OFF
                               ; Pas de préscaler

;*****************************************************************
;                              DEFINE                             *
;*****************************************************************

#DEFINE LED         PORTB,1              ; Led rouge

;*****************************************************************
;                              MACRO                              *
;*****************************************************************

LEDON macro
          bsf       LED
    endm

LEDOFF macro
          bcf       LED
    endm

    bcf             STATUS,RP0           ; repasser banque 0
    goto  start                 ; sauter au programme principal
;*****************************************************************
;                    DECLARATIONS DE VARIABLES                    *
;*****************************************************************

    CBLOCK 0x020                ; début de la zone variables

    cmpt1 : 1                   ; compteur de boucles 1
    cmpt2 : 1                   ; compteur de boucles 2
    cmpt3 : 1                   ; compteur de boucles 3

        ENDC                    ; Fin de la zone
```

```
;*****************************************************************
;                       DEMARRAGE SUR RESET                      *
;*****************************************************************

    org 0x000                   ; Adresse de départ après reset
    goto    init                ; Adresse 0: initialiser

;*****************************************************************
;                        INITIALISATIONS                         *
;*****************************************************************

init
    clrf   PORTB                ; sorties portB à 0
    bsf    STATUS,RP0           ; sélectionner banque 1
    movlw  OPTIONVAL            ; charger masque
    movwf  OPTION_REG           ; initialiser registre option

                    ; initialisations spécifiques
                    ; ---------------------------
    bcf    LED          ; LED en sortie (banque1)
    ; LEDOFF           ; ou utiliser LEDOFF

