République Algérienne Démocratique et Populaire

Ministère de l'enseignement Supérieur

La Recherche scientifique

Université Echahid Hamma Lakhdar d'El-Oued

Faculté des Sciences et de la Technologie

Mémoire de Fin d'Étude

En vue de l'obtention du diplôme de

MASTER ACADEMIQUE

Domaine : Sciences et Technologie

Filière : Génie Électrique

Spécialité: Commande Électrique

Thème

Commande des équipements électriques par microcontrôleurs

"Simulations et Réalisations"

Réalisé par:

Adjiba Brahim.

Chalghoum Abdelmonaim.

Encadré par:

ALLAL ABDERAHIM

Soutenu en juin 2015

sommaire

CHAPITRE I: *L' architecture Générale Des Microcontrôleurs*

Introduction Générale ……..……..……**1**

Introduction……..……..……**2**

I.1 :Rôle d'un système à microcontrôleur…..…….……...….**2**

I.2: Schéma fonctionnel de l'organisation d'un système à microcontrôleur

……..……..……**3**

I.3:Rôle des différents éléments composants de l'organisation fonctionnelle d'un système à microcontrôleur…….…….……**5**

I.3:a. Le microprocesseur……...……**5**

I.3:b. Les mémoires du microcontrôleur……...….**5**

I.3:c. Le contrôle du microcontrôleur ……...……**5**

I.3:c.1 L'horloge du microcontrôleur…….**5**

I.3:c.2 Le chien de garde du microcontrôleur……**5**

I.3:c.3 Le reset à la mise sous tension ……....…….**6**

I.3:c.4 Surveillance de l'alimentation ……..……...……**6**

I.4: Les périphériques d'un microcontrôleur ……..……..……**6**

I.4:a. Les CAN (Conversion Analogique Numérique) et CNA (Conversion Numérique Analogique) ……...……...……..…**6**

I.4:b. Les ports d'entrées/sorties d'un microcontrôleur……..…….…**6**

I.4:c. La transmission de données séries asynchrone et synchrone….. ….**6**

I.4:d. La gestion Ethernet……..…….....**6**

I.4:e. La gestion de bus CAN ……..……..……**7**

I.4:f. La gestion de bus USB …….……..……...**7**

I.5: Les périphériques externes d'un microcontrôleur ……..…**7**

I.5:a. Le décodage d'adresses ……..……..**7**

I.5:b. Les bus du microcontrôleur……....…….**7**

sommaire

I.5:c. Les mémoires ……..……...……..**8**

I.5:d. Les périphériques optionnels ……...…...…….**8**

I.6: Applications –critères de choix ……..……...…..**8**

I.7: Critères de choix d'un microcontrôleur……...…….**8**

I.8: Type de microcontrôleur ……..…….**9**

I.8:a. Micro chip……...……..**9**

I.8:b. Atmel ……..…….**9**

I.8:c. Philips ……...……...……...**10**

I.8:d. Motorola ……..……..…….**11**

I.9: Définition d'un PIC ……..……...…**12**

I.9:a. Les différentes familles des PIC ……...…….…....**12**

I.9:b. Identification d'un Pic……...……**12**

Conclusion**…….……...…12**

CHAPITRE II *:le microcontrôleur 16F84*

Introduction**……...……..14**

II.1: Qu'est-ce qu'un PIC**…….…….……...14**

II.2: PIC16F84**…….……15**

II.2:a. Brochage et fonction des pattes**…….……15**

II.2:b. Architecture générale ……..…….**16**

II.3: Organisation de la mémoire……...……..……**17**

II.3:a. Mémoire de programme……..……..……**17**

II.3:b. Mémoire de données……..……..…..…**18**

II.3:b.1- Registres généraux ……..……...…...**18**

II.3:b.2- Registres spéciaux - SFRs ……...…….……....**19**

II.3:b.3- Mémoire EEPROM ……...……..…….**21**

sommaire

II.4: Modes d'adressages **……...……..21**

II.4:a. Adressage immédiat **……...……..21**

II.4:b. Adressage direct **……..…...……..21**

II.4:c. Adressage indirect **……..……..21**

II.5: Ports d'entrées/Sorties**…….…….……....22**

II.5:a. Port A **……..……..…22**

II.5:b. Port B **……..……..…23**

II.6: Compteur **…….…….…….……...…24**

II.6:a. Registre TMR0 **……...…..…24**

II.6:b. Choix de l'horloge **……..……...…25**

II.6:c. Pré-diviseur **……...……...25**

II.6:d. Fin de comptage et interruption **……...……26**

II.6:e. Registres utiles à la gestion de timer0 **……...……...26**

Conclusion**……....26**

CHAPITRE III*: les logiciels utilisés*

Introduction**……..27**

III.1:Définition du logiciel "PROTEUS"**…….27**

III.1:a. logiciel proteus **……...……....27**

III.1:b. Présentation générale**……..…….27**

III.1:c. ISIS**…….……..…27**

III.1:d. SAISIE DU SCHEMA**……..……..28**

III.1:d.1-Démarrer le logiciel **…….28**

III.1:d.2-Création de la liste des composants**……...……...28**

III.1:d.3-Placement des composants sur le schéma**…….……28**

III.1:d.4-Placement des différentes connexions externes**……...……28**

sommaire

III.1:d.5-Placement des alimentations et des générateurs…..……**29**

III.1:d.6- Réalisations des connexions…….…….…..**29**

III.1:e. placement des instruments de mesure…….…….**30**

III.1:f. animation-simulation du fonctionnement…….**30**

III.1:g. utilisation avancée des appareils de mesure…….....….**30**

III.1:h. simulation par graphe (Chronogrammes) …….……..**31**

III.1:h.1- Mise en place des points de mesure…….……...**31**

III.1:h.2-Création de la fenêtre graphique……...……**31**

III.1:h.3-Indications des courbes à tracer……**31**

III.1:h.4. Réglage du temps de simulation……...……**31**

III.2: Langage et compilateur mikroC pour PIC……..……..**32**

III.2:a. Compilateur mikroC PRO pour PIC……..**32**

III.2:b. Création d'un nouveau projet……..……**34**

III.2:c. Compilation……**38**

III.2:d. Fichiers Sources……..…**39**

Conclusion…….……...…**41**

CHAPITRE IV: *Simulation Et Réalisation*

Introduction ……....….**42**

IV.1: Première projet ……...…..……..……...**42**

IV.1:a. Première étape : conception du schéma électrique avec le logiciel de simulations (Proteus) ISIS**...**…….…...…**42**

IV.1:b. Deuxième étape : Ecrire un programme à partir du logiciel de programmation (MikroC) qui correspond à la conception électrique **.....43**

IV.1:c. La troisième étape : Réalisation……...……...…**45**

IV.1:c.1- Tester le projet sur la plaque d'essai……...……..…**45**

sommaire

IV.1:c.2- Graver le programme sur le pic **……...……46**

IV.1:c.3- Raccordement des éléments de circuit **……..……..…47**

IV.2: Deuxième projet **……...……..…..…47**

IV.2:a. Réalisation**……..……...…….…….49**

IV.2:a.1- Tester le projet sur la plaque essai**……..…… 49**

IV.2:a.2- Raccordement des éléments de circuit**……50**

Conclusion **…….…… 50**

Conclusion générale**. ……..……..…51**

Bibliographies

sommaire

Liste des figures

CHAPITRE I

Figure I.1 : Organisation fonctionnel d'un système à microcontrôleur**.....4**

CHAPITRE II

Figure II.1 : Liste des composants présentés dans la documentation n°DS30430C**. ……..……..15**

Figure II.2 : Brochage du circuit**…….……15**

Figure II.3 : Architecture générale du PIC 16F8X**…….…… 16**

Figure II.4 : Organisation de la mémoire de programme et de la pile**....17**

Figure II.5 : Organisation de la mémoire de données**…….……18**

Figure II.6 : Description des SFR.**…….……..…19**

Figure II.7 : Registre d'étai du PIC - STATUS**……...……20**

Figure II.8 : Adressages direct et indirect à la mémoire de données**..…22**

Figure II.9 : Câblage interne d'une patte du port A**……..…23**

Figure II.10 : Câblage interne d'une patte du port B**……..…..24**

Figure II.11 : Organigramme du Timer0**……24**

Figure II.12 : Prise en compte de l'écriture dans le registre TMR0**……24**

Figure II.13 : Valeurs du pré-diviseur en fonction de PSA, PS2, PS2 et PS0**. ……..25**

Figure II.14: Registres utiles à la gestion de timer0**……...……26**

CHAPITRE III

Figure III.1:L'environnement IDE du compilateur microC PRO**…...…32**

Figure III.2: Boîte de dialogue « Options »**…….……33**

Figure III.3: Assistant de code**…….……33**

Figure III.4:Avertissement des erreurs**……39**

CHAPITRE IV

Figure IV.1: Plan du projet par le logiciel de simulations (Proteus) isis**…….……..…42**

Figure IV.2: programmateur de pic**……..…46**

Figure IV.3: Installation du circuit électrique sur la carte d'essai**……..47**

Figure IV.4: Plan du projet par le programme isis**……..48**

Figure IV.5: Installation du circuit électrique sur la carte d'essai**…..…50**

Liste des tableaux

Tab. I. 1 : Microcontrôleurs Microchip**…….……9**

Tab. I. 2: Microcontrôleurs Atmel**……...…..……..……10**

Tab. I. 3: Microcontrôleurs Motorola**……...12**

Tab .III .1: compilation (mikroC)**……..…….……39**

Tab. III.2 : Instructions d'itération**……...……...…41**

Introduction Générale

La technologie de pointe utilise pour la commande des processus industriels surtout les équipements électriques jusqu'a les navettes spéciales les microcontrôleurs qui sont le cerveau de guidage et contrôle majeur dans l époque du numérique.

Pour les raisons citées au dessus, nous avons vu de se pencher sur l étude de la commande des équipements par microcontrôleurs et montrer quelques exemples démonstratifs pour bien éclaircir ce type de commande puisqu' ils ont généralement la même technique.

Notre mémoire comprend quatre chapitres :

Le premier chapitre va étudier l architecture générale des microcontrôleurs

Le deuxième chapitre va se focaliser sur le microcontrôleur 16F84 en étudiant ces caractéristiques.

Le troisième chapitre abordera le sujet sur les logiciels utilisés dans la simulation et la réalisation de nos projets .

Le quatrième chapitre expliquera d'une manière claire les étapes à suivre pour la simulation ou la réalisation.

Enfin ; nous achèverons notre étude par une conclusion générale et les prochaines perspectives à faire au futur.

Chapitre I

L' architecture Générale Des Microcontrôleurs

Introduction

Un microcontrôleur est un circuit intégré qui rassemble aux éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte pour le programme, mémoire vive pour les données), unités de périphériques et interfaces d'entrées/sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

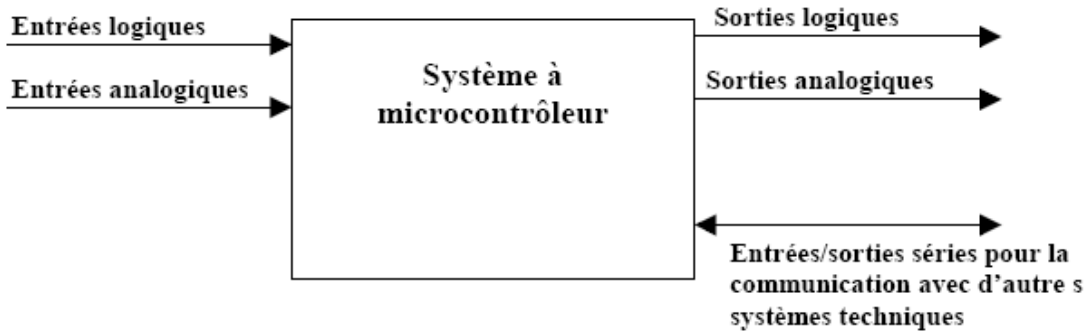
Par rapport à des systèmes électroniques à base de microprocesseurs et autres composants séparés, les microcontrôleurs permettent de diminuer la taille, la consommation électrique et le coût des produits. Ils ont ainsi permis de démocratiser l'utilisation de l'informatique dans un grand nombre de produits et de procédés.

Les microcontrôleurs sont fréquemment utilisés dans les systèmes embarqués, comme les contrôleurs des moteurs automobiles, les télécommandes, les appareils de bureau, l'électroménager, les jouets, la téléphonie mobile, etc.

I.1:Rôle d'un système à microcontrôleur [1]:

Un système à microprocesseur permet :

- d'acquérir des entrées logiques et analogiques représentant l'état du système technique,
- d'interpréter, la signification de ces entrées,
- de calculer, mémoriser, récupérer des variables logicielles intermédiaires,
- de gérer le temps,
- d'agir sur des sorties logiques et analogiques en fonction des entrées et des calculs réalisés de manière à modifier le fonctionnement du système technique (commande moteur, affichage d'informations,…)
- de communiquer par des liaisons séries avec d'autres systèmes techniques et/ou un ordinateur



I.2:Schéma fonctionnel de l'organisation d'un système à microcontrôleur [1]:

Ce schéma (Figure I.1) de la page suivante représente les différents périphériques internes à un microcontrôleur ainsi que le moyen utilisé pour communiquer avec des périphériques externes au microcontrôleur.

Attention, un même microcontrôleur ne possède pas forcément tous les périphériques représentés.

Le schéma fonctionnel de l'organisation fonctionnelle d'un microcontrôleur se trouve dans la page suivante :

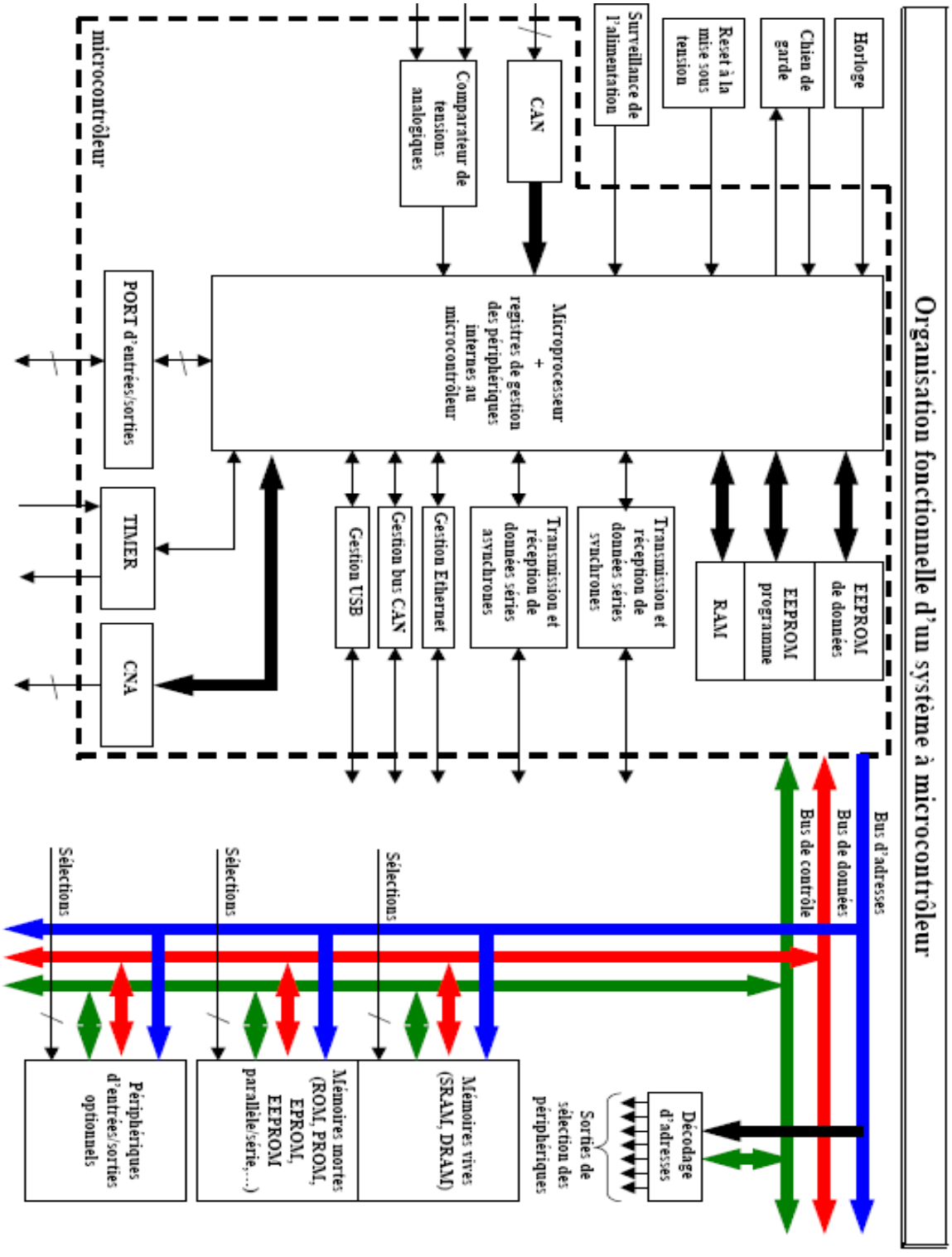


Figure I.1 : Organisation fonctionnel d'un système à microcontrôleur

I.3: Rôle des différents éléments composants de l'organisation fonctionnelle d'un système à microcontrôleur [1]:

On appelle microcontrôleur un circuit intégré qui est constitué d'un microprocesseur associé à un ou plusieurs périphériques.

