République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ahmed Daria Adrar

Faculté des Sciences et de la Technologie

Département sciences de la technologie

Mémoire de fin d'étude en vue de l'obtention du diplôme de Master en :

Filière : Électrotechnique

Spécialité : Commande Électrique

Thème :

Système d'irrigation intelligent
(Commande par microcontrôleur)

Préparé par :

M. MESSAUDI LAHCEN.

M. GHOUZAL BACHIR.

Membres de jury d'évaluation :

M. OULED ALI Omar	**Examinateur**	**MAA.**	**Univ. Adrar**
M. LARIBI Slimane	**Examinateur**	**MCA**	**Univ. Adrar**
M.MANSOURI Smail	**Encadreur**	**MCA**	**Univ. Adrar**

Année Universitaire : 2020/2021

الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne populaire et démocratique
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSITE AHMED DRAYA - ADRAR
BIBLIOTHÈQUE CENTRALE
Service de recherche bibliographique
N°.....B.C/S.R.B//U.A/2021

جامعة احمد دراية - ادرار
المكتبـــة المركـــزية
مصلحة البحث الببليوغرافي
الرقم.....م .م/م.ب.ب /ج.أ/2021

شهادة الترخيص بالإيداع

انا الاستاذ(ة): منصوري سماعيل

المشرف مذكرة الماستر.

الموسومة بـــ: **Système d'irrigation intelligent**
(Commande par microcontrôleur)

من انجاز الطالب(ة): مسعودي لحسن

و الطالب (ة): غوزال بشير

الكلية: العلوم والتكنولوجيا

القسم: علوم التكنولوجيا

التخصص: تحكم كهربائي

تاريخ تقييم / مناقشة: 22 جوان 2021

أشهد ان الطلبة قد قاموا بالتعديلات والتصحيحات المطلوبة من طرف لجنة التقييم / المناقشة، وان المطابقة بين النسخة الورقية وا ل لكترونية اس توفت جميع شروطها.

و با مكانهم ايداع النسخ الورقية (02) والاليكترونية (PDF)

- امضاء المشرف:

ادرار في2021/07/07

مساعد رئيس القسم:

د. عرباوي إلياس
رئيس قسم علوم التكنولوجيا بكلية
العلـــوم والتكنولوجيـــا

ملاحظة :لاتقبل أي شهادة بدون التوقيع والمصادقة.

Dédicace

“

L ouange à Dieu, par la grâce duquel les bonnes actions sont accomplies, et bénédictions et paix sur notre maître Muhammad et toute sa famille et ses compagnons. Je dédie cette thèse à :

À mon idéal éternel, celui qui s'est toujours sacrifié pour me voir réussir, à qui je porte fièrement son nom à mon cher père, que Dieu prolonge sa vie,

À qui le jour n'est pas agréable seulement de la voir, et les jours ne sont heureux que par sa présence, la flamme de mon cœur, ma vie et mon bonheur, ma mère bien-aimée, que Dieu prolonge sa vie,

À mon frère, mes sœurs, mes sœurs, mes neveux, et toute ma famille et mes proches qui ont été à mes côtés toute ma vie.

À mes amis, mes compagnons de voyage et ma source de bonheur qui ont été à mes côtés durant mes années scolaires et avec qui j'ai partagé toutes sortes d'épreuves et de peurs, je vous remercie pour tous les moments que j'ai passés avec vous.

Merci.

”

- LAHCEN

Dédicace

“

Tout d'abord, nous remercions Dieu, qui nous a permis d'être bien pour faire ce travail du début à la fin .Je dédie cette thèse à l'esprit pur de « ma mère » et je prie Dieu de la couvrir de sa miséricorde,

À mon soutien moral et source de joie et de bonheur, celui qui s'est toujours sacrifié pour me voir réussir, à toi PAPA »,

À mes frères, mes sœurs, mes nièces et neveux, à toute ma famille, à tous mes amis et parents qui ont été à mes côtés pendant ces années d'études, et à tous ceux qui m'ont appris la lettre de mes professeurs et professeurs,

À u compagnon du chemin et celui qui a partagé l'effort avec moi « LAHCEN »

Merci.

”

-BACHIR

Remerciements

“

Tout d'abord, je remercie Dieu Tout-Puissant de nous avoir donné le courage et la patience de faire ce travail. Je tiens à Remercier tout particulièrement notre superviseur, ***Dr.Mansouri Smail****, pour son aide compétente, ainsi que pour sa patience et ses encouragements,*

Son regard critique nous a été précieux pour organiser le travail et améliorer la qualité des différents services,

Je tiens également à remercier les moniteurs de contrôle électrique ainsi que tous les conseils et informations qu'ils nous ont prodigués avec une patience et un professionnalisme sans égal,

Merci beaucoup et appréciation Au ***Dr.Chabani Boumediene*** *et au* ***M. Dahabi Hassan*** *pour leur aide et leurs encouragements inestimables. Enfin, je tiens à remercier tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail,*

”

Résumé

L'agriculture est l'une des activités les plus anciennes et les plus importantes que l'homme ait pratiquées et continue de pratiquer en raison de la grande importance qu'elle revêt pour lui pour assurer sa nourriture.