; etc ...
END                     ; fin code
```

III.5 Manipulation de bits :

a - Forçage de bits :

Il s'agit de 2 instructions permettant de mettre à l'état logique "0" ou "1" un bit d'un octet de l'espace mémoire SFR. Elles sont le plus souvent utilisées pour positionner des bits des registres du microcontrôleur PIC.

Syntaxe : BSF f,b pour mettre à l'état logique "1" ou BCF f,b pour mettre à l'état logique "0"

Exemples : BCF PORTA, 2 ; Mise à l'état logique "0" du bit n°2 du registre ; PORTA
BSF STATUS, 0 ; Mise à l'état logique "1"du bit n°0 du registre STATUS, c'est-à-dire la retenue (CARRY)

b - Test de bits :

Il s'agit de 2 instructions permettant de tester un bit d'un octet de l'espace mémoire SFR. Elles sont le plus souvent utilisées pour déterminer l'état des bits des registres du microcontrôleur PIC. En fonction du résultat du test :

- le programme se poursuit avec l'instruction suivante (résultat du test faux).
- le programme saute l'instruction qui suit le test. **[07]**

Syntaxe : BTFSS f,b ou BTFSC f,b

Syntaxe : BTFSS f,b ou BTFSC f,b

III.6 Configuration des ports (REGISTRES PORTX ET TRISX) :

Tous les ports sont pilotés par deux registres :

- le registre TRISX : C'est le registre de direction. Il détermine si le port X ou certaines lignes de port fonctionnent en entrée ou en sortie. L'écriture d'un "1" logique correspond à

une entrée ("1" comme Input) et l'écriture d'un "0" logique correspond à une sortie ("0" comme Output).

- le registre PORTX. Il détermine l'état des lignes configurées en sortie par TRISX.
- lors de l'initialisation (RESET), toutes les lignes des ports sont configurées en entrées.
- les registres TRISX appartiennent à la BANQUE 1.
- Lors de l'initialisation du microcontrôleur PIC, il ne faut pas oublier de changer de page mémoire pour les configurer. **[07]**

III.7 Programmation en C d'un µC PIC :

III.7.1 Le plate forme de logiciel 'mikro c' :

III.7.1.1 Introduction :