I.3:a. Le microprocesseur[1] :

Toutes les informations transitées par le microprocesseur, il exécute un programme contenu en mémoire. Ce programme est constitué d'un ensemble d'instructions élémentaires codées, qui seront décodées puis exécutées au fur et à mesure par le microprocesseur.

Le microprocesseur est composé entre autre:

- D'un décodeur d'instruction qui va déterminer la tâche à exécuter.
- D'un séquenceur qui contrôle le fonctionnement de l'ensemble du microprocesseur.
- D'une Unité Arithmétique et Logique qui est chargée des opérations élémentaires (opérations logiques, addition, soustraction., comparaison, multiplication division,…).
- D'un compteur ordinal qui génère l'adresse de l'instruction qui devra être exécutée ou de la donnée qui devra être traitée.

Le microprocesseur utilisera un certain nombre de registres qui permettront de configurer et agir sur les différents périphériques.

I.3:b. Les mémoires du microcontrôleur[1] :

Il existe différents types de mémoires :

- **EEPROM programme :** c'est une mémoire morte dans laquelle on va stocker le programme qui va gérer le fonctionnement du système technique.
- **EEPROM données :** c'est une mémoire vive dans laquelle; on va stocker les données devant être sauvegardées si le système technique est mis hors tension.
- **RAM :** mémoire vive dans laquelle ;on va stocker des données temporaires nécessaires à l'exécution du programme de gestion du système technique. Ces données ne seront plus disponibles si le système technique est mis hors tension.

I.3:c. Le contrôle du microcontrôleur[1] :

I.3:c.1- L'horloge du microcontrôleur [1]:

Elle va donner la référence temporelle au microprocesseur pour exécuter les instructions.

L'horloge d'un microprocesseur est souvent réalisée grâce à un Quartz. Il existe certains microcontrôleurs qui ont la possibilité de sélectionner une horloge interne (sans composants externes) ce qui permet d'utiliser les broches de l'horloge pour d'autres périphériques.

I.3:c.2-Le chien de garde du microcontrôleur [1]:

C'est une structure, qui peut être interne ou externe au microcontrôleur, qui permet de vérifier le bon déroulement du programme.

Le microcontrôleur envoie des impulsions espacées de durées fixes au chien de garde.

Tant que les impulsions espacées de durées fixes arrivent au chien de garde, tout se passe bien.

Par contre dès que le chien de garde détecte l'absence d'une impulsion (le programme est bloqué), il produit une mise à zéro du programme de gestion du système technique de manière à débloquer le programme.

I.3:c.3- Le reset à la mise sous tension [1]:

Tout microcontrôleur a besoin d'un temps minimum avant de pouvoir commencer à lancer le programme. Ce temps est donné par la documentation constructrice.

Il faut par conséquent produire un signal de reset d'une durée supérieur à la mise sous tension.

I.3:c.4- Surveillance de l'alimentation[1] :

C'est une structure qui permet de produire un reset du microcontrôleur si une chute de l'alimentation est détectée (problème sur le système technique).

I.4 Les périphériques d'un microcontrôleur [1]:

I.4:a. Les CAN (Conversion Analogique Numérique) et CNA (Conversion Numérique Analogique) [1]:

Les CAN : ce périphérique se trouve souvent implémenté dans le microcontrôleur, il permet d'acquérir des grandeurs électrique de type analogique directement à partir d'une ou plusieurs broches du microcontrôleur la sortie est un nombre binaire.

Les CNA : ce périphérique permet de produire une tension analogique à partir de mots numériques internes au microcontrôleur.

I.4:b. Les ports d'entrées/sorties d'un microcontrôleur [1]:

Ces périphériques sont indispensables au microcontrôleur ils permettent :

-d'acquérir les entrées de types logiques indiquant l'état du système technique,

-de produire des sorties de types logiques permettant de commander les périphériques du système techniques (afficheurs, moteurs, buzzer,…).

I.4:c. La transmission de données séries asynchrone et synchrone [1]:

Ces périphériques permettent la communication avec d'autres systèmes technique et/ou un PC.

I.4:d. La gestion Ethernet[1] :

Les nouveaux microcontrôleurs disposent d'un périphérique permettant de gérer la liaison réseau de type Ethernet.

Ceci permet notamment de commander des systèmes techniques et/ou visualiser son état de fonctionnement à distance grâce à une page internet.

I.4:e. La gestion de bus CAN [1]:

Ce périphérique permet la communication série de données numériques avec des systèmes techniques dans des milieux perturbés notamment dans le domaine de l'automobile.

I.4:f. La gestion de bus USB [1]:

Ce périphérique permet de gérer le protocole de communication USB afin de connecter des appareils utilisant ce même protocole.

I.5: Les périphériques externes d'un microcontrôleur[1] :

Si les périphériques contenus dans le microcontrôleur ne sont pas suffisants, on peut rajouter certains périphériques externes.

Pour cela; il faut que le microcontrôleur dispose d'un bus d'adresses et d'un bus de données.

I.5:a. Le décodage d'adresses [1]:

Cette fonction permet d'affecter une plage d'adresses à un seul périphérique de manière à éviter les conflits de bus.

I.5:b. Les bus du microcontrôleur [1]:

Un bus est un ensemble de lignes, transportant des informations codées binaires. Chacune de ces lignes est affectée d'un poids binaire.

C'est par l'intermédiaire de ces lignes que s'effectuent les échanges entre les différents éléments du système. On distingue 3 types de bus.

Le bus de données :

Ce bus transporte les données échangées par les différents périphériques externes du microcontrôleur.

C'est un ensemble de lignes bidirectionnelles de 8, 16 ou 32 voies. La taille du bus de données détermine l'appartenance du microprocesseur du système: Un microprocesseur avec un bus de données de 16 voies sera appelé « microprocesseur 16 bits ».

Ce bus est bidirectionnel; c'est à dire que les informations qu'il véhicule peuvent transiter:

_ Du microcontrôleur vers l'un de ses périphériques,

_ D'un périphérique vers le microcontrôleur.

Le bus d'adresse

A chaque mot de donnée correspond un numéro: l'adresse. Pour avoir accéder à une donnée, il suffira de présenter son adresse sur le bus d'adresse. De même, pour mémoriser une donnée, il faudra présenter sur le bus d'adresse, l'adresse à laquelle on désire stocker cette donnée.

Par conséquent, le bus d'adresse véhicule l'adresse qui spécifie l'origine ou la destination de l'information qui transite sur le bus de données.

Le bus d'adresse est un ensemble de lignes unidirectionnelles. La taille du bus d'adresse caractérise la capacité d'adressage du microprocesseur du système: Un microprocesseur qui a n fils d'adresse peut présenter, sur son bus d'adresse, 2n adresses distinctes.

Ce bus est unidirectionnel; c'est à dire que les informations qu'il véhicule transitent du microprocesseur vers l'un de ses périphériques. Le bus de contrôle.

C'est un ensemble de lignes transportant les différents signaux de commande et de synchronisation nécessaires pour le bon déroulement des échanges entre les divers éléments du système. Les lignes de ce bus ne sont pas affectées d'un poids binaire, contrairement aux lignes du bus de donnée et du bus d'adresse.

I.5:c. Les mémoires [1]:

Si les mémoires internes au microcontrôleur sont insuffisantes (programme de gestion trop important, les données temporaires à sauvegarder trop importantes,….), on choisira des mémoires externes de manière à compléter ou suppléer les mémoires internes au microcontrôleur.

I.5:d. Les périphériques optionnels [1]:

Si les périphériques internes au microcontrôleur ne sont pas suffisants alors on pourra ajouter des périphériques externes.

I.6: Applications –critères de choix [1]:

Les applications des microcontrôleurs sont innombrables .De nos jours tous les systèmes techniques autonomes devant gérer de nombreux périphériques sont gérés autour d'un microcontrôleur.

L'avantage d'un microcontrôleur est que l'on peut faire évoluer le fonctionnement du système technique en modifiant son programme.

I.7: Critères de choix d'un microcontrôleur [1]:

-les périphériques disponibles en interne,
-la capacité des mémoires programme et données,
-la possibilité de gérer des périphériques externes si nécessaire,
-la rapidité de calcul.

I.8: Type de microcontrôleur :

Plusieurs types de microcontrôleurs :

- **Microchip** : PIC ; familles 12Cxxx, 16Cxxx, 16Fxxx, 18Fxxx, …
- **Atmel**: AT; familles AT89Sxxxx, AT90xxxx, …
- **Philips**: P89C51RD2BPN, …
- **Motorola**: famille 68HCxxx,

I.8:a. Microchip :

Nom	Description	Fabricant
12C672	Spécifications De Programmation De Mémoire d'cEprom	Microchip
12CE673	8-Goupilles/microcontrôleur de 8 bits de CMOS avec de la mémoire ANALOGIQUE-numérique de convertisseur et de données d'cEeprom	Microchip
12F629	Les 8-Goupilles Flash-Ont basé Des Microcontrôleurs De 8-Bit CMOS	Microchip
12F629	8-Goupilles, microcontrôleur de 8-Bit CMOS avec de la mémoire ANALOGIQUE-NUMÉRIQUE de convertisseur et de données d'EEPROM	Microchip
16C554	Microcontrôleur De 8 bits EPROM-Basé de CMOS	Microchip
16C62X	Microcontrôleur De 8 bits EPROM-Basé de CMOS	Microchip
16F628	Microcontrôleurs De 8 bits Flash-Basés de CMOS	Microchip
16F84	Microcontrôleurs De 8 bits De la 18-goupille Flash/EEPROM	Microchip
16F84A	La 18-goupille A augmenté Le Microcontrôleur De 8 bits De Flash/EEPROM	Microchip
16F871	Microcontrôleurs INSTANTANÉS De 8 bits Des 28/40-Goupilles CMOS	Microchip

Tab. I. 1 : Microcontrôleurs Microchip

I.8:b. Atmel:

Devices	F.max (MHz)	CPU Core	Power Supply (V)	Pb-Free Packages
AT91FR40162S	75	ARM7TDMI	1.65 -1.95 Core 2.7-3.6 IO	TFBGA 121
AT91M40800	40	ARM7TDMI	1.8-3.6	LQFP 100
AT91R40008	75	ARM7TDMI	1.65-1.95 Core 2.7-3.6 IO	LQFP 100
AT91RM3400	66	ARM7TDMI	1.65-1.95 Core 1.65-3.6 IO	LQFP 100
AT91SAM7A3	60	ARM7TDMI	3.0-3.6	LQFP 100
AT91SAM7S256	55	ARM7TDMI	3.0-3.6	QFN 64 LQFP 64

AT91SAM7S32	55	ARM7TDMI	3.0-3.6	QFN 48 LQFP 48
AT91SAM7SE256	48	ARM7TDMI	3.0-3.6	LBGA 144 LQFP 128
AT91SAM7SE32	48	ARM7TDMI	3.0-3.6	LBGA 144 LQFP 128
AT91SAM7SE512	48	ARM7TDMI	3.0-3.3	LBGA 144 LQFP 128
AT91SAM7X128	55	ARM7TDMI	3.0-3.6	LQFP 100
AT91SAM7XC256	55	ARM7TDMI	3.0-3.6	LQFP 100
AT91SAM9260	180	ARM926EJ-S	1.65-1.95 Core 3.0-3.6 IO	PQFP 208 LFBGA 217

Tab. I. 2: Microcontrôleurs Atmel

I.8:c. Philips :

- **Microcontrôleur Philips 80C552**

Microcontrôleur Philips cmos 8 bits dérivé du 80C51, remplaçant le PCB80C552-5-24WP. Caractéristiques: 256 octets de RAM - 64 K de programme - 64 K de données - 40 lignes d'E/S bidirectionnelles et adressables individuellement - 8 lignes d'entrées analogiques ou digitales - convertisseur A/D 10 bits 8 canaux - 2 sorties PWM 8 bits - 3 timer/compteurs 16 bits - watchdog timer - full duplex USART - interface I²C - 15 interruptions internes (2 niveaux de priorité) - 6 interruptions externes – Fréquence d'horloge: 24MHz. Alim.:5V. Boîtier PLCC68.

- **Microcontrôleur Philips 87xxx:**

- **MICROCONTROLEURS(87C75x)**
  Microcontrôleurs Philips cmos 8 bits série 87C750/1. Caractéristiques: 64 octets de RAM - 19 lignes d'E/S bidirectionnelles et adressables individuellement - sorties pour la commande directe de LEDs - timer/compteur 16 bits - 5 interruptions internes - 2 interruptions externes - Alimentation: 5 V. Boîtiers DIL24 et DIL28.

- **MICROCONTROLEURS(87C5x)**
  Microcontrôleurs Philips 8 bits dérivés du 80C51 à faible consommation. 32 E/S. 3 temporisateurs/compteurs 16 bits. 6 sources d'interruption. Port E/S série. UART full duplex. Alimentation de 2.7V à 5.5 V.

- **MICROCONTROLEURS(87LPCxxx)**
  Microcontrôleurs Philips 8 bits dérivés du 8051 - alimentation de 2.7 à 6.0 Vcc - watchdog intégré - 2 timers/compteurs 16 bits - 2 comparateurs analogiques - UART full duplex - bus I²C.

- **MICROCONTROLEUR(P89C51RD2HBP)**
  Microcontrôleur Philips ISP cmos 8 bits dérivé du 80C51 - remplace le P89C81RD+IN - programmation possible sur le circuit sans démonter le composant - 512 x 8 octets de RAM - 64 kB de mémoire flash - 4 ports R/S 8 bits - 3 temporisateurs 16 bits - Fréquence d'horloge: 33 MHz - boîtier DIP40.

I.8:d. Motorola:

Nom de partie	Description	Fabricant
68HC05B16	Microcontrôleur	Freescale (Motorola)
68HC05B32	Microcontrôleur	Freescale (Motorola)
68HC05B6	Microcontrôleur	Freescale (Motorola)
68HC05BD5	Microcontrôleur	Freescale (Motorola)
68HC05C8A	Microcontrôleur	Freescale (Motorola)

68HC05J1A	Microcontrôleur	Freescale (Motorola)
68HC05J5A	Microcontrôleur	Freescale (Motorola)
68HC05JB3	Microcontrôleur	Freescale (Motorola)

Tab. I. 3: Microcontrôleurs Motorola

I.9: Definition d'un PIC :

Un PIC est un microcontrôleur qui signifie (Peripheral Interface Controller), c'est une unité de traitement d'information de type microprocesseur à laquelle on a ajouté des périphériques internes permettant de faciliter l'interfaçage avec le monde extérieur sans nécessiter l'ajout de composants externes.

Les Pics sont des composants RISC (Reduced Instructions Set Computing) ou encore composant à jeu d'instructions réduit. L'avantage est que plus on réduit le nombre d'instructions, plus facile et plus rapide en est le décodage, et plus vite le composant fonctionne.

Alors, les microcontrôleurs 16F84 et 16F84A représentent le cerveau de notre projet, d'où nous allons les étudier.

I.9:a. Les différentes familles des PIC :

La famille des pics est subdivisée en trois grandes familles :

- Base-line : c'est une famille qui utilise des mots d'instructions de 12 bits.
- Mid-range : c'est une famille qui utilise des mots de 14 bits (dont font partie les 16F84,16F84A et 16F877)
- High-end : c'est une famille qui utilise des mots de 16 bits.

I.9:b. Identification d'un Pic

Pour identifier un PIC, on utilise simplement son numéro :
16 : indique la catégorie du PIC, c'est un Mid-range.
L : indique qu'il fonctionne avec une plage de tension beaucoup plus tolérante.
C : indique que la mémoire programme est un EPROM ou une EEPROM.
CR ou F : indique le type de mémoire ; CR(ROM) ou F (FLASH).
XX : représente la fréquence d'horloge maximale que le PIC peut recevoir. Une dernière indication qu'on le trouve est le type de boîtie

Conclusion

Les microcontrôleurs sont et continueront à être largement utilisés pour les applications de régulation et de commande de processus industriels.

Pratiquement tous les fabricants de microprocesseurs (Microship, Motorola, Intel, Hitachi, Texas Instrument, Toshiba, ST Microélectronique-ex SGS-Thomson, etc.) proposent une ou plusieurs gammes de microcontrôleurs .

Pour programmer un microcontrôleur; il est nécessaire de connaître sa structure interne : registres, mémoires, ports d'entrées sorties, et toutes leurs possibilités.

La commande des systèmes électriques modernes se base essentiellement sur les microcontrôleurs; pour cela un bon spécialiste de commande doit impérativement savoir utiliser les microcontrôleurs pour commander des processus industriels.

Chapitre II

le microcontrôleur 16F84

Introduction:

Dans ce chapitre, nous allons présenter les caractéristiques et les composants du microcontrôleur utilisé dans notre étude et notre projet.

II.1: Qu'est-ce qu'un PIC[3]:

Un PIC est un microcontrôleur de chez Microchip. Ses caractéristiques principales sont : Séparation des mémoires de programme et de données (architecture Harvard) : On obtient ainsi une meilleure bande passante et des instructions et des données pas forcément codées sur le même nombre de bits.

Communication avec l'extérieur seulement par des ports : il ne possède pas de bus d'adresses, de bus de données et de bus de contrôle comme la plupart des microprocesseurs.