Mais obtenir de la qualité, gagner du temps et réduire l'effort sur lui-même l'ont obligé à se lancer dans le monde de la numérisation et à utiliser des techniques qui y contribuent, car nous pouvons adopter de nombreuses technologies, comme les microcontrôleurs.

Dans notre projet, nous avons utilisé (PIC16F84A) pour contrôler la pompe d'irrigation, le niveau d'humidité à travers le capteur d'humidité, ainsi que le besoin en air et l'exposition au soleil.

Clés : agriculture, microcontrôleurs, PIC16F84A, pompe à rayons, humidité, air, soleil.

Abstract

Agriculture is one of the oldest and most important activities that man has practiced and continues to practice because of the great importance it is for him to ensure its food.

But get quality, save time and reduce the effort on himself have forced to embark on the scanning and use techniques that contribute to us, because we can adopt many technologies, such as microcontrollers.

In our project, we used (PIC16F84A) to control the irrigation pump, humidity level through moisture sensor, as well as air and exposure to the sun exposure.

Keys : agriculture, microcontrollers, PIC16F84A, ray pump, humidity, air, sun.

ملخص

تعد الزراعة من اقدم وأهم الانشطة التى مارسها الانسان ولا يزال يمارسها لما تحمله من أهمية عظمة له لتأمين غذائه .

لكن الحصول على جودة واكتساب الوقت وتقليل الجهد على نفسه تطلب عليه التوجه لعالم الرقمنة واستعمال تقنيات تساهم في ذالك ، اذ يمكننا تبني العديد من التقنيات ، مثل المتحكمات الدقيقة .

في مشروعنا هذا استخدمنا (PIC16F84) من أجل التحكم في مضخة الري ، ومستوى الرطوبة من خلال مستعشر الرطوبة ، وكذالك التحكم في الحاجة للهواء والتعرض لشمس .

المفاتيح : الزراعة ، المتحكمات الدقيقة ، PIC16F84A ، مضخة الراي ، الرطوبة ، الهواء ، الشمس .

Table des matières

Table des figures

Liste des tableaux

Liste des sigles et acronymes

LCD	*liquid Crystal Display*
LED	*Light Emetting Diode*
USB	*Universal Serial Bus*
PWM	*Pulse width Modulation*
A/N	*Convertisseur analogique – numérique*
N/A	*Convertisseurs Numériques – Analogiques*
E/S	*Entré Sortie*
CPU	*Unité centrale de traitement*
PIC	*Programmable Interface Contrôler*
ALU	*Unité arithmétique et logique*
MCLR	*Reset*
I2C	*(IIC) Inter Integrated Circuit*
ISIS	*Logicielle de simulation*

Introduction générale

Il ya plusieurs années, l'agriculture a connu un grand développement dans toutes les régions concernées par la volonté d'augmenter la qualité et la quantité des produits agricoles pour faire face au marché concurrentiel, les systèmes d'irrigation sont devenus très développés et cet objectif a été atteint grâce à l'automatisme. contrôle du système dans les serres.

Afin d'assurer nos plantes vertes et saines, de gagner du temps et de nous permettre de partir en vacances en toute tranquillité sans que je n'ai besoin de demander l'aide des autres, nous allons donc mettre en œuvre le projet de système d'irrigation intelligent, un microcontrôleur (PIC16F84A) a été choisi pour contrôler la pompe à eau et son fonctionnement, ainsi que pour contrôler les processus suivants : humidité et ventilation et exposition au soleil, et le microcontrôleur a prouvé son existence dans tous les domaines car il est largement utilisé pour contrôler les serres en raison de sa simplicité et de sa facilité utile.

Le premier système informatique sur une puce optimisée pour les applications de contrôle était un microprocesseur Intel 8048 avec à la fois RAM et ROM sur la même puce. La plupart des microcontrôleurs à cette époque avaient deux variantes ; L'un avait une mémoire de programme EEPROM, qui était beaucoup plus chère que la variante PROM qui n'était programmable qu'une seule fois. L'introduction de l'EEPROM a permis aux microcontrôleurs (à commencer par Microchip PIC16 $\times$ 84) d'essuyer électriquement rapidement sans le package coûteux requis pour l'EPROM. La même année, Atmel a présenté le premier microcontrôleur utilisant la mémoire flash. D'autres entreprises ont rapidement emboîté le pas. De nos jours, les microcontrôleurs sont peu coûteux et facilement accessibles aux amateurs, avec de grandes communautés en ligne autour de processeurs spécifiques. Traditionnellement, les microcontrôleurs sont programmés en utilisant le langage d'assemblage de l'équipement cible. Bien que le langage d'assemblage soit rapide. Les microcontrôleurs fabriqués par différentes sociétés ont différents langages d'assemblage, de sorte que l'utilisateur doit apprendre une nouvelle langue avec tout ce qu'il utilise. Les microcontrôleurs peuvent également être programmés en utilisant un langage de haut niveau, tel que BASIC, PASCAL ou C. Les langages de haut niveau sont plus faciles à apprendre que les langages d'assemblage et ils facilitent également le développement de programmes volumineux et complexes..[**5**]

On va résumer notre projet en 4 chapitres principaux :

- Le 1er chapitre Etat de l'art sur les microcontrôleurs.
- Le 2éme chapitre les langages de programmation de microcontrôleurs.