Le « MikroC » est un compilateur pour PIC Conçu par la société « Mikroéléktronika », le compilateur C nouvelle génération "MikroC" pour microcontrôleurs PIC bénéficie d'une prise en main très facile. Il comporte plusieurs outils intégrés (mode simulateur, terminal de communication, gestionnaire 7 segments, analyseur statistique, correcteur d'erreur, explorateur de code...) ; Il a une capacité à pouvoir gérer la plupart des périphériques rencontrés dans l'industrie (Bus I2C, 1Wire, SPI, RS485, Bus CAN, cartes compact Flash, signaux PWM, afficheurs LCD et 7 segments...); de ce fait il est un des outils de développement incontournable et puissant. Il est conçu pour fournir les solutions les plus faciles que possibles pour des applications se développant pour les systèmes à microcontrôleur. Il contient un large ensemble de bibliothèques de matériel, de composant et la documentation complète. **[13]**

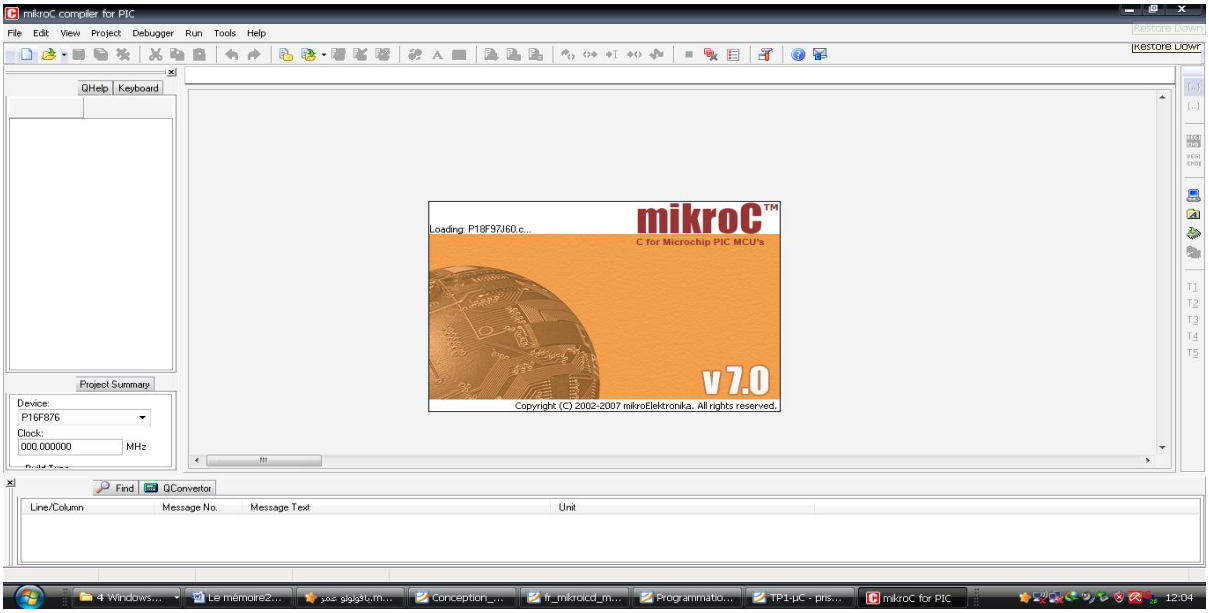

Figure III.3 :Le plate forme du logiciel MikroC

III.7.1 .2. Création d'un premier projet :

MikroC PRO pour PIC ® permet de créer des projets qui contiendront un seul fichier projet (fichier avec l'extension mcppi.) et un ou plusieurs fichiers sources (fichiers avec l'extension c.). Il est possible de gérer plusieurs projets à la fois. Les fichiers sources ne peuvent être compilés qu 'ils font partie d'un projet.

Un fichier "projet" contient les informations suivantes :

• le nom du projet et éventuellement une description,

• le µC utilisé,

• la fréquence d'horloge,

• la liste des fichiers sources,

• des fichiers binaires etc.

A présent, nous allons créer un nouveau projet, écrire du code, le compiler et le tester. Le but est de faire clignoter les LEDs du port B de notre µC. Après avoir inséré le dongle (sentinelle) dans l'une des prises USB de votre machine, double cliquer sur l'icône du compilateur mikroC PRO for PIC® du menu Start ou sur le raccourci du bureau de votre ordinateur pour le faire démarrer.**[14].**