Utilisation d'un jeu d'instructions réduit, d'où le nom de son architecture : RISC (Reduced Instructions Set Construction). Les instructions sont ainsi codées sur un nombre réduit de bits, ce qui accélère l'exécution (1 cycle machine par instruction sauf pour les sauts qui requirent 2 cycles). En revanche, leur nombre limité oblige à se restreindre à des instructions basiques, contrairement aux systèmes d'architecture CISC (Complex Instructions Set Construction) 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

Un PIC est identifié par un numéro de la forme suivant : xx(L)XXyy –zz

- xx : Famille du composant (12, 14, 16, 17, 18)
- L : Tolérance plus importante de la plage de tension
- XX : Type de mémoire de programme
  - C - EPROM ou EEPROM
  - CR - PROM
  - F - FLASH
- yy : Identification
- zz : Vitesse maximum du quartz

Nous utiliserons un PIC 16F84 –10, soit :

- 16 : Mid-Line
- F : FLASH
- 84 : Type
- 10 : Quartz à 10 MHz au maximum

II.2: PIC 16F84 [3]:

Device	Program Memory (words)	Data RAM (bytes)	Data EEPROM (bytes)	Max. Freq (MHz)
PIC16F83	512 Flash	36	64	10
PIC16F84	1 K Flash	68	64	10
PIC16CR83	512 ROM	36	64	10
PIC16CR84	1 K ROM	68	64	10

Figure II.1 : Liste des composants présentés dans la documentation n°DS30430C. [4]

Il s'agit d'un microcontrôleur 8 bits à 18 pattes. La documentation technique n°DS30430C porte sur plusieurs composants (Figure II.1).

Principales caractéristiques :

- 35 instructions
- Instructions codées sur 14 bits
- Données sur 8 bits
- 1 cycle machine par instruction, sauf pour les sauts (2 cycles machine)
- Vitesse maximum 10 MHz soit une instruction en 400 ns (1 cycle machine = 4 cycles d'horloge)
- 4 sources d'interruption
- 1000 cycles d'effacement/écriture pour la mémoire flash, 10.000.000 pour la mémoire de donnée EEPROM

II.2:a. Brochage et fonction des pattes[3]:

La Figure II.2 montre le brochage du circuit. Les fonctions des pattes sont les suivantes :

- V_{SS}, V_{DD} : Alimentation
- OSC1,2 : Horloge
- RA0-4 : Port A
- RB0-7 : Port B
- T0CKL : Entrée de comptage
- INT : Entrée d'interruption
- MCLR : Reset : 0V

Choix du mode programmation : 12V - 14V exécution : 4.5V - 5.5V

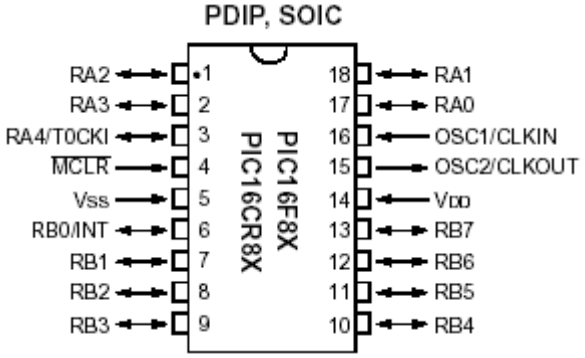


Figure II.2 : Brochage du circuit. [4]

II.2:b. Architecture générale [3]:

La Figure II.3 présente l'architecture générale du circuit. Il est constitué des éléments suivants :

- un système d'initialisation à la mise sous tension (power-up timer, …)
- un système de génération d'horloge à partir du quartz externe (timing génération)
- une unité arithmétique et logique (ALU)
- une mémoire flash de programme de 1k "mots" de 14 bits
- un compteur de programme (program counter) et une pile (stack)
- un bus spécifique pour le programme (program bus)
- un registre contenant le code de l'instruction à exécuter
- un bus spécifique pour les données (data bus)
- une mémoire RAM contenant
- les SFR
- 68 octets de données
- une mémoire EEPROM de 64 octets de données
- 2 ports d'entrées/sorties
- un compteur (timer)
- un chien de garde (watchdog)

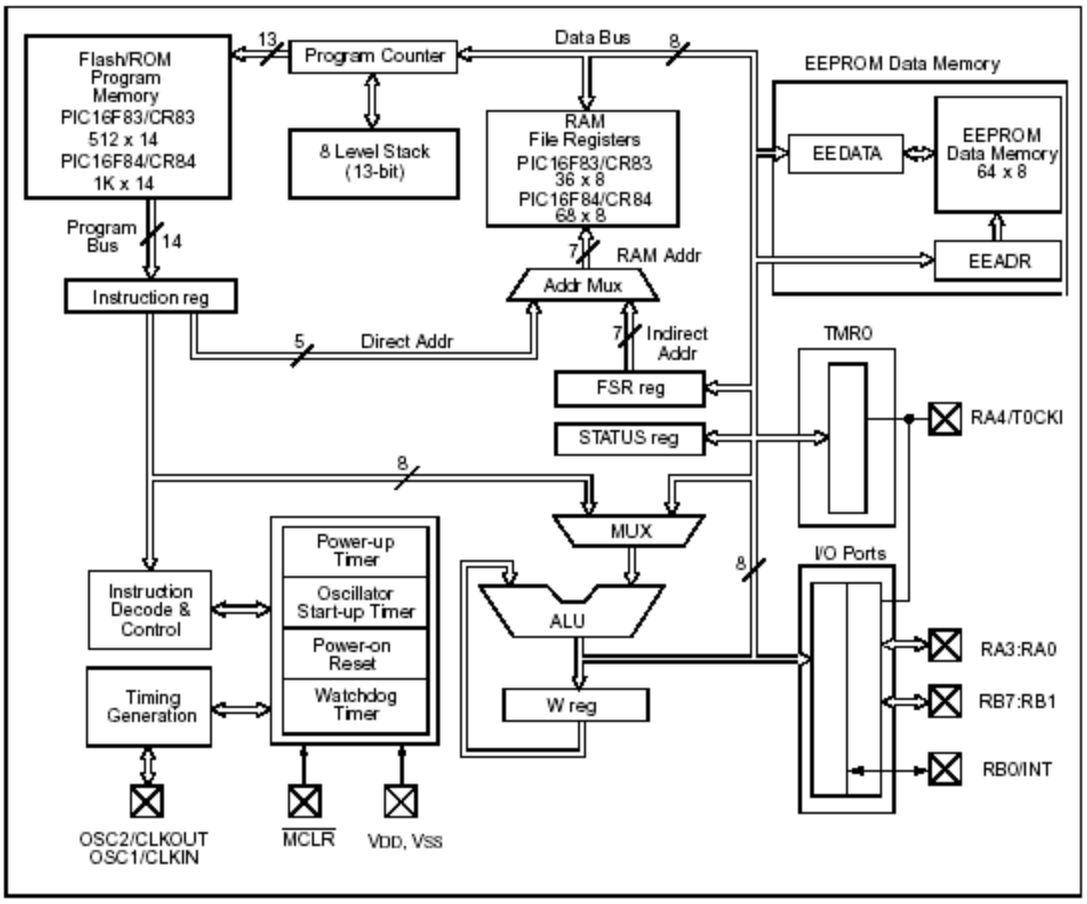


Figure II.3 : Architecture générale du PIC 16F8X. [4]

II.3: Organisation de la mémoire[3]:

Le PIC contient de la mémoire de programme et de la mémoire de données. La structure Harvard des PICs fournit un accès séparé à chacune. Ainsi, un accès aux deux est possible pendant le même cycle machine.

II.3:a. Mémoire de programme[3]:

C'est elle qui contient le programme à exécuter. Ce dernier est téléchargé par liaison série . La Figure II.4 montre l'organisation de cette mémoire. Elle contient 1k "mots" de 14 bits dans le cas du PIC 16F84, même si le compteur de programme (PC) de 13 bits peut en adresser 8k. Il faut se méfier des adresses images ! L'adresse 0000h contient le vecteur du reset, l'adresse 0004h l'unique vecteur d'interruption du PIC.
La pile contient 8 valeurs. Comme le compteur de programme, elle n'a pas d'adresse dans la plage de mémoire. Ce sont des zones réservées par le système.

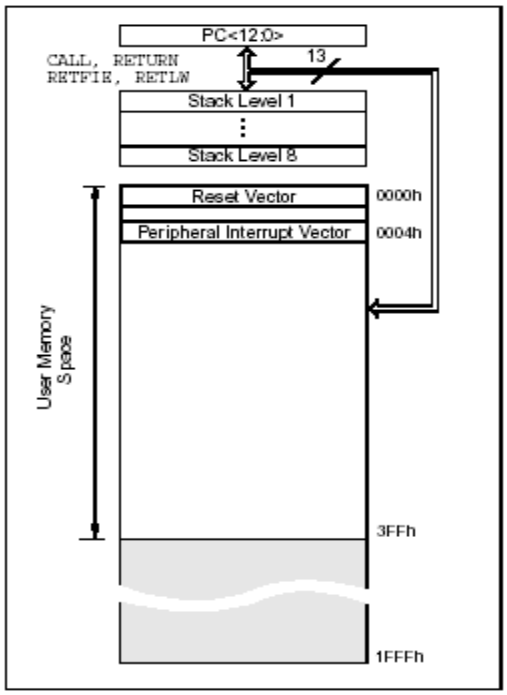


Figure II.4 : Organisation de la mémoire de programme et de la pile. [4]

II.3:b. Mémoire de données[3]:

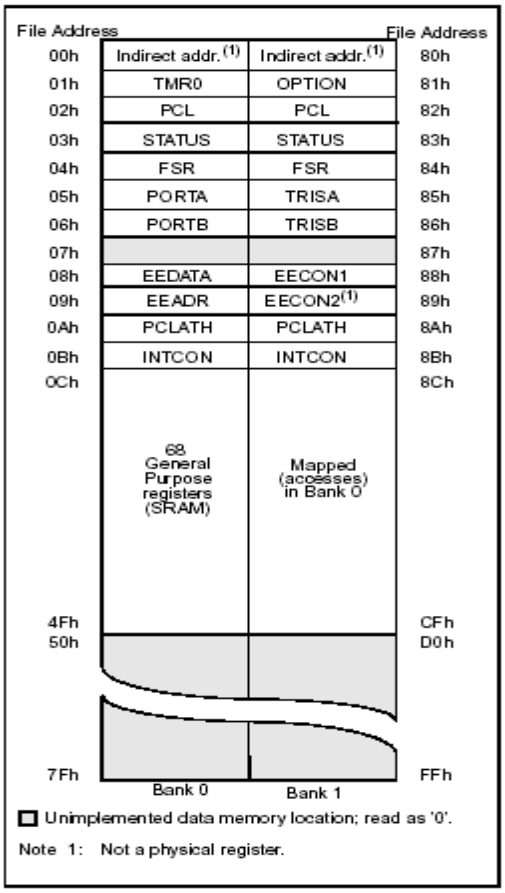


Figure II.5 : Organisation de la mémoire de données. [4]

Elle se décompose en deux parties de RAM (Figure II.5) et une zone EEPROM. La première contient les SFRs (Special Function Registers) qui permettent de contrôler les opérations sur le circuit. La seconde contient des registres généraux, libres pour l'utilisateur. La dernière contient 64 octets.

Comme nous le verrons dans le paragraphe IV, les instructions orientées octets ou bits contiennent une adresse sur 7 bits pour désigner l'octet avec lequel l'instruction doit travailler. D'après la Figure II.5, l'accès au registre TRISA d'adresse 85h, par exemple, est impossible avec une adresse sur 7 bits. C'est pourquoi le constructeur a défini deux banques. Le bit RP0 du registre d'état (STATUS.5) permet de choisir entre les deux. Ainsi, une adresse sur 8 bits est composée de RP0 en poids fort et des 7 bits provenant de l'instruction à exécuter.

II.3:b.1- Registres généraux[3]:

Ils sont accessibles soit directement soit indirectement à travers les registres FSR et INDF .

II.3:b.2- Registres spéciaux - SFRs [3]:

Ils permettent la gestion du circuit. Certains ont une fonction générale, d'autres une fonction spécifique attachée à un périphérique donné. La Figure II.6 donne la fonction de chacun des bits de ces registres. Ils sont situés de l'adresse 00h à l'adresse 0Bh dans la banque 0 et de l'adresse 80h à l'adresse 8Bh dans la banque 1. Les registres 07h et 87h n'existent pas.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note 3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---- ----	---- ----
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---- ----	---- ----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---- ----	---- ----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition.
Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.
2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ reset.
3: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

Figure II.6 : Description des SFR. [4]

INDF (00h - 80h) : Utilise le contenu de FSR pour l'accès indirect à la mémoire .
TMR0 (01h) : Registre lié au compteur .

PCL (02h - 82h) : Contient les poids faibles du compteur de programmes (PC). Le registre PCLATH (0Ah-8Ah) contient les poids forts.

STATUS (03h - 83h) : Il contient l'état de l'unité arithmétique et logique ainsi que les bits de sélection des banques (Figure II.7).

FSR (04h - 84h) : Permet l'adressage indirect

PORTA (05h) : Donne accès en lecture ou écriture au port A, 5 bits. Les sorties sont à drain ouvert. Le bit 4 peut être utilisé en entrée de comptage.

PORTB (06h) : Donne accès en lecture ou écriture au port B. Les sorties sont à drain ouvert. Le bit 0 peut être utilisé en entrée d'interruption. EEDATA (08h) : Permet l'accès aux données dans la mémoire EEPROM.

EEADR (09h) : Permet l'accès aux adresses de la mémoire EEPROM.

PCLATCH (0Ah - 8Ah) : Donne accès en écriture aux bits de poids forts du compteur de programme.

INTCON (0Bh - 8Bh) : Masque d'interruptions .

OPTION_REG (81h) : Contient des bits de configuration pour divers périphériques.

TRISA (85h) : Indique la direction (entrée ou sortie) du port A.

TRISB (86h) : Indique la direction (entrée ou sortie) du port B.

EECON1 (88h) : Permet le contrôle d'accès à la mémoire EEPROM .

EECON2 (89h) : Permet le contrôle d'accès à la mémoire EEPROM .

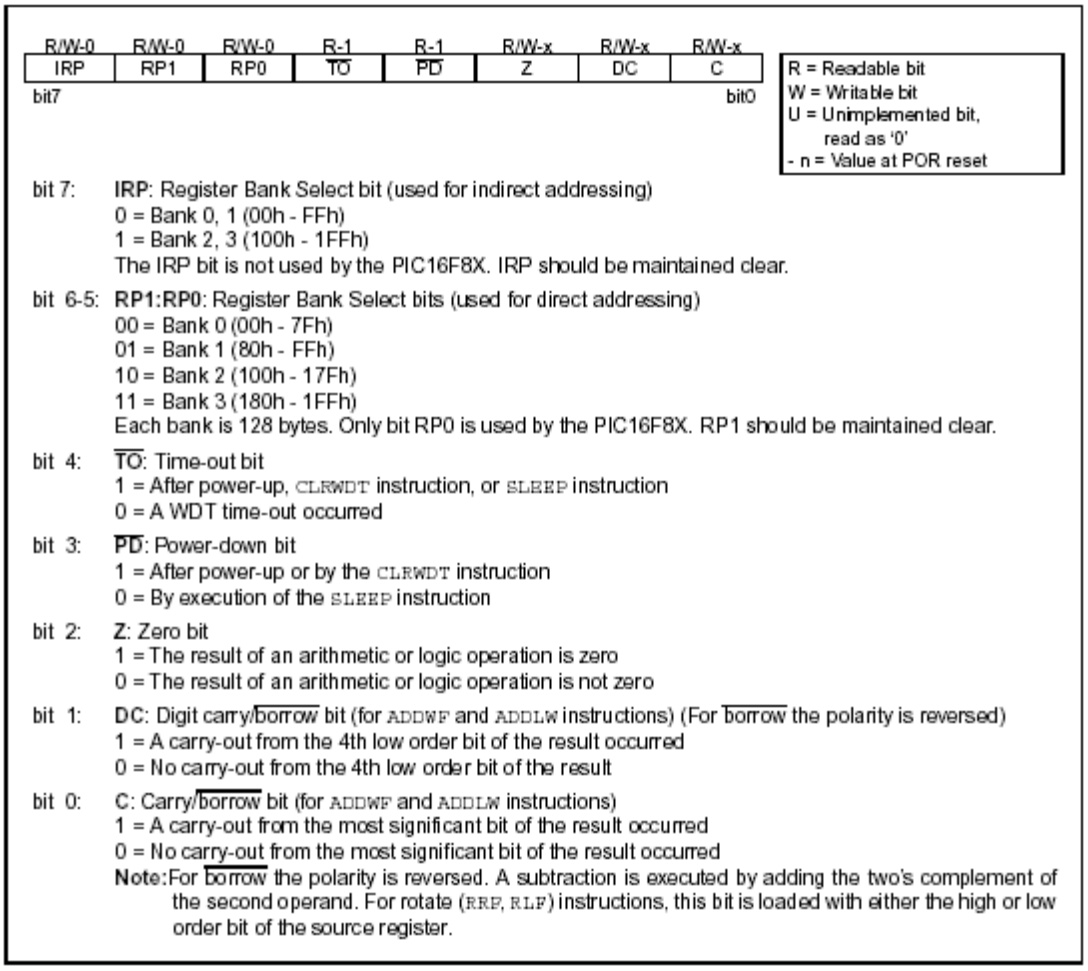


Figure II.7 : Registre d'étai du PIC - STATUS. [4]

II.3:b.3-Mémoire EEPROM [3]:

Le PIC possède une zone EEPROM de 64 octets accessibles en lecture et en écriture par le programme. On peut y sauvegarder des valeurs, qui seront conservées même si l'alimentation est éteinte, et les récupérer lors de la mise sous tension. Leur accès est spécifique et requiert l'utilisation de registres dédiés. La lecture et l'écriture ne peut s'exécuter que selon des séquences particulières décrite au paragraphe .