L'environnement de travail intégré du compilateur s'affiche sur l'écran. Il contient plusieurs fenêtres que l'on peut afficher ou non. Il est également possible des les déplacer ou encore de les réduire, ceci permet à chacun de configurer son propre espace de travail. Un arrangement possible est montré sur la figure suivante :

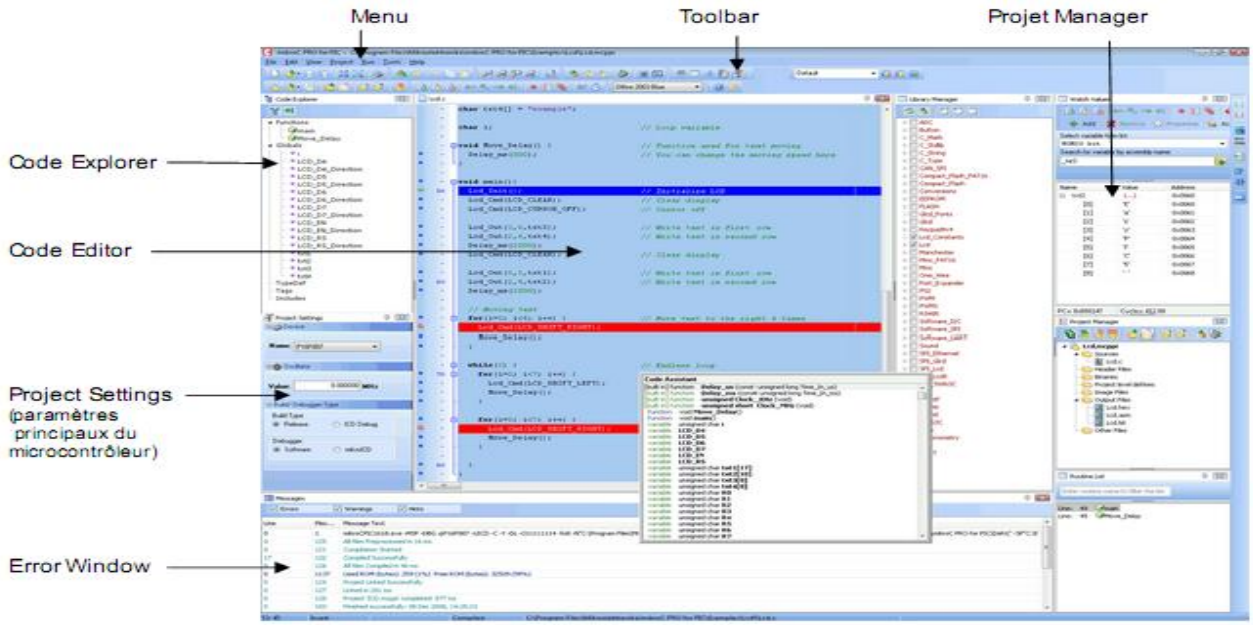

Figure III.4 : L'environnement de travail du logiciel MikroC

Vous allez maintenant pouvoir démarrer un nouveau projet. Sélectionnez l'option "New Project" dans le menu Project ou cliquez directement sur l'icône "New Project" dans la barre d'outils "Project". Il ne reste plus qu'à se laisser guider, appuyer sur le bouton "Next".**[14].[15].**

Figure III.5 : Création d'un nouveau projet dans logiciel MikroC

La première des choses à faire est de spécifier des informations générales sur le projet, son nom, son emplacement, le type de µC et la fréquence de l'horloge (8 MHz).

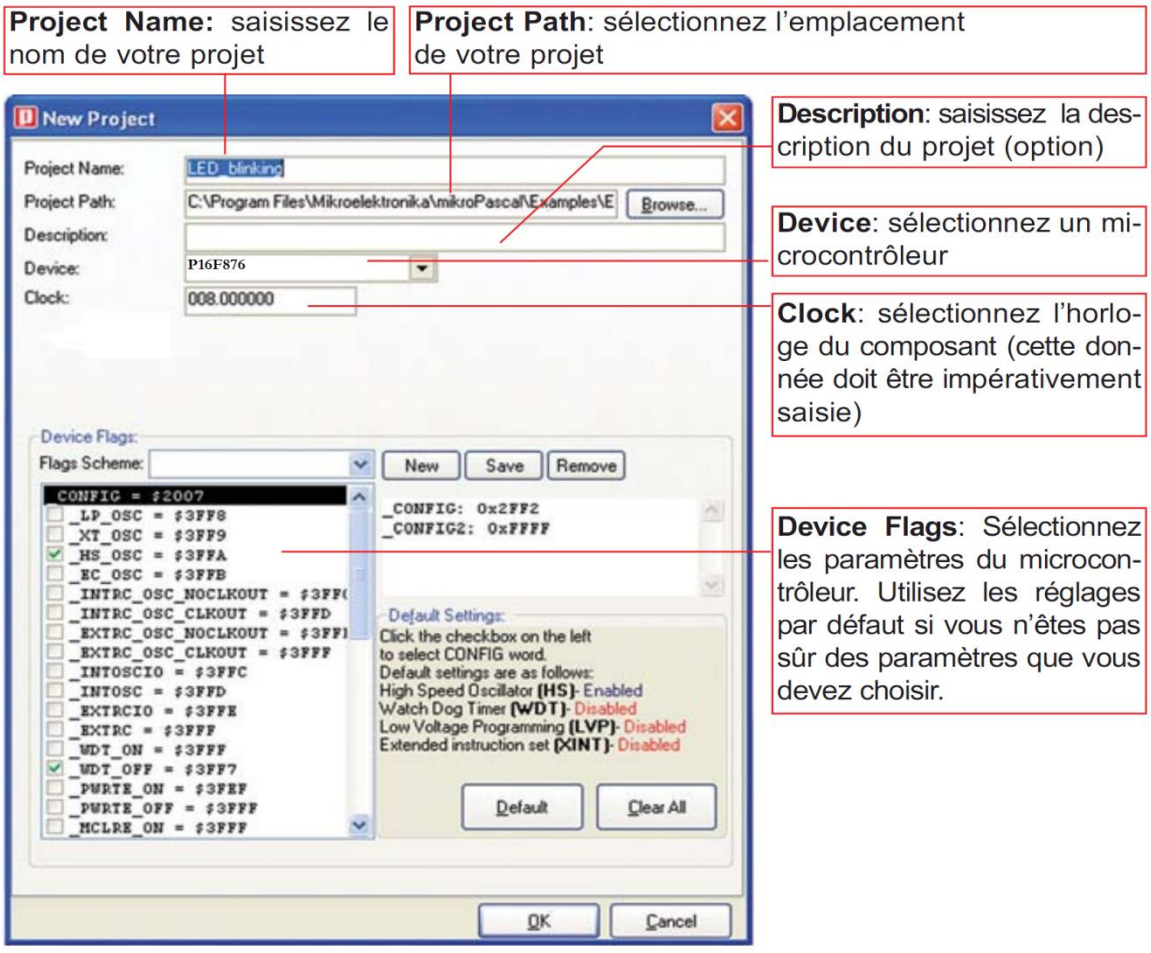

Figure III.6 : Configuration du projet (Project Settings)

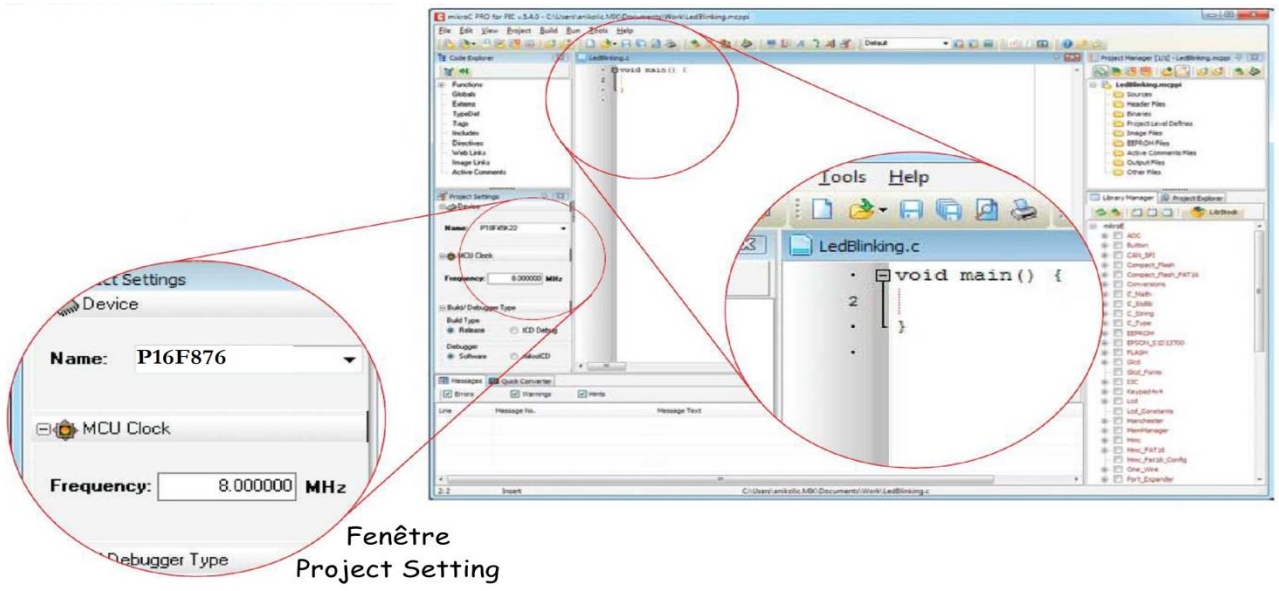

Figure III.7 : Fenêtre projet setting de logiciel MikroC

III.8 Compilation :

A présent, nous allons compiler le projet afin de créer le fichier **.hex** qui sera chargé dans le **µC**. La compilation inclue ici la compilation à proprement parler (génération d'un code machine par fichier), l'édition des liens ou linking (lien entre fichiers et bibliothèques) et l'optimisation, tâches qui seront faite de manière automatique.

Pour compiler le projet, cliquer soit sur l'icône de la barre des Tâches  ou dans le menu "Build", cliquer sur Build [CTRL+F9].

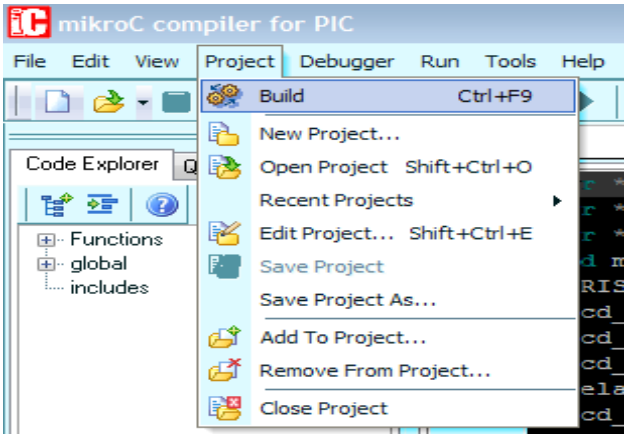

Figure III.8 : Compilation de projet avec le compilateur MikroC

La fenêtre "message", si elle est activée, contient des détails sur le résultat de la compilation. Le compilateur créé automatiquement les fichiers de sortie, dont le fichier LedPortB.hex

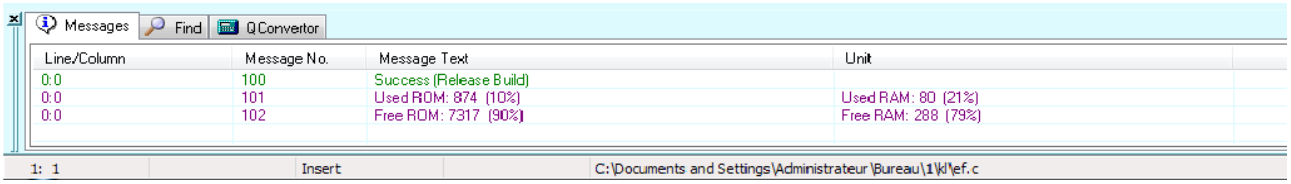

Figure III.9 : le résultat de la compilation dans la fenêtre "message" de logiciel MikroC

Barre de progression s'affiche pour vous informer sur l'état de la compilation. S'il y a des quelques erreurs, vous en serez informé dans la fenêtre d'erreur suivant :

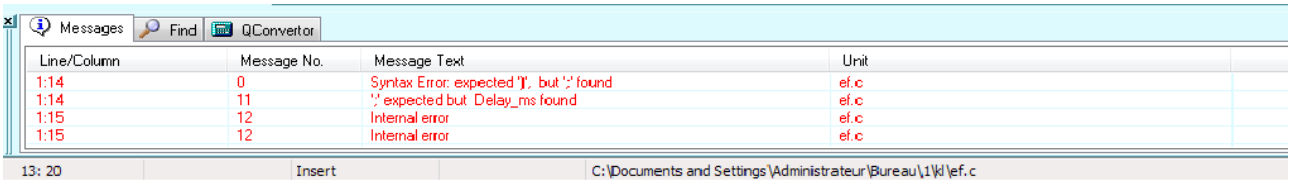

Figure III.10 : Avertissement des erreurs dans la fenêtre "message" de logiciel MikroC

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 d'être inscrits dans d'autres projets.	.mcl
List File	L'mage globale de l'allocation de mémoire du PIC pour : adresses d'instructions, les registres et les étiquettes du programme.	.lst
Assembler File	Le fichier en assembleur avec des noms symboliques, obtenus à partir de liste des fichiers.	.asm

Tableau III.1 : Les fichiers de codage dans l'environnement MikroC [14], [15]

III.9 L'environnement PROTEUS

III.9.1 Introduction:

Proteus est une suite logicielle destinée à l'électronique. Développé par la société Labcenter Electronics, les logiciels incluent dans Proteus permettent la CAO dans le domaine électronique. Deux logiciels principaux composent cette suite logicielle: ARES, ISIS, PROSPICE et VSM.

Le but de ce chapitre est de parcourir le processus de saisie d'un circuit de complexité moyenne pour vous familiariser avec les techniques requises pour gérer ISIS. Ce tutorial commence avec les sujets les plus simples, comme le placement et le câblage des composants, avant de passer

en revue l'utilisation de fonctionnalités plus complexes telle que la création des nouveaux composants.

III.9.2 Présentation générale :

Cette suite logicielle est très connue dans le domaine de l'électronique. De nombreuses entreprises et organismes de formation (incluant lycée et université) utilisent cette suite logicielle. Outre la popularité de l'outil, Proteus possède d'autres avantages

- Pack contenant des logiciels facile et rapide à comprendre et utiliser
- Le support technique est performant
- L'outil de création de prototype virtuel permet de réduire les coûts matériel et logiciel lors de la conception d'un projet. **[16].**

III.9. 2.1 ARES :