II.4: Modes d'adressages[3]:

On ne peut pas concevoir un programme qui ne manipule pas de données. Il existe trois grands types d'accès à une donnée ou modes d'adressage :

- Adressage immédiat : La donnée est contenue dans l'instruction.
- Adressage direct : La donnée est contenue dans un registre.

direct : L'adresse de la donnée est contenue dans un pointeur.

II.4:a. Adressage immédiat[3]:

La donnée est contenue dans l'instruction.
Exemple : movlw 0xC4 ; Transfert la valeur 0xC4 dans W

II.4:b. Adressage direct[3]:

La donnée est contenue dans un registre. Ce dernier peut être par un nom (par exemple W) ou une adresse mémoire.

Exemple : movf 0x2B, 0 ; Transfert dans W la valeur contenue à l'adresse 0x2B**.**
! L'adresse 0x2B peut correspondre à 2 registres en fonction de la banque choisie (Figure II.8). Le bit RP0 permet ce choix, le bit RP1 étant réservé pour les futurs systèmes à 4 banques.

II.4:c. Adressage indirect [3]:

L'adresse de la donnée est contenue dans un pointeur. Dans les PIC, un seul pointeur est disponible pour l'adressage indirect : FSR. Contenu à l'adresse 04h dans les deux banques, il est donc accessible indépendamment du numéro de banque. En utilisant l'adressage direct, on peut écrire dans FSR l'adresse du registre à atteindre. FSR contenant 8 bits, on peut atteindre les deux banques du PIC 16F84. Pour les PIC contenant quatre banques, il faut positionner le bit IRP du registre d'état qui sert alors de 9ème bit d'adresse (Figure II.8).

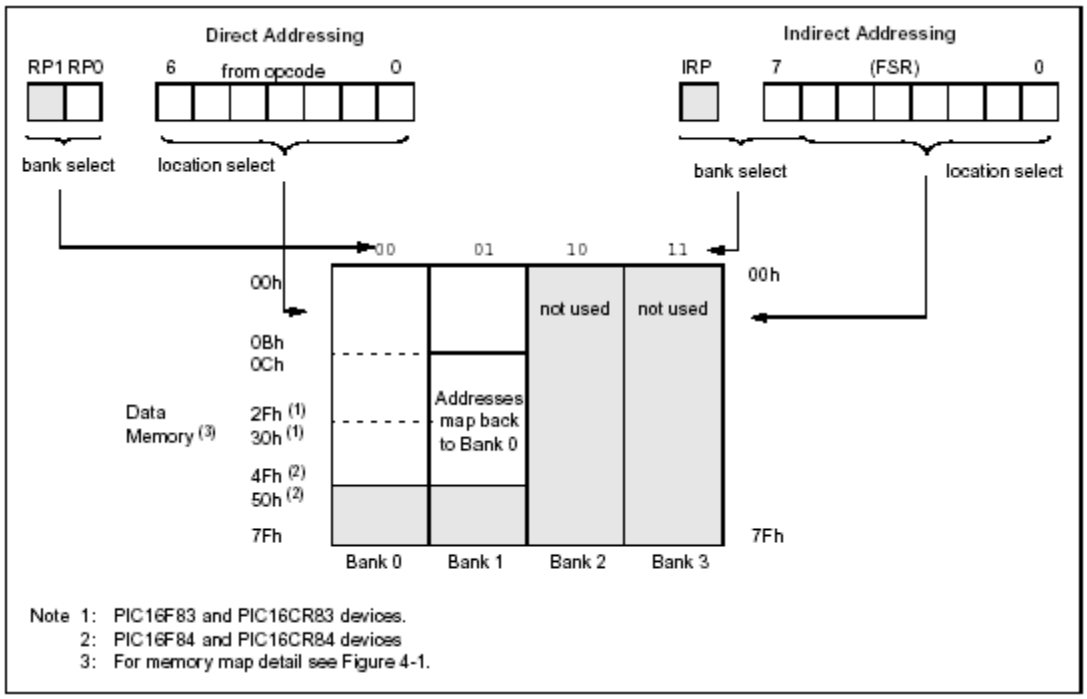


Figure II.8 : Adressages direct et indirect à la mémoire de données. [4]

L'accès au registre d'adresse contenue dans FSR se fait en utilisant le registre INDF. Il se trouve à l'adresse 0 dans les deux banques. Il ne s'agit pas d'un registre physique. On peut le voir comme un autre nom de FSR, utilisé pour accéder à la donnée elle-même, FSR servant à choisir l'adresse.

Exemple : movlw 0x1A ; Charge 1Ah dans W
movwf FSR ; Charge W, contenant 1Ah, dans FSR
movf INDF, 0 ; Charge la valeur contenue à l'adresse 1Ah dans W

II.5: Ports d'entrées/Sorties[3]:

Le PIC 16F84 est doté de deux ports d'entrées/Sorties appelés PortA et PortB.

II.5:a. Port A [3]:

Il comporte 5 pattes d'entrée/sortie bidirectionnelles, notées RAx avec x={0,1,2,3,4} sur le brochage du circuit (Figure II.2). Le registre PORTA, d'adresse 05h dans la banque 0, permet d'y accéder en lecture ou en écriture. Le registre TRISA, d'adresse 85h dans la banque 1, permet de choisir le sens de chaque patte (entrée ou sortie) : un bit à 1 positionne le port en entrée, un bit à 0 positionne le port en sortie.

La Figure II.9 donne le câblage interne d'une patte du port A :

- "Data Latch" : Mémorisation de la valeur écrite quand le port est en sortie.
- "TRIS Latch" : Mémorisation du sens (entrée ou sortie) de la patte.
- "TTL input buffer" : Buffer de lecture de la valeur du port. La lecture est toujours réalisée sur la patte, pas à la sortie de la bascule d'écriture.
- Tansistor N : En écriture : Saturé ou bloqué suivant la valeur écrite.
  En lecture : Bloqué.
- Transistor P : Permet d'alimenter la sortie.

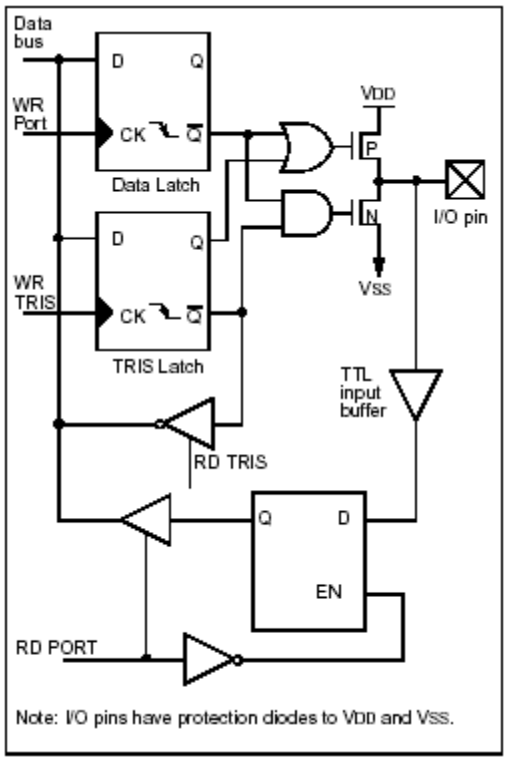


Figure II.9 : Câblage interne d'une patte du port A. [4]

La patte RA4 peut aussi servir d'entrée de comptage pour le timer0.

II.5:b. Port B [3]:

Il comporte 8 pattes d'entrée/sortie bidirectionnelles, notées RBx avec x={0,1,2,3,4,5,6,7} sur le brochage du circuit (Figure II.2).

Le registre PORTB, d'adresse 06h dans la banque 0, permet d'y accéder en lecture ou en écriture. Le registre TRISB, d'adresse 86h dans la banque 1, permet de choisir le sens de chaque patte (entrée ou sortie) : un bit à 1 positionne le port en entrée, un bit à 0 positionne le port en sortie.

Le câblage interne d'une porte du port B ressemble beaucoup à celui du port A (Figure II.10). On peut noter la fonction particulière pilotée par le bit RBPU (OPTION_REG.7) qui permet d'alimenter (RBPU=0) ou non (RBPU=1) les sorties.

Les quatre bits de poids fort (RB7-RB4) peuvent être utilisés pour déclencher une interruption sur changement d'état.
RB0 peut aussi servir d'entrée d'interruption externe.

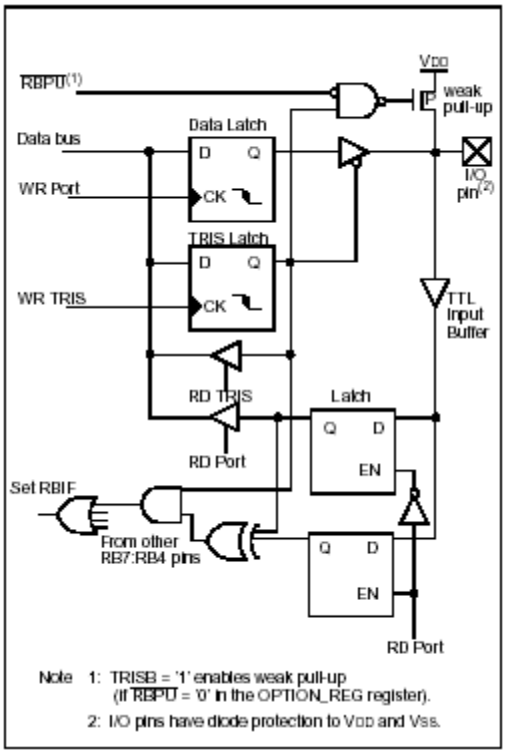


Figure II.10 : Câblage interne d'une patte du port B. [4]

II.6: Compteur [3]:

Le PIC 16F84 est doté d'un compteur 8 bits. La Figure II.10 en donne l'organigramme.

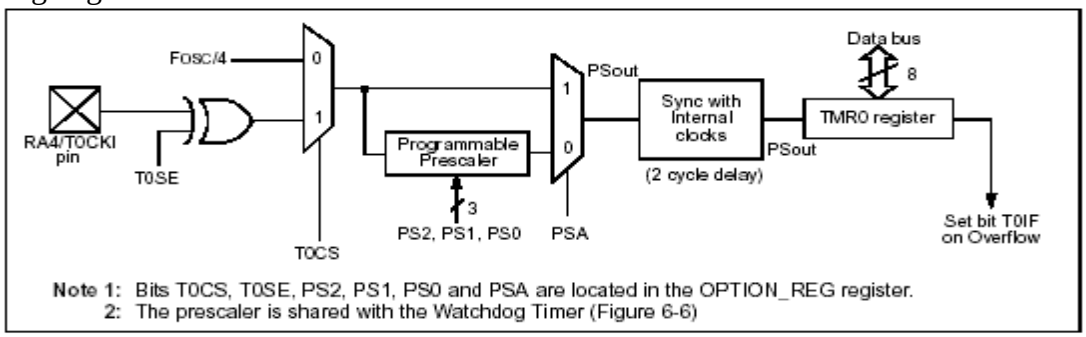


Figure II.11 : Organigramme du Timer0. [4]

II.6:a. Registre TMR0[3]:

C'est le registre de 8 bits qui donne la valeur du comptage réalisé. Il est accessible en lecture et en écriture à l'adresse 01h dans la banque 0.

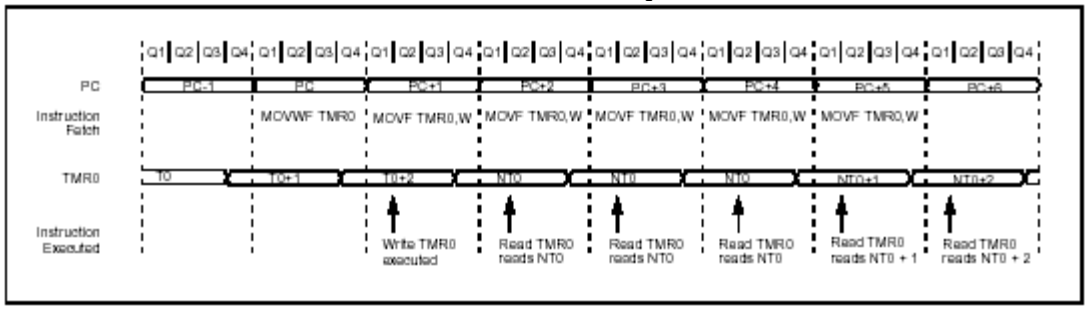


Figure II.12 : Prise en compte de l'écriture dans le registre TMR0. [4]

Lors d'une écriture dans TMR0, le comptage est inhibé pendant deux cycles machine (Figure II.12). Si l'on veut déterminer un temps avec précision, il faut tenir compte de ce retard au démarrage.

II.6:b. Choix de l'horloge [3]:

Le timer 0 peut fonctionner suivant deux modes en fonction du bit T0CS (OPTION_REG.5). En mode timer (T0CS=0), le registre TMR0 est incrémenté à chaque cycle machine (si le pré-diviseur n'est pas sélectionné).

En mode compteur (T0CS=1), le registre TMR0 est incrémenté sur chaque front montant ou chaque front descendant du signal reçu sur la broche RA4/T0CKl en fonction du bit T0SE (OPTION_REG.4). Si T0SE=0, les fronts montants sont comptés, T0SE=1, les fronts descendants sont comptés.

II.6:c. Pré-diviseur [3]:

En plus des deux horloges, un pré-diviseur, partagé avec le chien de garde, est disponible. La période de l'horloge d'entrée est divisée par une valeur comprise entre 2 et 256 suivant les bits PS2, PS1 et PS0 (respectivement OPTION_REG.2, .1 et .0) (Figure II.13). Le bit PSA (OPTION_REG.3) permet de choisir entre la pré-division de timer0 (PSA=0) ou du chien de garde (PSA=1).

PSA	PS2	PS1	PS0	/tmr0	/WD
0	0	0	0	2	1
0	0	0	1	4	1
0	0	1	0	8	1
0	0	1	1	16	1
0	1	0	0	32	1
0	1	0	1	64	1
0	1	1	0	128	1
0	1	1	1	256	1
1	0	0	0	1	1
1	0	0	1	1	2
1	0	1	0	1	4
1	0	1	1	1	8
1	1	0	0	1	16
1	1	0	1	1	32
1	1	1	0	1	64
1	1	1	1	1	128

Figure II.13 : Valeurs du pré-diviseur en fonction de PSA, PS2, PS2 et PS0. [4]

II.6:d. Fin de comptage et interruption[3]:

Le bit T0IF (INTCON.2) est mis à 1 chaque fois que le registre TMR0 passe de FFh à 00h. On peut donc tester ce bit pour connaître la fin de comptage. Pour compter 50 événements, il faut donc charger TMR0 avec la valeur 256-50=206 et attendre le passage de T0IF à 1. Cette méthode est simple mais bloque le processeur dans une boucle d'attente.

On peut aussi repérer la fin du comptage grâce à l'interruption que peut générer T0IF en passant à 1 . Le processeur est ainsi libre de travailler en attendant cet événement.

II.6:e. Registres utiles à la gestion de timer0 [3]:

Plusieurs registres ont été évoqués dans ce paragraphe. Ils sont synthétisés dans la Figure II.14.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
01h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 0000
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not associated with Timer0.

Figure II.14: Registres utiles à la gestion de timer0. [4]

Conclusion

En conclusion, nous pouvons dire que le microcontrôleur 16F84 peut bien jouer le rôle d'unité de commande pour notre système.
Pour programmer le microcontrôleur, nous devons connaître les langages de programmation et l'utilisation d'un programme de simulation avant l'application réelle.

Chapitre III

les logiciels utilisés

Introduction

Pour la simplicité et la facilité de la programmation, plusieurs langages ont été évolués dans le temps. En cherchant le compilateur le plus adapté aux microcontrôleurs PIC, on trouve le MikroBasic , MikroC et MikroPascal ,et pour la simulation on trouve le logiciel(PROTEUS ISI) .

III.1: Définition du logiciel "PROTEUS"

Proteus est une suite de logiciels permettant la CAO électronique éditée par la société Lab center Electronics. Proteus est composé de deux logiciels principaux : ISIS, permettant entre autres la création de schémas et la simulation électrique, et ARES, dédié à la création de circuits imprimés.Grâce à des modules additionnels, ISIS figure

III.1:a. logiciel proteus [2]:

Il est également capable de simuler le comportement d'un microcontrôleur (PIC, Atmel, 8051, ARM, HC11...) et son interaction avec les composants qui l'entourent.

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: ISIS, ARES, PROSPICE et VSM.

III.1:b. Présentation générale[2]:

Cette suite logicielle est très connue dans le domaine de l'électronique. De nombreuses entreprises et organismes de formation 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.

III.1:c. ISIS[2]:

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.