Le logiciel ARES est un outil d'édition et de routage qui complètement parfaitement ISIS. Un schéma électrique réalisé sur ISIS peut alors être importé facilement sur ARES pour réaliser le PCB de la carte électronique. Bien que l'édition d'un circuit imprimé soit plus efficiente lorsqu'elle est réalisée manuellement, ce logiciel permet de placer automatiquement les composants et de réaliser le routage automatiquement. **[16].**

III. 9.2.2 ISIS :

Le logiciel ISIS de Proteus est principalement connu pour éditer des schémas électriques. Par ailleurs, le logiciel permet également de simuler ces schémas ce qui permet de déceler certaines erreurs dès l'étape de conception. Indirectement, les circuits électriques conçus grâce à ce logiciel peuvent être utilisé dans des documentations car le logiciel permet de contrôler la majorité de l'aspect graphique des circuits. **[16].**

III.10 Définition des dimensions de la feuille de travail :

Sélectionnez dans le menu déroulant "Système" l'instruction "Définir taille des feuilles", puis cochez la rubrique "Perso", et donnez-lui les dimensions **8.26 in par11.69 in**, ce qui correspond au format A4 réel.

Si l'orientation souhaitée est le format dit "Paysage", il convient de permuter ces valeurs.

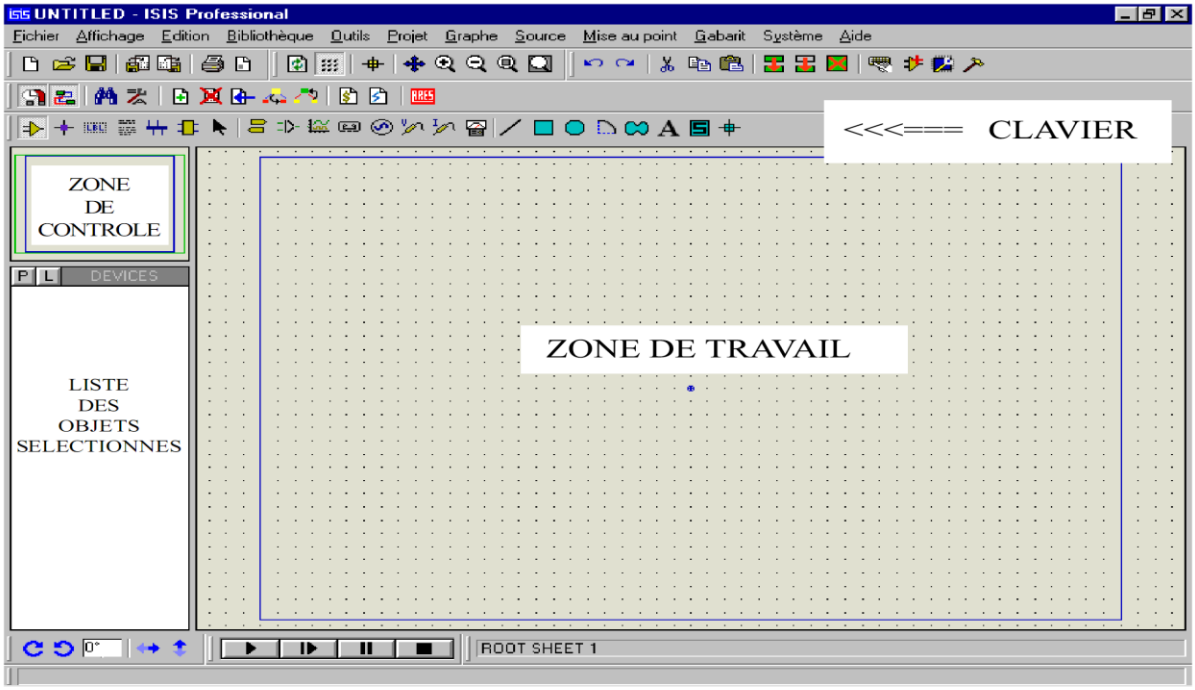

Figure III.11 : L'environnement de travail du logiciel ISIS

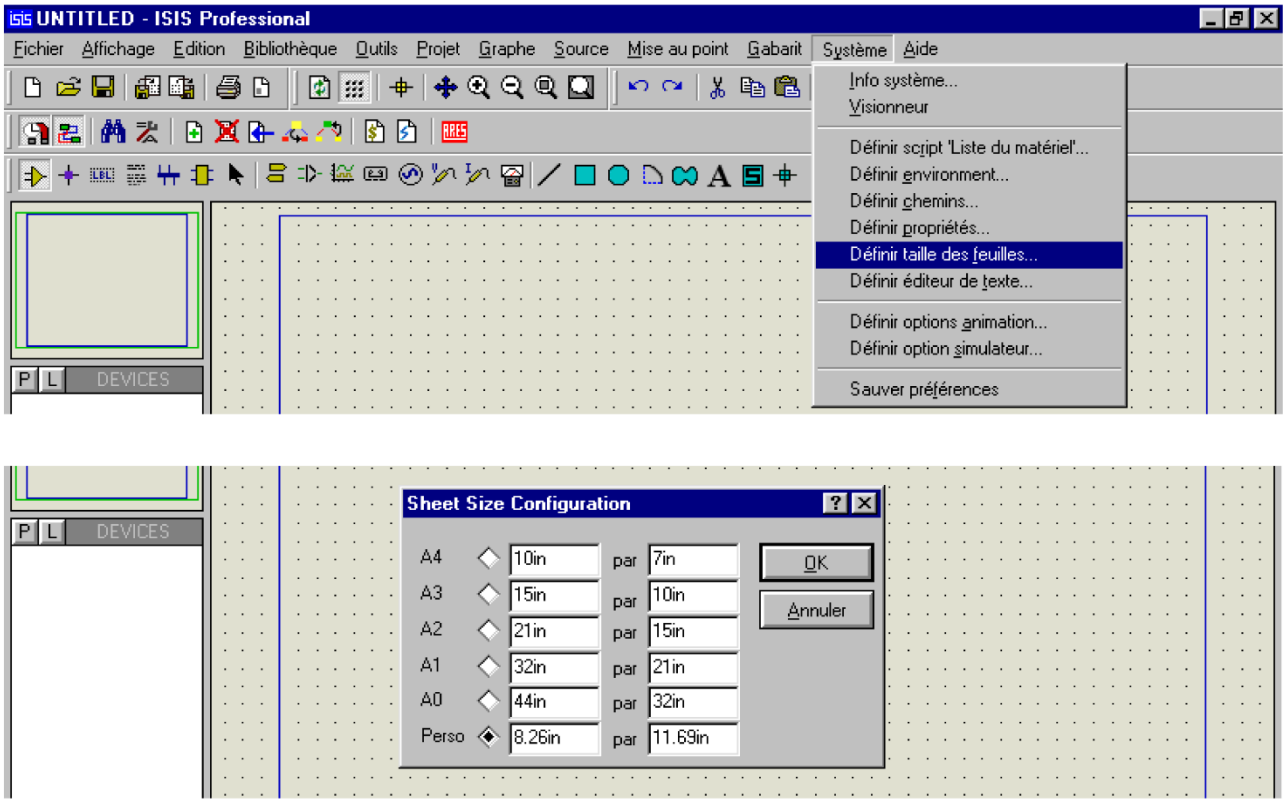

Figure III.12 : Définition la taille des feules sur L'environnement de travail du logiciel ISIS

III.11 Recherche des composants :

Sélectionnez la touche "Composant" du clavier, puis cliquez sur la lettre P (Prendre composants) ; les librairies contenant les composants apparaissent alors dans un ordre théoriquement alphanumérique.

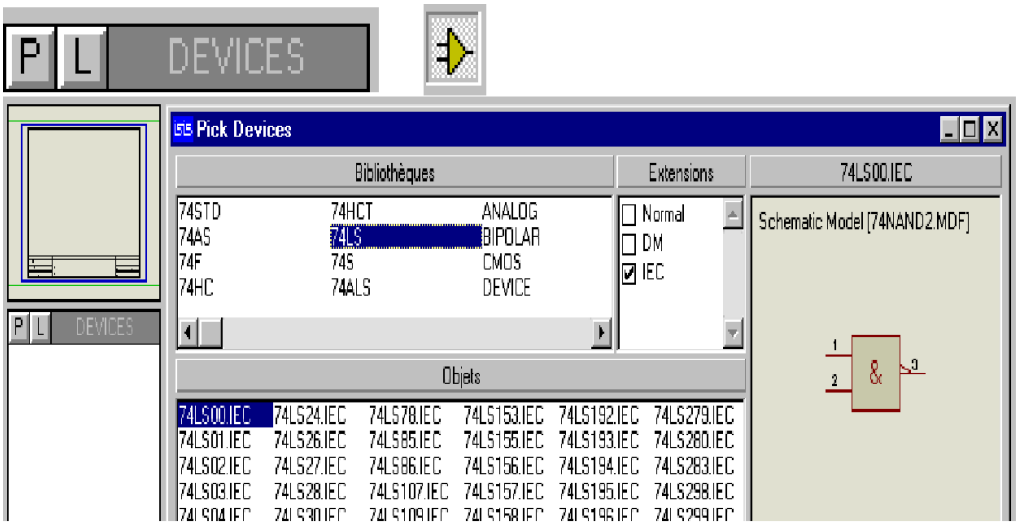

Figure III.13 : Recherche à des composants ou à des outillages électroniques sur L'environnement d'ISIS

III.12 La partie pratique (simulation de « LCD Library »)

III.12.1 programmation sur Mikro C Version 2007:

Pour programmé LCD Library On utilisé le logiciel ISIS (Proteus 6) qui est un très bon logiciel de simulation en électronique.

```
#include <16F8776.h>
#device adc=8
#FUSES NOWDT            //No Watch Dog Timer
#FUSES HS               //High speed Osc (<= 8mhz)
#FUSES NOPUT            //No Power Up Timer
#FUSES NOPROTECT        //Code not protected from reading
#FUSES NODEBUG          //No Debug mode for ICD
#FUSES BROWNOUT         //No brownout reset
#FUSES NOLVP            //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD            //No EE protection
#FUSES NOWRT            //Program memory not write protected
#use delay(clock=8000000)
#include <LCD.C>
```

```
char i;                          // Loop variable
void Move_Delay() {              // Function used for text moving
  Delay_ms(1000);                // You can change the moving speed here
}
char *text1 = "mikroElektronika";
char *text2 = "  Benzekri'Rabie";
char *text3 = "Doudou'Hammou";
char *text4 = "Kechar'Smail";
void main() {
  TRISB = 0;                      // PORTB is output
  Lcd_Init(&PORTB);               // Initialize LCD connected to PORTB
  Lcd_Cmd(Lcd_CLEAR);             // Clear display
  Lcd_Cmd(Lcd_CURSOR_OFF);  // Turn cursor off
  Delay_ms(1000);
  Lcd_Out(1, 1, text1);     // Print text to LCD, 2nd row, 1st column
  Delay_ms(30);
  Lcd_Out(2, 1, text2);     // Print text to LCD, 2nd row, 1st column
  Delay_ms(60);
  Lcd_Out(3, 0, text3);     // Print text to LCD, 2nd row, 1st column
  Delay_ms(80);
  Lcd_Out(4, 0, text4);     // Print text to LCD, 2nd row, 1st column
}                           //~!
```

III.2.2.Simulation avec ISIS-Proteus 6:

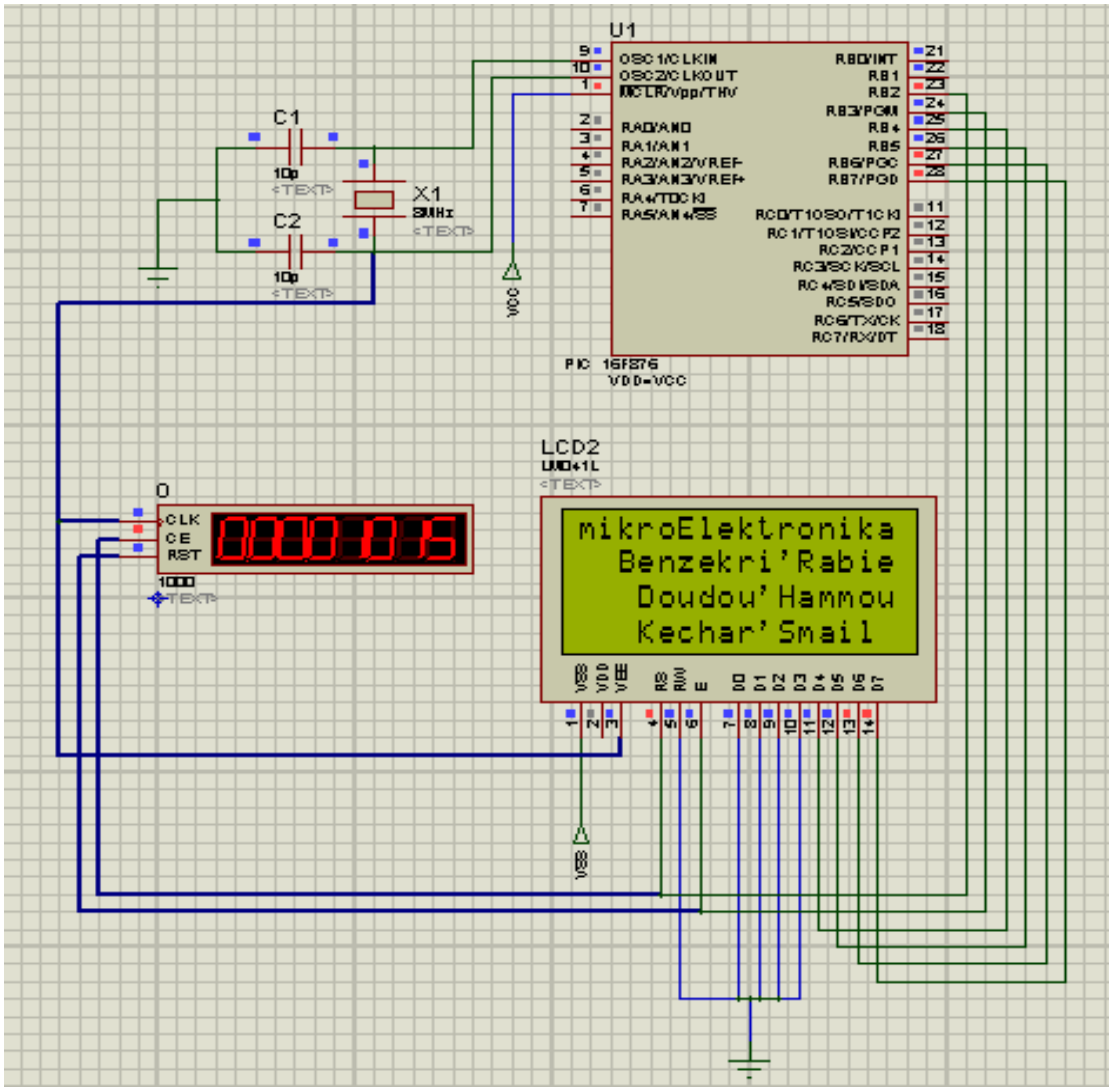

Figure III.14 : Simulation avec ISIS-Proteus6

III .13. Conclusion :

Pour terminer ce troisième chapitre autour d'un microcontrôleur PIC on peut déjà dire qu'avec un minimum de moyen (un programmateur de PIC), on va pouvoir réaliser une application simple ne comportant que très peu de composants et peu coûteuse. Il faudra bien évidemment se familiariser (petit à petit !) avec le langage assembleur ou Mikro C, qui paraît au premier abord assez compliqué et surtout avec les instructions, ainsi qu'à la façon de les utiliser.

Nous avons analysé et compilé notre premier programme sous Mikro C, dans le prochain numéro nous aborderons les aspects programmateur afin de pouvoir transférer le fichier binaire vers le PIC .

Nous avons enfin téléchargé notre programme vers la mémoire du PIC. Sans oublier la partie simulation qui peut être bien sûr approfondie afin de bénéficier des possibilités d'apprentissage du fonctionnement interne des registres. à réaliser pour tester vos programmes ainsi qu'un afficheur LCD pour PIC 16F876 qui fera peut être qui sait sensation pour les Panneau de publicité.

Ainsi que la partie simulation : Sans oublier la partie simulation sur l'environnement ISIS à proteus.

Conclusion générale :