III.1:d. SAISIE DU SCHEMA[2]:

III.1:d.1- Démarrer le logiciel

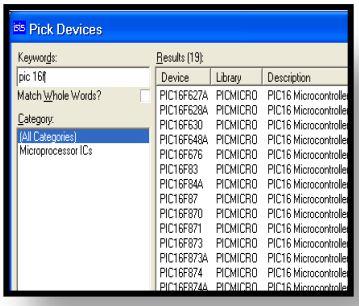
Lancer depuis le raccourci du bureau ou du menu Démarrer de votre ordinateur le Logiciel Proteus – ISIS :

III.1:d.2- Création de la liste des composants

- Sélectionner l'icone composant :
- Dans la fenêtre du sélectionneur d'objets, cliquer sur P : 

- Une nouvelle fenêtre (Pick Devices) de choix de composants s'ouvre :

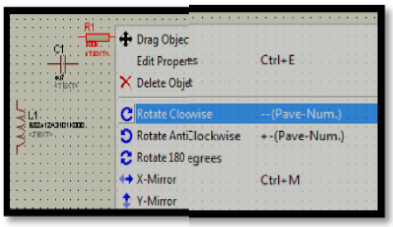
Trouver les composants désirés à partir d'un mot clé, ou Dans la catégorie proposée, puis dans la sous-catégorie (par défaut choisissez dans la sous-catégorie Générique.



- Cliquer une fois pour faire apparaitre le composant, s'il convient double cliquer pour qu'il s'inscrive sur la liste de travail DEVICES.
- Procéder ainsi, pour placer dés le début du TP tous les composants que vous aurez besoin par la suite sur la liste de travail, puis fermer la fenêtre en cliquant sur OK.
- Dans notre cas, il faut sélectionner les composants « RES », « CAP » et « SWITCH », qui se mettent dans le « panier »

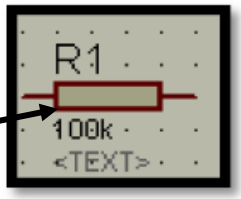
III.1:d.3- Placement des composants sur le schéma[2]:

- Sélectionner le composant à placer dans la liste faite précédemment et le placer dans la zone de travail de la fenêtre d'édition.
- Eventuellement le tourner ou lui faire faire une symétrie à l'aide des outils d'orientation du menu de la boite à outils :



Les composants placés, il faut définir leurs paramètres : Cliquer gauche sur le composant, le symbole devient rouge, recliquer gauche, une fenêtre ,pour éditer les propriétés ,qui s'ouvre.

Exemple : (R1,C1,L1) et affectez les valeurs suivantes : R1 = 1k et C1 = 10nF…..

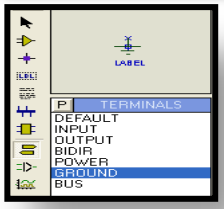


Clique 2fois et Change la valeur de R1

III.1:d.4- Placement des différentes connexions: externes:

Pour GROUND ; OUTPUT ; INPUT…

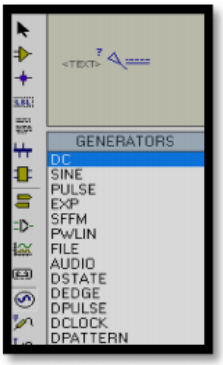
>Après avoir cliqué sur cette fonction Sélectionnez la connexion et après l'avoir mise dans le « panier », la placer sur



le schéma. Pour notre schéma, nous utiliserons la masse (GROUND) .

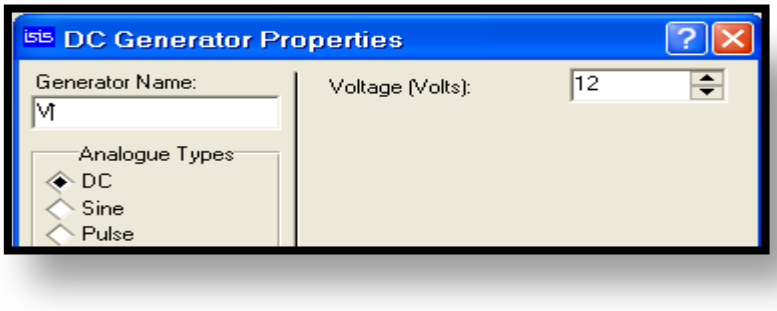
III.1:d.5- Placement des alimentations et des générateurs:

Il faut maintenant placer les différentes alimentations ou générateurs utilisés.



*Cliquer sur et sélectionner un générateur (DC, SINE, PULSE…).
*Ensuite cliquer sur le schéma pour le placer (ici Pulse = rectangle).
*Les alimentations ou générateurs placés, il faut leur donner un nom et régler les différents paramètres qui sont fonctions du type de Générateurs choisi.
Cliquer gauche sur le générateur, le symbole devient rouge, recliquer gauche, une fenêtre, pour éditer les propriétés, qui s'ouvre.

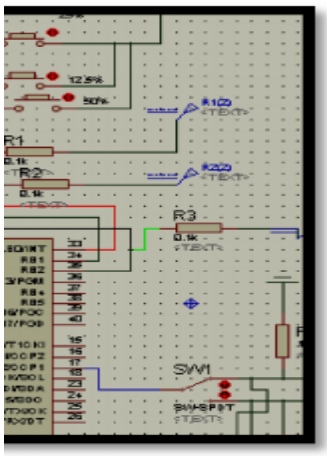
- Donner un nom (V).
- Régler la tension :



III.1:d.6- Réalisations des connexions:

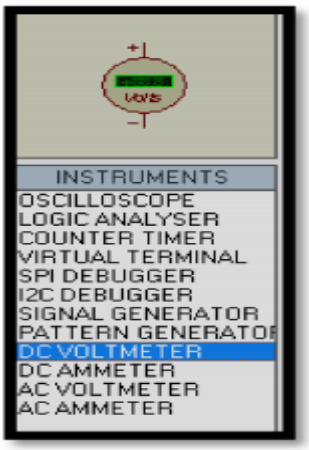
Il nous reste à relier les connexions entre les différents symboles.

- Sélectionner dans le menu : Aller sur l'extrémité d'un composant, cliqué, le début de la liaison est créée, puis aller cliquer sur le composant à relier pour finir la liaison, le logiciel Positionne seul le chemin.

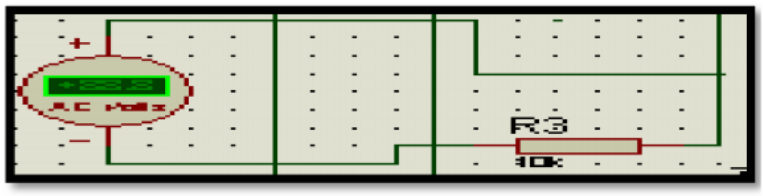


III.1:e. placement des instruments de mesure:

Il faut choisir le ou les appareils de mesure que l'on souhaite utiliser en cliquant sur l'icone  :



Pour un voltmètre continu (DC VOLTMETER), comme pour un voltmètre réel, il mesure une DDP (Différence de potentiel) entre les 2 points connectés, il faut donc le relier à 2 points.



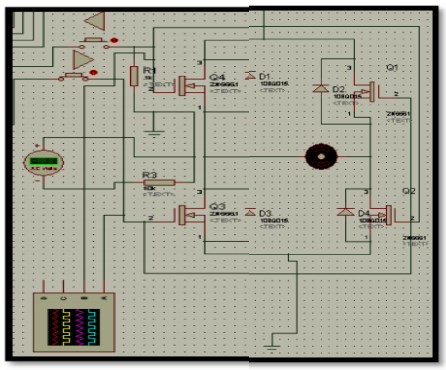
III.1:f. animation-simulation du fonctionnement

Pour lancer l'animation. Cliquer sur la flèche, comme sur un appareil multimédia,

Fermer l'interrupteur K en cliquant sur la flèche de commande, il est maintenant possible de lire les informations données par le voltmètre en temps réel.
Quelle est la valeur affichée par le voltmètre ?

III.1:g. utilisation avancée des appareils de mesure:

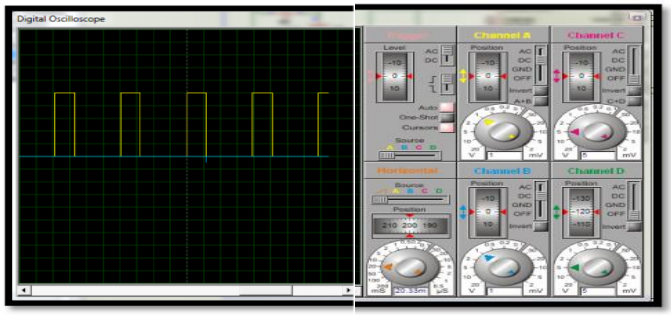
Le générateur VE devra être paramétré ainsi :
VE compris entre 0 V et 12 V ; forme rectangulaire
de rapport cyclique (soit une largeur d'impulsion de 4% 25%.50% ,75% ,98%) ; fréquence = 1MHz.
Placer les instruments de mesures suivants) ; un voltmètre (DC VOLTMETER) et un oscilloscope.



- Lancer la simulation.

Si l'écran de l'oscilloscope n'apparait pas, vous devez arrêter la simulation, cliquer gauche une fois sur l'oscillo (il devient rouge) et aller sur le menu Mise au point puis valider sur Réinitialiser fenêtres popup. Vous pouvez alors relancer la simulation, elle se déroule en temps réel. Procéder ensuite aux réglages :
*Mettre les voies C et D de l'oscilloscope sur OFF.

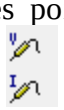
*Regler la base de temps de l'oscilloscope pour avoir plusieurs périodes sur l'ecran.
*Mettre la voie A et B sur DC et régler la sensibilité avec le commutateur, au besoin bien repositionner le signal avec la molette.



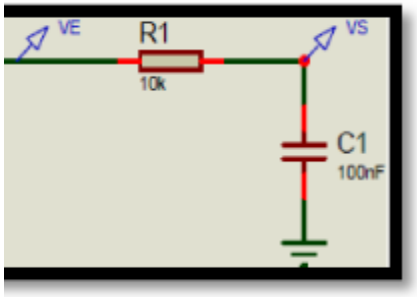
Avoir une bonne amplitude, au besoin bien repositionner le signal avec la molette.

III.1:h. simulation par graphe (Chronogrammes) :

III.1:h.1- Mise en place des points de mesure:

- Placer des points de mesure (intensité ou tension) sur les conducteurs du schéma à  l'aide des icones représentant des sondes.

Dans notre exemple, on souhaite visualiser le signal VE et le signal VS on placera donc 2 sondes de tensions, un nom pour chaque sonde est donné par défaut.



III.1:h.2- Création de la fenêtre graphique

- Sélectionner  puis le type de graphe désiré, ici : ANALOGUE (analogique).

Créer une fenêtre graphique en allant sur l'espace travail et tout en cliquant sur le bouton gauche, faire glisser la souris pour dérouler la fenêtre du graphe

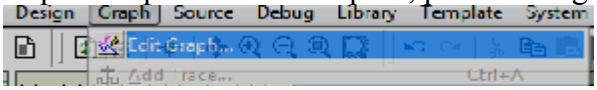
III.1:h.3- Indications des courbes à tracer:

Il suffit d'abord de cliquer sur le schéma la sonde à sélectionner, puis par un simple glissé déposé la mettre dans la fenêtre graphique du côté gauche (axe des abscisses).

III.1:h.4- Réglage du temps de simulation:

● On accède au menu d'édition du graphe soit directement en double cliquant sur le graphe.

● Soit en passant par le menu Graphes, puis Editer graphe :



- Régler le temps de départ (en général 0) et le temps de fin pour avoir 2 ou 3 périodes. (ici 10ms).

III.2: Langage et compilateur mikroC pour PIC[5]:

Le langage mikroC pour PIC a trouvé une large application pour le développement de systèmes embarqués sur la base de microcontrôleur.

Il assure une combinaison de l'environnement de programmation avancée *IDE* (Integrated Development Environment) , et d'un vaste ensemble de bibliothèques pour le matériel, de la documentation complète et d'un grand nombre des exemples.

Le compilateur *mikroC* pour PIC bénéficie d'une prise en main très intuitive et d'une ergonomie sans faille. Ses très nombreux outils intégrés (mode simulateur, terminal de communication Ethernet, terminal de communication USB, gestionnaire pour afficheurs 7 segments, analyseur statistique, correcteur d'erreur, explorateur de code, mode Débug ICD...) associé à sa capacité à pouvoir gérer la plupart des périphériques rencontrés dans l'industrie (Bus I2C™, 1Wire™, SPI™, RS485, Bus CAN™, USB, gestion de cartes compact Flash et SD™/MMC™, génération de signaux PWM, afficheurs LCD alphanumériques et graphiques, afficheurs LEDs à 7 segments, etc...) en font un outil de développement incontournable pour les systèmes embarqués, sans aucun compromis entre la performance et la facilité de débogage.

III.2:a. Compilateur mikroC PRO pour PIC[5]:

La 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...

1. Dans la suite nous utiliserons le compilateur mikroC PRO v.1.65
2. La simulation des applications de programmation nous réalisons à l'aide du logiciel PROTEUS v.7.6 SP

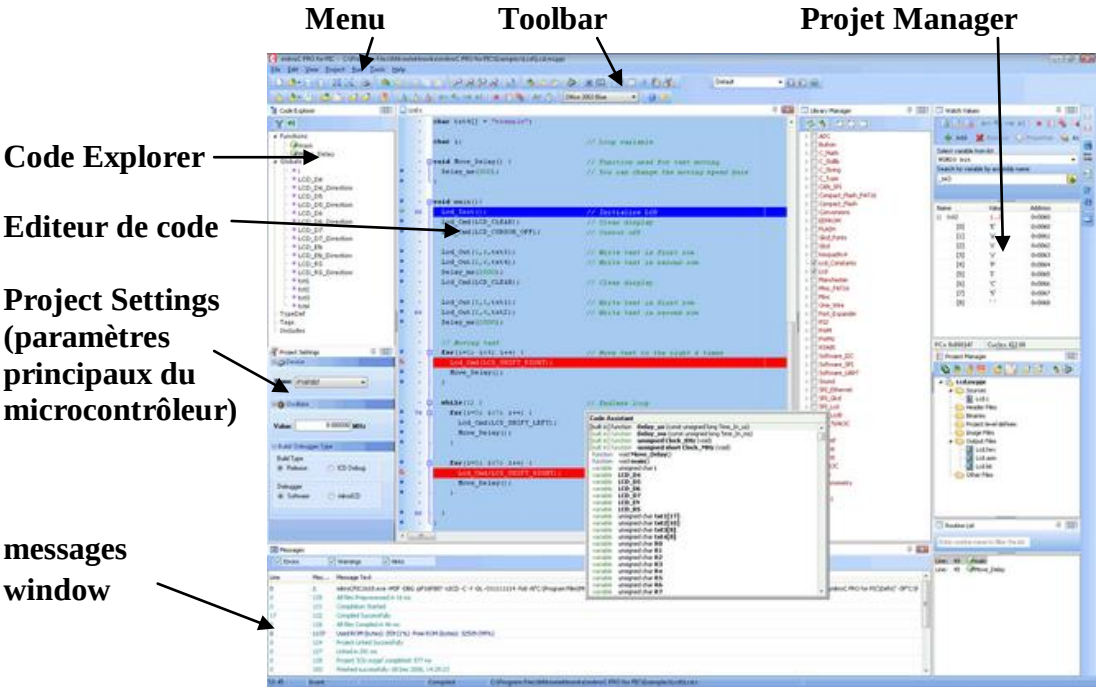


Figure III.1:L'environnement IDE du compilateur microC PRO

Editeur de code (voire la figure III.1 : Code Editeur)
L'éditeur de code est le même que toute éditeur de texte standard pour l'environnement de Windows, y compris Copie, Coller, Annuler les actions etc… Il possède en plus des ressources comme suit :

- Coloration syntaxique réglable
- Assistant de code
- Assistant de paramètre
- Mise en forme automatique

Dans la boîte de dialogue *Options* (figure III.2) vous pouvez configurer la coloration syntaxique, l'aide pour le code et paramètres, la correction automatique etc. Pour accéder à ce dialogue cliquez sur Tools > Options du menu déroulant ou sur l'icône
Assistant de code
Si vous imprimez les premières lettres du mot et puis appuyez sur Ctrl + Espace, tous les identificateurs autorisés correspondant aux caractères imprimés seront offerts dans une fenêtre (voir la figure III.3). Maintenant, nous pouvons continuer à réduire le choix de taper ou d'utiliser la sourie pour sélectionner l'option appropriée dans la proposée et appuyez sur Entrée.

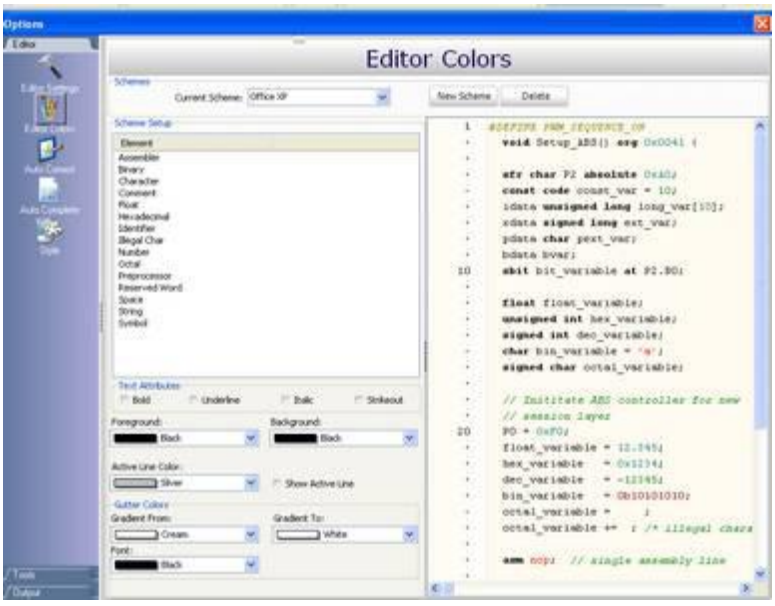


Figure III.2: Boîte de dialogue « Options »

Figure III.3: Assistant de code

Mise en forme automatique

Afin d'obtenir une lisibilité maximale, il est possible de générer automatiquement la mise en forme de certaines instructions. Par exemple, tapez l'instruction IF, puis CRTL + J. A ce stade, l'éditeur ajoute tout seul les instructions. A vous ensuite de compléter le programme.

Affichage syntaxique coloré

Toujours dans le but d'obtenir une lisibilité maximale de votre programme, il vous est possible de configurer entièrement les couleurs de chaque partie du listing (figure III.2). Par exemple les commentaires en "vert", les instructions en "noir", les valeurs numériques en "bleu clair", etc.

Outils intégrés

Le compilateur "MikroC PRO" pour PIC intègre différents petits outils très pratiques qui vous simplifieront l'écriture des programmes de vos applications.

III.2:b. Création d'un nouveau projet[5]:

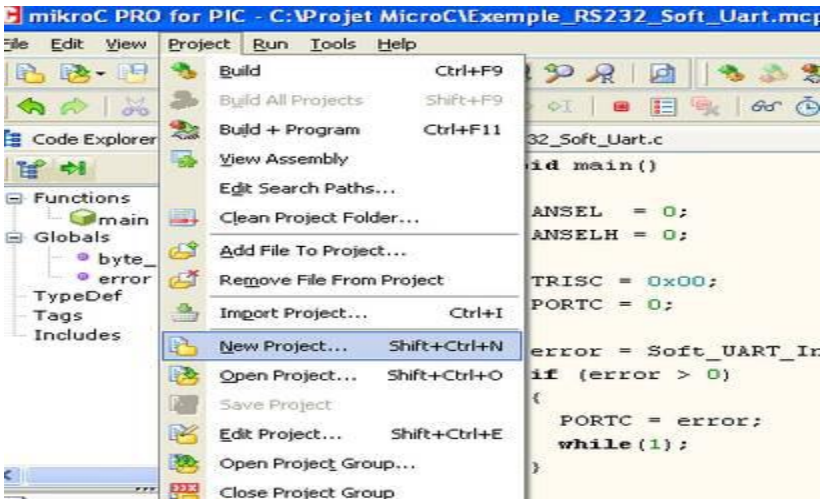
Le mikroC PRO pour PIC organise des applications dans des projets, composé d'un seul fichier de projet (extension. mcppi) et un ou plusieurs fichiers sources (extension).

Les fichiers source peuvent être compilés que si elles font partie d'un projet.

Le fichier projet contient les informations suivantes :

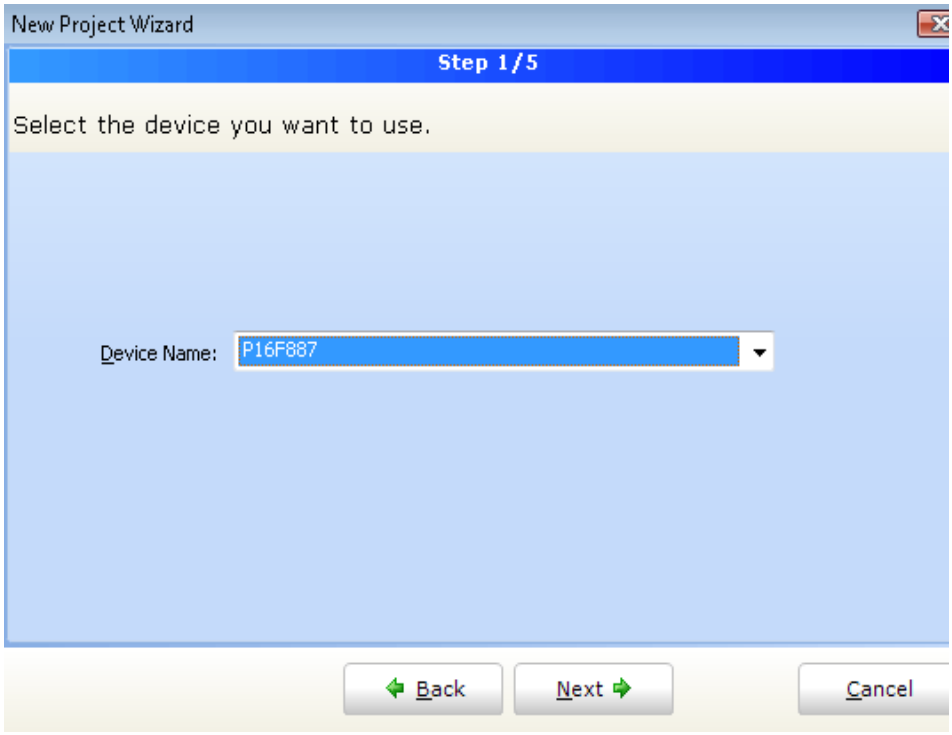
- Nom du projet et une description facultative
- Composant cible
- Option du composant
- Fréquence d'horloge du composant
- La liste des fichiers source du projet avec les chemins
- Fichiers d'image
- Fichiers binaires (* mcl.)
- D'autres fichiers

La meilleure façon de créer un projet c'est à l'aide de l'Assistant Nouveau projet (menu Project> New Project) ou en cliquant sur l'icône Nouveau projet  à partir de la barre d'outils du projet.

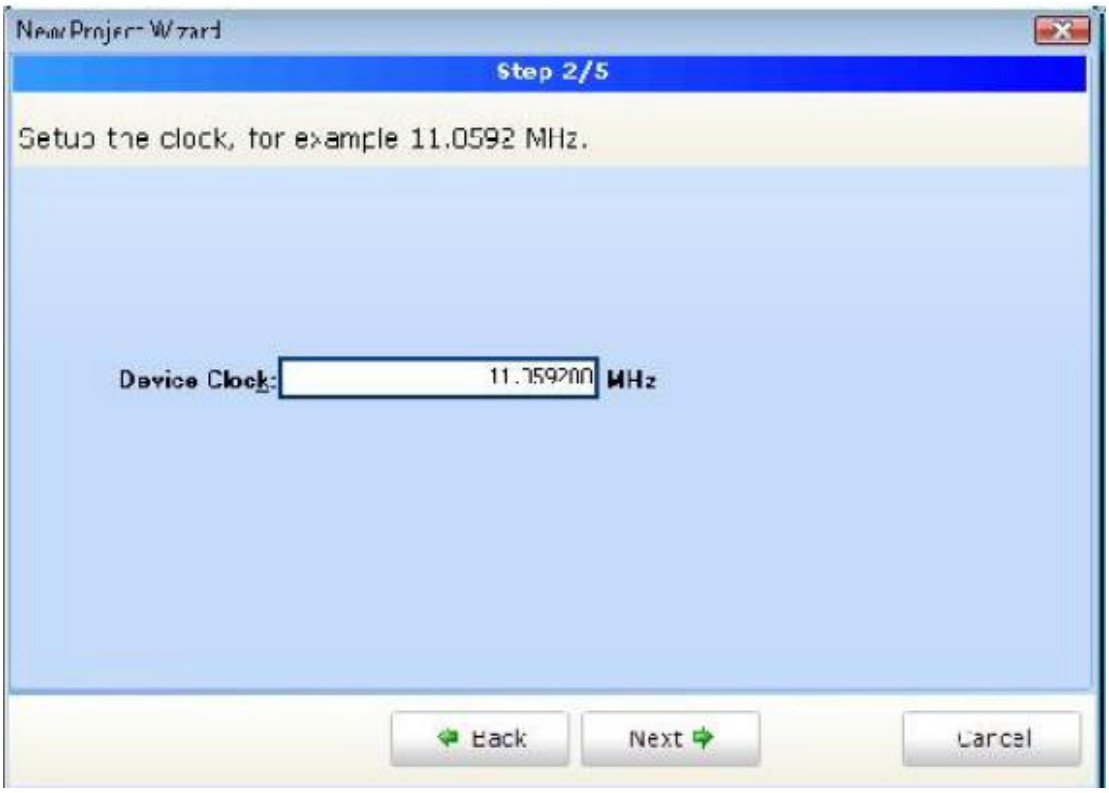


Nouvelles étapes de l'Assistant de projet .
Commencez à créer votre nouveau projet, en cliquant sur le bouton **Next** :

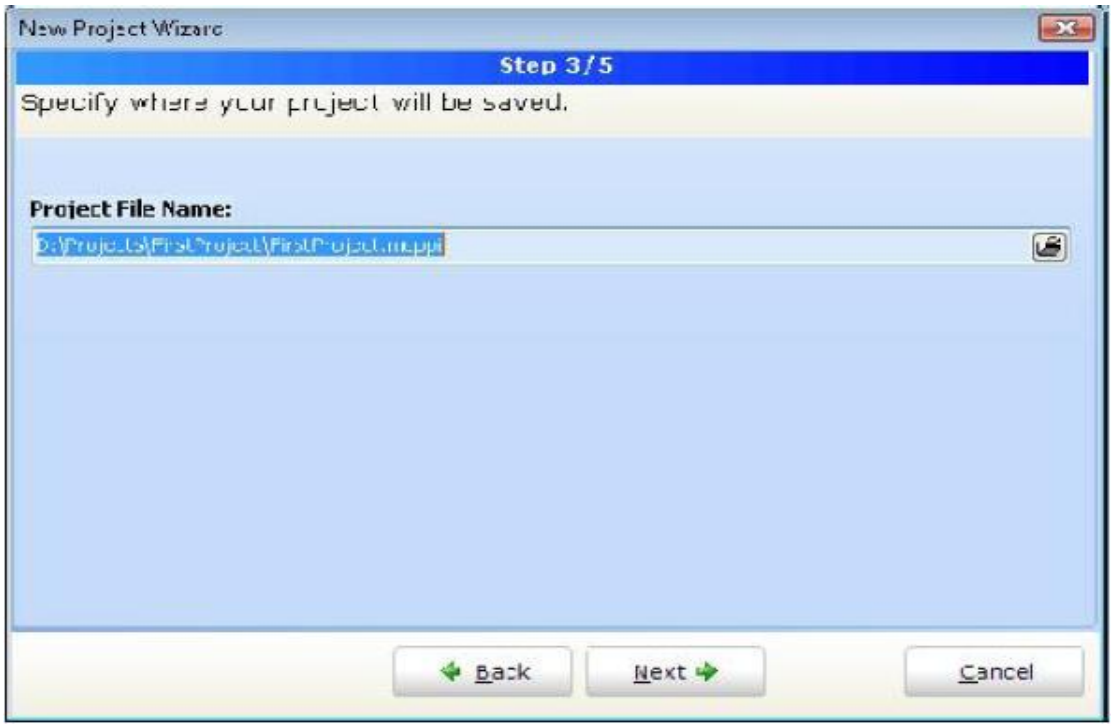
Premier pas - Sélectionnez le périphérique dans le périphérique dans la liste déroulante.



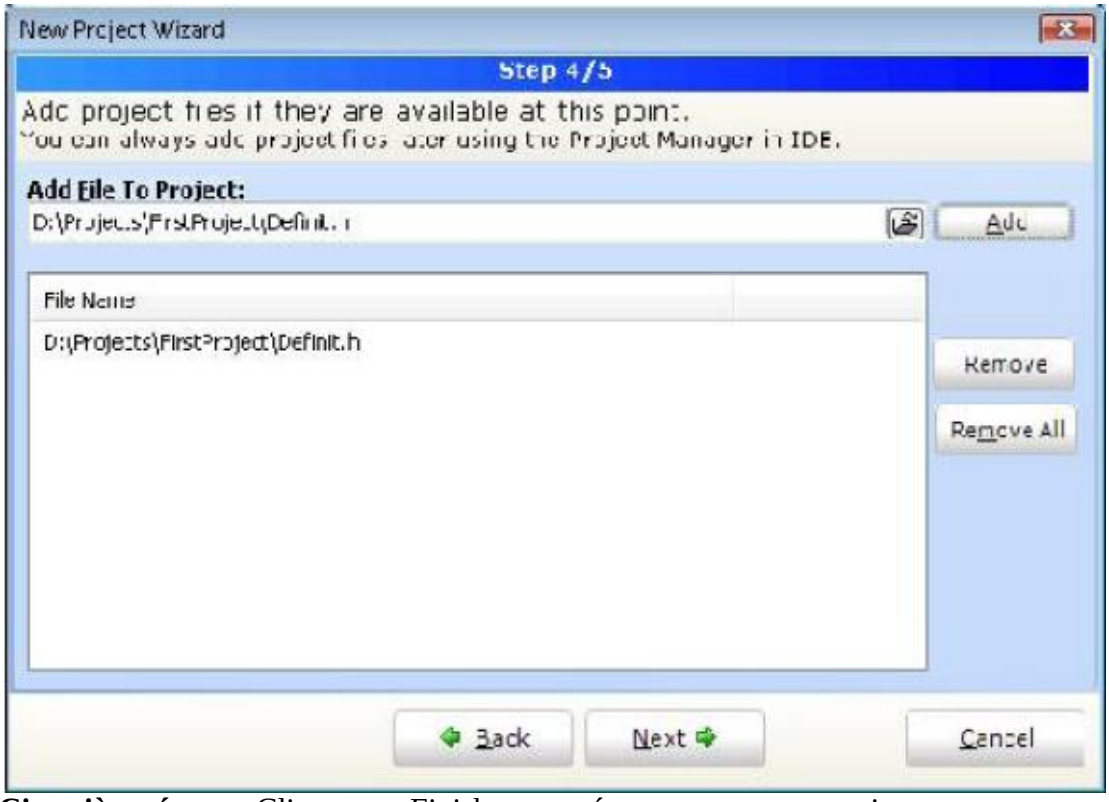
Deuxième pas - Saisir la valeur de fréquence de l'oscillateur



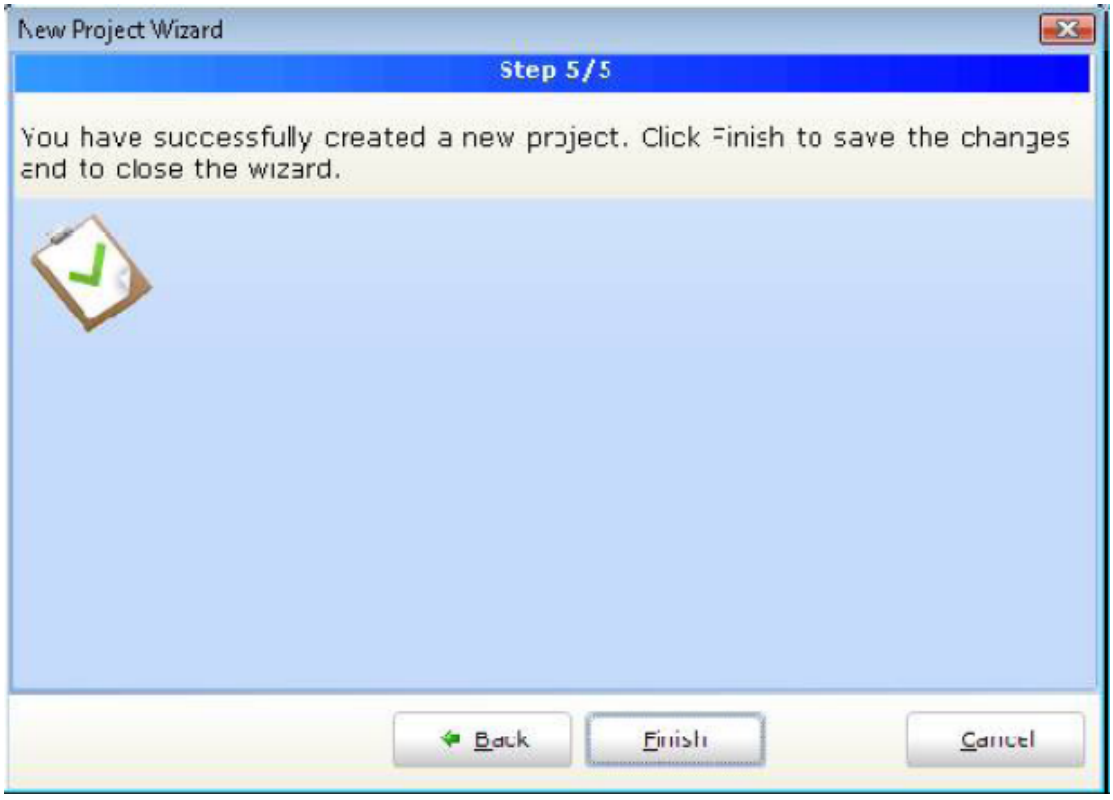
Troisième pas - Spécifiez l'emplacement où votre projet sera enregistré.