Dans notre vie quotidiennf on remarque que les domaines d'application des circuits programmable très vaste dont informatique, Machinerie Industrielle, Télécommunication, Instrumentation, transport, militaire.....etc.et il existe plusieurs types des circuits programmable comme FPGA, DSP et les PIC….etc. Pour ce la on a choisi le microcontrôleur (μC) pic 16F876 pour étudier son architecture puis on a passé à l'étape suivant dont elle est: comment programmé le μC pic16F876, et il existe plusieurs types de langage de programmation comme Assembleur, MikroC…… etc. Donc on a utilisé le langage de programmation (MikroC) pour programmer le microcontrôleur pic16F876 et à la fin pour simuler ce programme on a plusieurs logiciel de simulation comme BoostC, MPLAB, Proteus….. etc. On a utilisé le logiciel ISIS Proteus pour la simulation, comme celui de notre étude, on a fonctionné l'afficheur LCD à partir d'écriture le programme dans « MikroC » puis à l'aide de logiciel ISIS Proteus on a raccordé entre le 'μC pic16F876' et l'afficheur LCD puis on a mis ce programme dans le 'μC pic16F876', Ensuite on a le simulé. Enfin, selon ce qu'on a vu précédemment on dit que les circuits programmables sont très nécessaires.

Bibliographie :

[01] Les microcontrôleurs, Jérôme Vicente, Dpt ME - Otion SIIC 2ème année 2005-2006 – ver 4.0

[02] Microcontrôleurs Famille Mid-Range de Microchip Le PIC 16F876/877, A. Oumnad.

[03] DSP et temps réel, M. Correvon.

[04] Arnaud Tisserand INRIA LIP Arénaire, Séminaire MIM 16 décembre 2003.

[05] ELE3301 – Systèmes logiques programmables.

[06] Informatique Industrielle, Cours Master SIS, Microcontrôleurs Microchip,
Intervenants : Marc Allain - marc.allain@fresnel.fr
Julien Marot - julien.marot@fresnel.fr
Université Paul Cézanne-Aix Marseille III

[07] Microcontrôleur 16F876, PDF.

[08] Rapport de projet de fin d'études, Conception et réalisation d'un enregistreur de données, Réalisé par: Alibi Elmehdi et Jawadi Sami, Université Virtuelle de Tunis.

[09] http://fr.wikipedia.org/wiki/assembleur

[10] Introduction à la programmation de microcontrôleurs PIC16F876 Jean-MarcLICHTLE

03 novembre 2002.

[11] Introduction aux microcontrôleurs et à leur assembleur Illustration par le PIC16F876,

F. Senny, Université de Liège, Faculté des Sciences Appliquées.

[12] PIC : COURS ASSEMBLEUR, S.T.I. Génie Electrique option Electronique, Programmation_assembleur_des_PIC.

[13] Rapport de projet de fin d'études, Conception et réalisation d'un enregistreur de données, Réalisé par: Alibi Elmehdi et Jawadi Sami, Université Virtuelle de Tunis.

[14] Informatique Industrielle, TP 1 – Microcontrôleur Prise en main de la platine EasyPic7 et du compilateur mikroC PRO for PIC, Thomas Quiniou et Albert Ranaivosoloarimanana , université de la nouvelle calédonie.

[15] Programmation en mikroC. Application pour les microcontrôleurs de la famille PIC, V. TOURTCHINE, UNIVERSITE M'HAMED BOGARA DE BOUMERDES FACULTE DES SCIENCES - DEPARTEMENT PHYSIQUE, BOUMERDES - 2012

[16] http://Elektronique.fr/Proteus (ISIS et ARES).