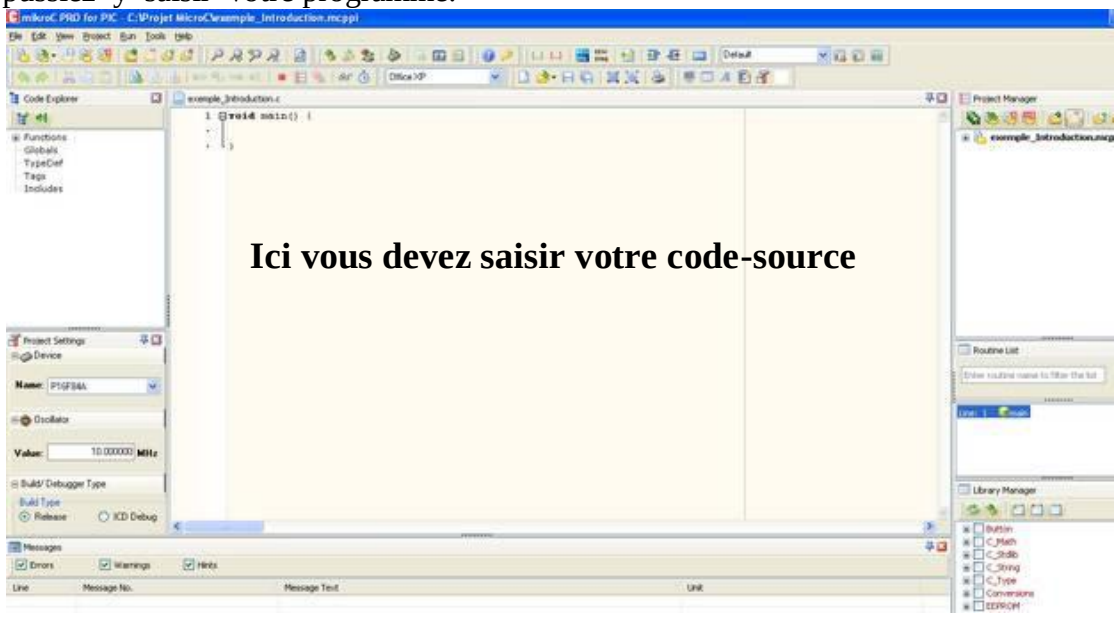
Quatrième pas - Ajout pour le projet un fichier s'ils sont disponibles en ce moment. Vous pouvez toujours ajouter des fichiers de projet plus tard
en utilisant Project Manager.



Cinquième étape - Cliquez sur Finish pour créer votre nouveau projet.

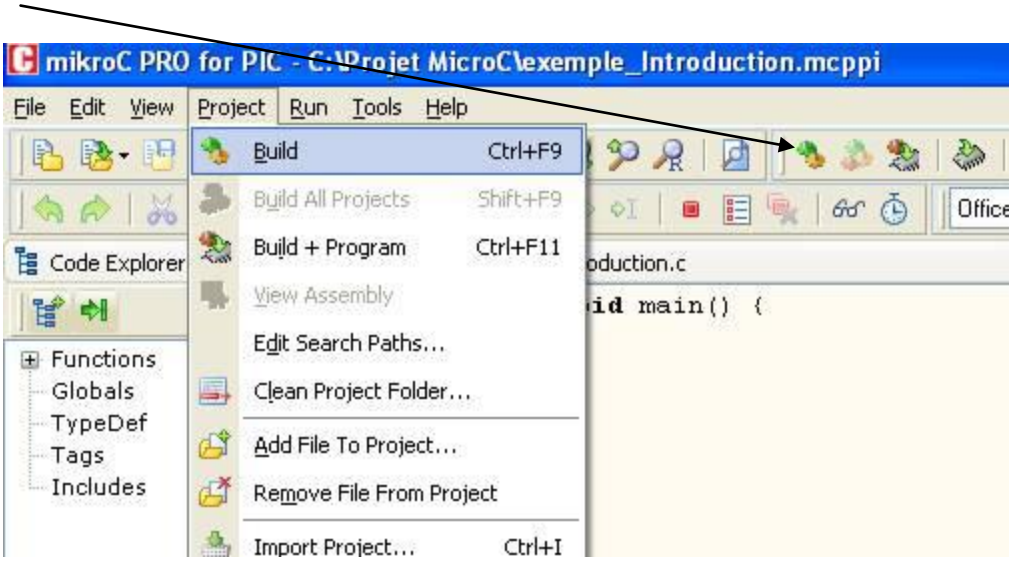


A ce stade, une nouvelle fenêtre vide (fig. III.4) s'affiche afin que vous pussiez y saisir votre programme.



III.2:c. Compilation[5]:

Lorsque vous avez créé le projet et écrit le code source, il est temps de le compiler. Sélectionnez **Project** → **Build** à partir du menu déroulant ou cliquez sur l'icône **Build** dans la barre d'outils du projet.



Si plus d'un projet est ouvert, vous pouvez compiler tous ouverts projets en sélectionnant Project > Build All dans le menu déroulant, ou cliquez sur l'icône  de la barre d'outils du projet.

Barre de progression s'affiche pour vous informer sur l'état de la compilation. Si il y a des quelques erreurs, vous en serez informé dans la fenêtre d'erreur (fig. III.4).

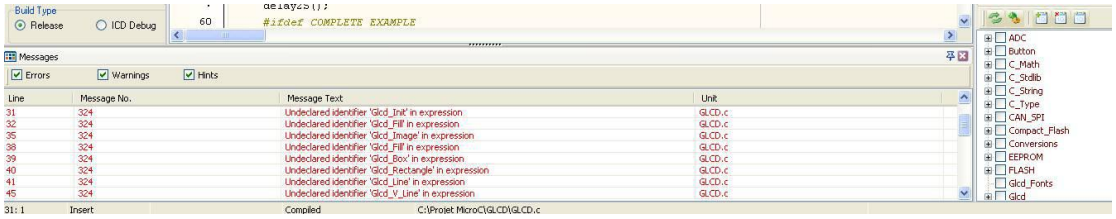


Figure III.4:Avertissement des erreurs

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

Tab .III .1: compilation (mikroC)

III.2:d. Fichiers Sources[5]:

Création d'un nouveau fichier source

Pour créer un nouveau fichier source, procédez comme suit :
Sélectionne New depuis le menu File ou pressez Ctrl+N ou cliquez sur l'icône (New File) depuis la barre d'outils File.
Une nouvelle fenêtre s'ouvrira dans laquelle vous pourrez saisir votre code source. Sélectionnez alors Save depuis le menu

File ou pressez Ctrl+S ou cliquez sur l'icône (Save File) depuis la barre d'outils (File) et nommez ce dernier comme vous le voulez.

Ouvrir un fichier existant

Sélectionnez Open depuis le menu File ou pressez Ctrl+Q ou cliquez sur l'icône (Open File) depuis la barre d'outils(File). Dans la boite de dialogue Ouvrir, sélectionnez alors l'emplacement du fichier que vous désirez ouvrir et cliquez sur le bouton Ouvrir. Le fichier sélectionné s'affiche dans sa propre fenêtre. Si le fichier sélectionné est déjà ouvert, sa fenêtre d'édition (Editor) deviendra alors active.

Sauvegarder un fichier

Assurez-vous que la fenêtre contenant le fichier que vous voulez sauvegarder est active.
Sélectionnez ensuite Save depuis le menu File ou pressez Ctrl+S ou cliquez sur l'icône Save Fail de la barre d'outils File.

Sauvegarder un fichier sous un nom différent

Assurez-vous que la fenêtre contenant le fichier que vous voulez sauvegarder est active. Sélectionnez ensuite Save As depuis le menu File. Une boîte de dialogue 'Save [nom du fichier] .cas' s'affichera alors. Dans cette boîte, sélectionnez l'emplacement dans lequel vous voulez sauvegarder le fichier. Dans le champ *Nom du fichier*, modifiez le nom du fichier.

Règles générale d'écriture en microC

- Les instructions propres au langage microC doivent être écrites en minuscule (**void** main (void)).
- Les instructions particulières aux microcontrôleurs doivent être écrites en majuscule (TRISB).
- Les retours à la ligne et les espaces servent uniquement à aérer le code
- Toutes instructions ou actions se terminent par un point virgule « ; ».

Début et fin d'un programme

En langage microC, un programme commence avec les mots-clés:
void main()
Après cela, une accolade ouvrante est utilisée pour indiquer le début du corps de programme .Le programme se termine par une accolade fermante.
le programme a la structure suivante :

```
void main()
{
 //Votre code ici
}
```

Fin d'une instruction

Le point virgule « ; » indique la fin d'une instruction, sinon une erreur du compilateur sera générée.

```
j = 5; // correcte
j = 5  // erreur
```

Espaces blancs

Les espaces blancs sont des espaces, des flans, les tabulations et les caractères de nouvelle
ligne. Le compilateur *microC* ne tient pas compte tous les espaces blancs. Ainsi, les trois
séquences suivantes sont identiques :

```
int i; char j;
ou
int i;
char j;
ou
int i;
   char j;
```

Instructions d'itération for, while, do, goto, continue et break

Les instructions d'itération nous permettent d'effectuer des boucles dans un programme, où une partie d'un code doit être répété un certain nombre de fois. Dans mikroC l'itération peut être
effectuée de quatre façons. Nous se penchera sur chacun des exemples :

- Utilisation de for
- Utilisation de while
- Utilisation de do
- Utilisation de goto, continue et break

ADC	Utilis**é** pour conversion analogique/numérique
PWM	Utilisé pour les opérations avec le module de PWM
LCD_CLEAR	Effacer l'affichage
LCD	Utilisé pour le fonctionnement avec LCD standard
Delay_ms	Crée le retard constant dans les unités de millisecondes

Tab. III.2 : Instructions d'itération

Conclusion

Après identification du logiciel proteus (ISIS) et logiciel Mikroc.

Nous considérons l'un des meilleurs logiciels pour programmation et simulation des microcontrôleurs .

Par conséquent, nous allons utiliser ces deux logiciels pour réaliser nos projets d'étude et programmer le PIC16F84 .

Chapitre IV

Simulation Et Réalisation

Introduction

Après que nous avons vu les caractéristiques du microcontrôleur pic16F84. Alors nous savions comment utiliser les programmes (Proteus) et MikroC , Dans ce chapitre, Nous allons simuler et réaliser deux projets consécutifs en utilisant le logiciel de simulations (Proteus) ,le logiciel de programmation (MikroC) et enfin les plaques d'essais pour la réalisation .
Il ya plusieurs étapes pour bien parachever nos deux projets :

IV.1: Première projet :contrôle des feux de circulation

IV.1:a Première étape : conception du schéma électrique avec le logiciel de simulations (Proteus) ISIS

Nous avons besoin de:

- Un microcontrôleur Pic16F84.
- 12 résistances (220Ω).
- 2 condensateurs (c=15pf).
- oscillateur à quartz XTAL : F= 4MHz.
- 12 LEDs (3 rouge, 3 vert, 3 jaune).
- Source de tension DC (5v et 12v).

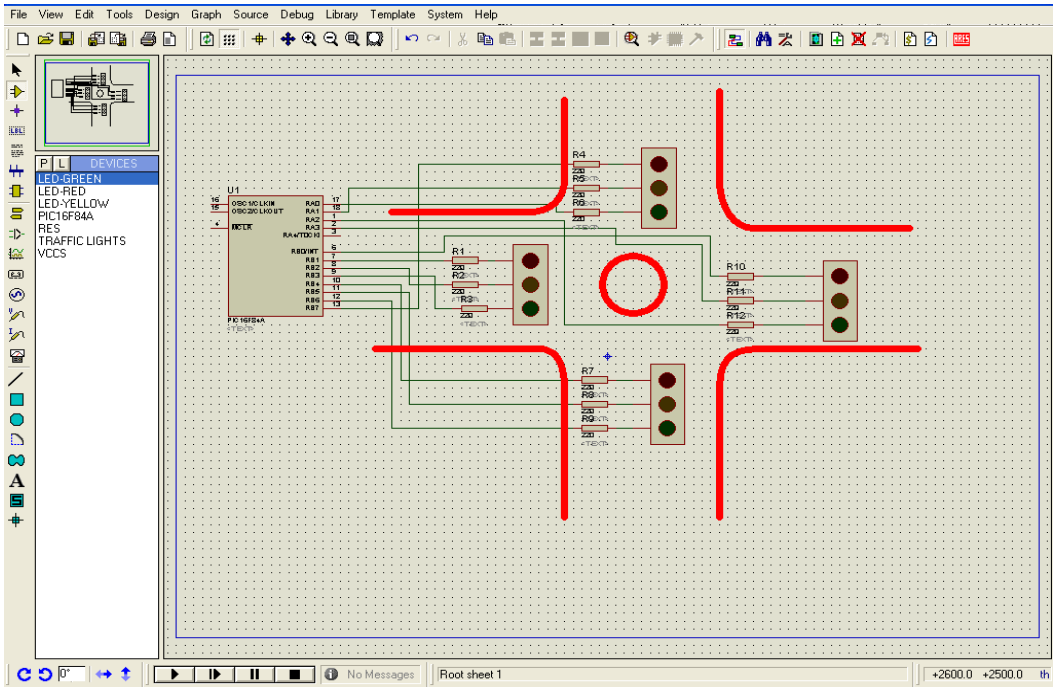


Figure IV.1: Plan du projet par le logiciel de simulations (Proteus) isis

IV.1:b Deuxième étape : Ecrire un programme à partir du logiciel de programmation (MikroC) qui correspond à laconception électrique

Dans ce projet. Nous avons écrit un programme qui permet au microcontrôleur PIC16F84 à bien commander et contrôler notre système électrique.
Nous avons utilisé la language mikroC pour écrire ce programme parce que nous considérons qu'il est beaucoup plus facile que la langue assembleur et mikroBasic.
Apres avoir terminé le programme et s'assuré qu'il n'y ait pas d'erreurs. Nous cliquons sur le (build projet). Le MikroC va compiler automatiquement le programme écrit en en langage MikroC en programme langage machine (Numérique) qui s'appelle le fichier hexadécimal avec l'extension HEX(FILE.HEX).

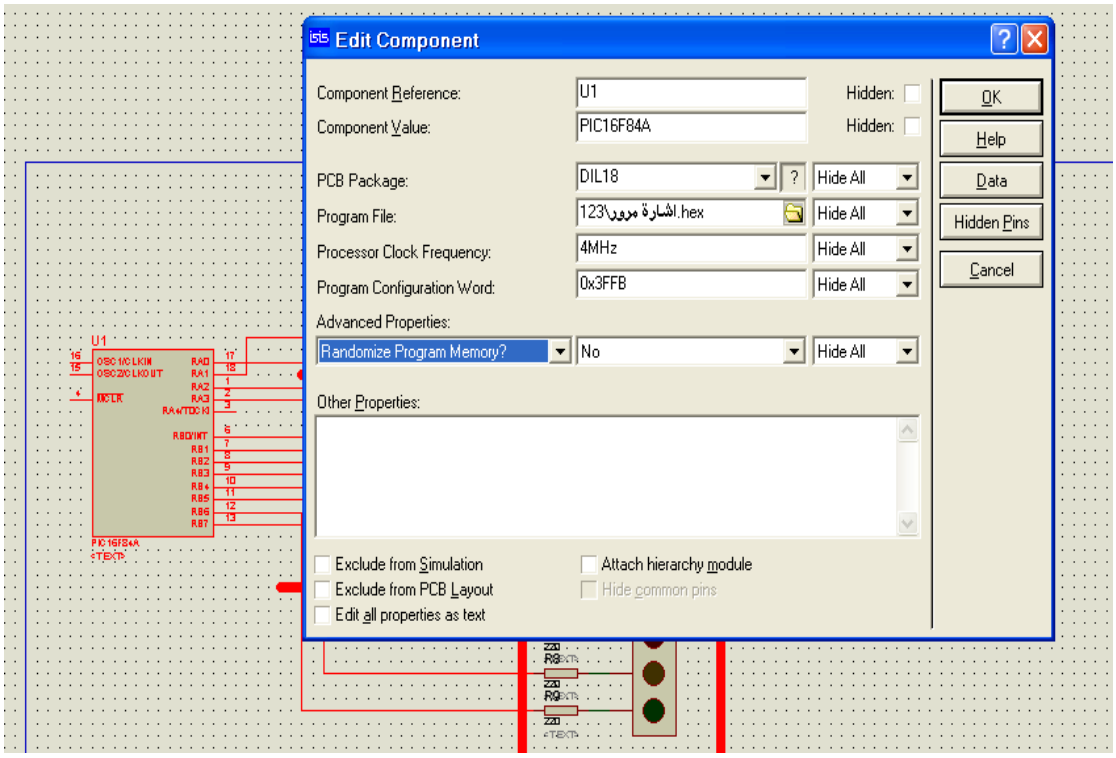
```
void main() {
TRISA=0X00;
TRISB=0X00;
for(;;)
{
PORTB=0b01000011;
PORTA=0b00000001; delay_ms(7000);
PORTB=0b00100100;
PORTA=0b00001010;delay_ms(7000);
PORTB=0b10011000;
PORTA=0b00000100;delay_ms(7000);
PORTB=0b00100100;
PORTA=0b00001010;delay_ms(7000);
  }
  }
```

- **Language mikroC**

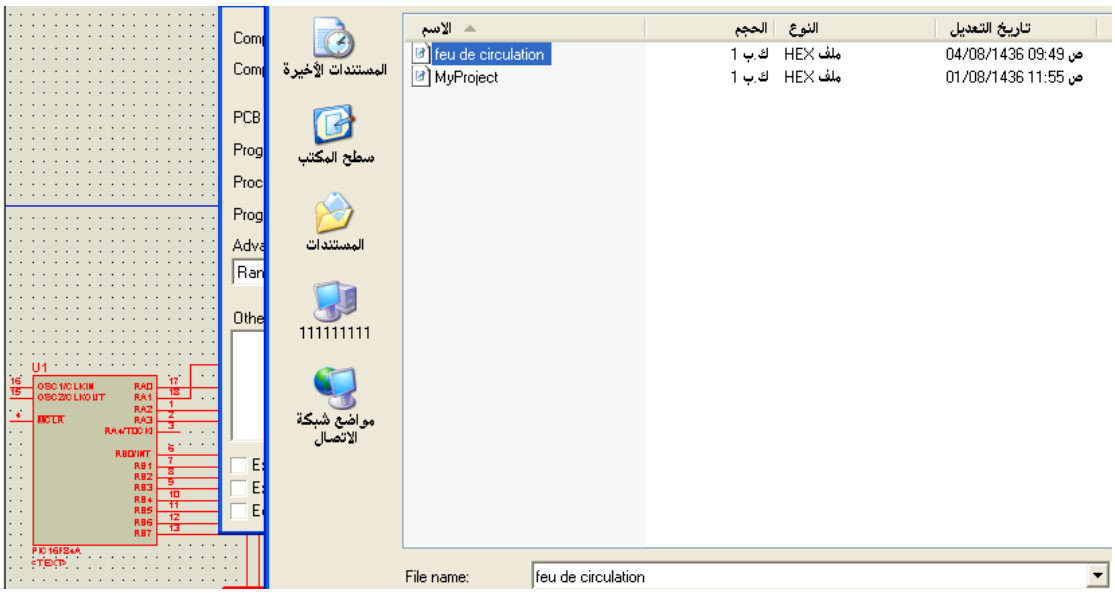
```
        C00060083120D088A000C0882000800 1C0:
A110A128000840A8C0A03198D0AAD1000120003208:
              C003031D09280800BA08002200:
A008316850186014330831286000130850 0DC10002:
A002430CB008330CC00CF30CD00CD0B23282910003:
A00CC0B2328CB0B2328243086000A308500CA10004:
A002430CB008330CC00CF30CD00CD0B3328F910005:
A00CC0B3328CB0B33289830860004308500 1C10006:
A002430CB008330CC00CF30CD00CD0B4328C910007:
A00CC0B4328CB0B4328243086000A3085004A10008:
A002430CB008330CC00CF30CD00CD0B53289910009:
        C00AA00CC0B5328CB0B532818285A28E50:
                          E00FA3F7702400:
                             FF00000001:
```

- **Language machine (HEX)**

- Aller à la logiciel Isis et de la conception électrique.
- Double-cliquez sur PIC16F84.



- Entrez sur (program file) et apportez le fichier (file.HEX) et appuyez sur ok.



- Le programme est chargé dans le PIC16F84.
- Assurez-vous que tous les fils sont connectés.

- Cliquez sur le bouton (play) 

IV.1:c La troisième étape : Réalisation

IV.1:c.1-Tester le projet sur la plaque d'essai

- **Préparation des composantes du projet**

Nous avons acheté tous les composants électroniques

- traitement du signal et microcontrôleur dans Le hall technologique. Un microcontrôleur PIC16F84.

- 12 résistances (220Ω).

- 2 condensateurs (c=15pf).

- oscillateurs à quartz XTAL : F= 4MHz .

- 12 LEDs (3 rouge,3 vert , 3 jaune).

- Source de tension DC (5v et 12v).

IV.1:c.2- Graver le programme sur le pic :

Nous avons chargé le programme dans le microcontrôleur pic16F84 par le programmateur UTProgPicv1.3 à l'aide de l'adaptateur avec l'ordinateur de transfert du file.HEX le logiciel appelé WIN PIC800. Tout cela nous l'avons fait au l'laboratoire

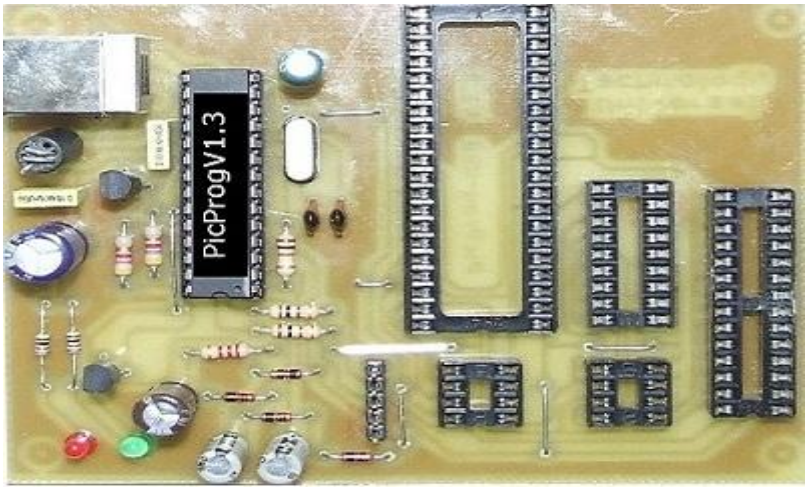


Figure IV.2: programmateur de pic

Présentation

Le UTProgPicv1.3 est utilisé pour programmer les microcontrôleurs de la marque « Microchip » de type « Pic » des familles 12F, 16F et 18F ayant 8, 18, 28 et 40 pins (plus de 200 Pic !) et les mémoires 24Cxx.

Caractéristiques

1. Connexion USB.
2. Programmateur auto-alimenté.
3. Capable de programmé environs 200 microcontrôleurs de types Pic de Microchip des familles : 12F, 16F, 18F (et les versions C, LF...) ayant 8, 18, 28 et 40pins.
4. Supporte la programmation des mémoires de type 24Cxx.
5. Dimensions trés pratique : 9,5x7,5cm.
6. Comporte des Leds de signalisation : Verte(Connécté) et Rouge(Programmation)

IV.1:c.3-Raccordement des éléments de circuit:

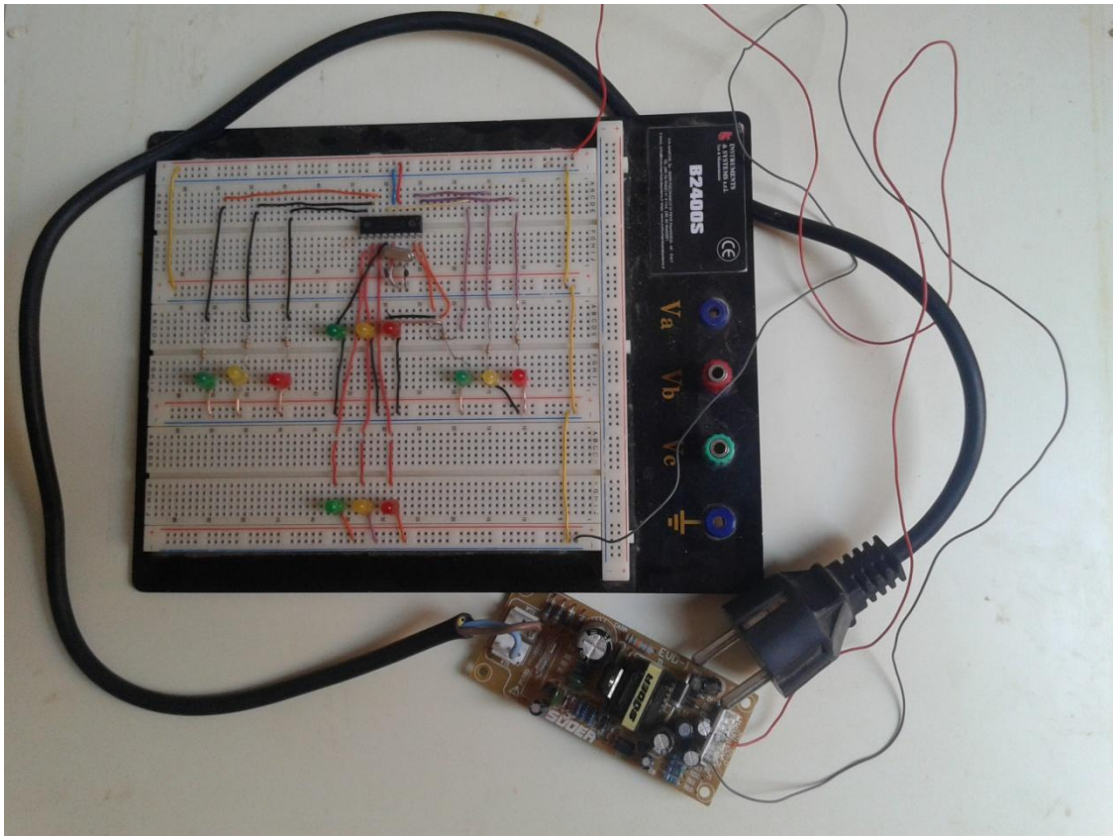


Figure IV.3: Installation du circuit électrique sur la carte d'essai

IV.2: Deuxième projet :Contrôle du sens de rotation d'un moteur CC

Dans ce projet nous avons besoin de

- Un microcontrôleur Pic16F84.
- Un afficheur type LM020L.
- Petit moteur du type de courant continu(6VDC).
- 4 résistances (1kΩ).
- 2 Bouton.
- 2 transistors type BC548.
- Source de tension DC (5v et 12v).
- 2 Relais (6VDC) .

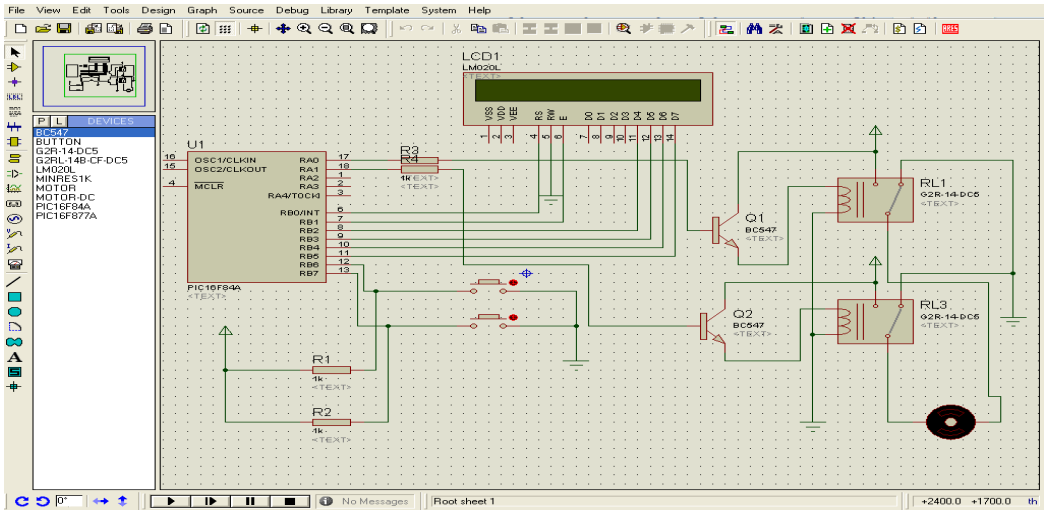


Figure IV.4: Plan du projet par le programme isis

En suivant les mêmes étapes dans le projet précédent et en utilisant le programme suivant :

```
// Lcd pinout settings
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;

// Pin direction
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;
void main() {
TRISA=0x00;
PORTA=0;
TRISB.B6=1;
TRISB.B7=1;
Lcd_Init();
Lcd_Cmd(_LCD_CLEAR);
Lcd_Cmd(_LCD_CURSOR_OFF);
for(;;)
{
while(PORTB.B6==0)
{
PORTA=0b00000001;
Lcd_Out(1,1,"DROITE ");
}
while(PORTB.B7==0)
{
PORTA=0b00000010;
Lcd_Out(1,1,"GAUCHE ");
}
Lcd_Out(1,1,"STOP  ");

PORTA=0b00000100;
```

IV.2:a. Réalisation :

IV.2:a.1-Tester le projet sur la plaque essai :

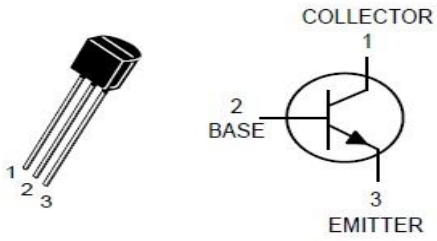
- Un microcontrôleur Pic16f84.

- Petit moteur du type de courant continu (6VDC).

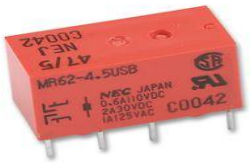
- 4résistances (1kΩ).

- 2 Switch on off.

- 2 transistors type BC548.



- 2 Relais (6VDC) .



- Source de tension DC (5v et 12v).

IV.2:a.2- Raccordement des éléments de circuit :

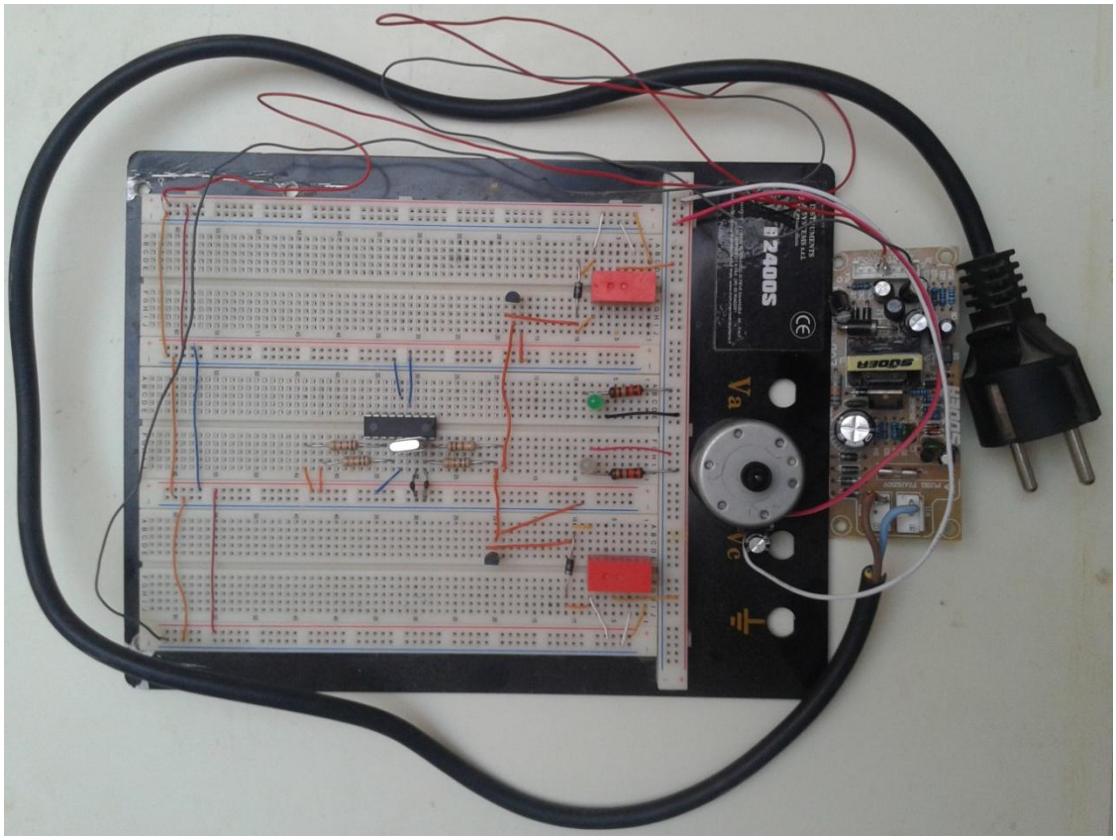


Figure IV.5: Installation du circuit électrique sur la carte d'essai

Conclusion:

Après avoir terminé avec succès nos projets Nous avons pu avoir une bonne idée et une initiation claire sur le contrôle des systèmes électriques en utilisant des microcontrôleurs .Cette étude offre les éléments essentiels de la commande avancée utilisé par la technologie de pointe indispensable dans les différents domaines

Conclusion Générale

Notre mémoire a bien montré les technique essentielles qu'il faut avoir pour maitriser la commande par les microcontrôleurs.

Nous avons donné des exemples simples et claires qui aident finalement le lecteur a bien s'initier avec la commande par les microcontrôleurs .

D'autre part , nous avons montré qu' il faut avoir chez le futur chercheur des acquits de devers disciplines (électrique , électronique, informatique, ...) pour conquérir le domaine de la commande par les microcontrôleurs .

Par ailleurs, n'oublions pas que le jeune chercheur doit jouir du savoir faire pratique pour pouvoir appliquer la théorie sur des équipements électriques réels : c'est le grand défit pour les chercheurs .

En définitive, la perspective de notre travail qui reste c'est faire synchroniser fonctionnant parallèlement deux microcontrôleurs pour commander un processus industriel quelconque.

Bibliographies

- [1]Organisation fonctionnelle d'un système à microcontrôleur **(G BERTHOME – Lycée Mireille GRENET – COMPIEGNE)**
- [2]Mémoire **" réalisation d'un hacheur a faible puissance contrôlé par un microcontrôleur Le pic16F877A"**
- [3]PIC16F84 " **Philippe Hoppenot Professeur dans université d'evry "**
- [4]Microchip : "PIC16F8X – 18-pin Flash/EEPROM 8-bit micro-controllers– DS30430C,
- http://www.microchip.com/1010/suppdoc/
- [5]Programmation en mikroC. Application pour les microcontrôleurs de la famille PIC