- Le 3eme chapitre Description et rédaction du programme.
- Le 4eme chapitre la partie de simulation de programmation ainsi que la réalisation du système.

Et enfin, une conclusion générale sanctionnera l'ensemble de ce travail.

Chapitre 1

Etat de l'art sur les microcontrôleurs

1.1 Introduction

Nous vivons aujourd'hui à l'ère des technologies de l'information, où notre ère connaît un développement rapide, notamment dans le domaine des technologies informatiques et du grand nombre d'appareils qui ont été inventés dans ce domaine, comme le microcontrôleur, devenu ces derniers temps années, L'un des éléments électroniques les plus importants, et cela est dû à de nombreuses raisons, notamment : que vous pouvez le programmer pour qu'il exécute tout ce que vous voulez, et pas seulement cela, mais vous pouvez le reprogrammer plusieurs fois si un changement se produit dans votre esprit que vous souhaitez ajouter, en plus bien sûr de sa petite taille et c'est ce qui le distingue de l'ordinateur ce qui l'a fait Il remplace l'ordinateur dans de nombreux À partir des applications de contrôle. Le microcontrôleur est actuellement utilisé dans de nombreuses applications et industries. Il est utilisé dans l'industrie automobile, en particulier les voitures modernes qui contiennent une commande automatique, et est utilisé dans la fabrication d'appareils électroménagers, de jouets et de commande de robots, ainsi que dans ses utilisations dans diverses commandes. processus tels que le contrôle et le contrôle de la température.

1.2 Généralités sur le microcontrôleurs :

Le microcontrôleur est un dérivé du microprocesseur. Sa structure est celle des systèmes à base de microprocesseurs. Il est donc composé en plus de l'unité centrale de traitement, d'une mémoire (mémoire vive RAM et mémoire morte ROM), une (ou plusieurs) interface de communication avec l'extérieur matérialisé par les ports d'entrée/sortie. En plus de cette configuration minimale, les microcontrôleurs sont dotés d'autres circuits d'interface qui vont dépendre du microcontrôleur choisi à savoir les systèmes de comptage (TIMER), les convertisseur analogique/numérique (CAN) intégré, gestion d'une liaison série ou parallèle, un Watch dog (surveillance du programme), une sortie PWM (modulation d'impulsion),…

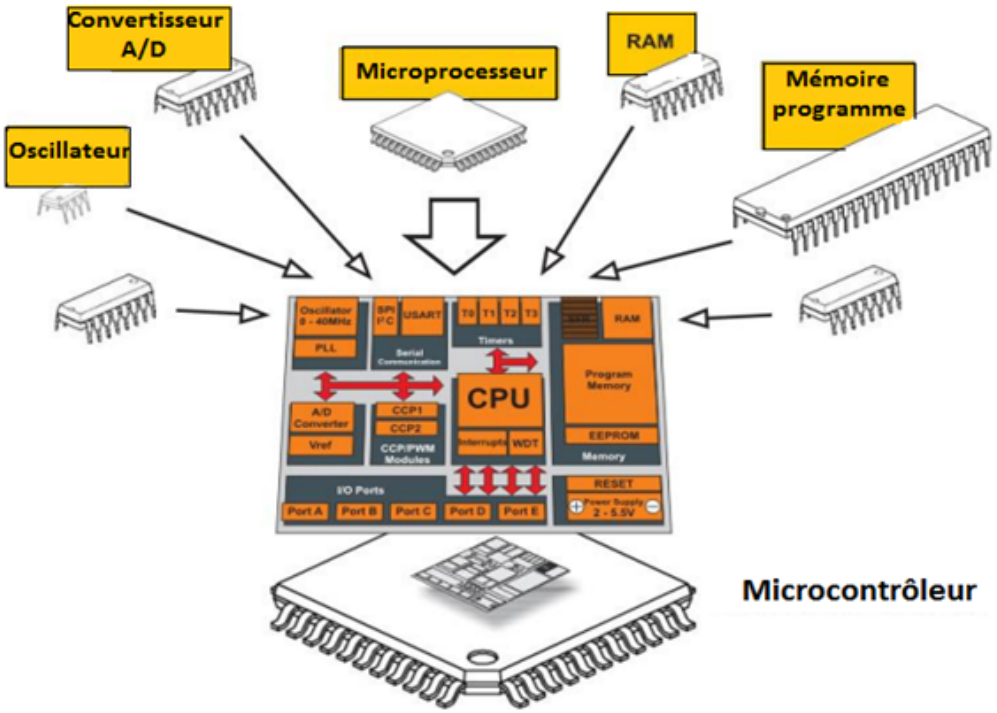


FIG. 1.1 : Contenu type d'un micro contrôleur

[**1**]

Les microcontrôleurs améliorent l'intégration et le coût (lié à la conception et à la réalisation) d'un système à base de microprocesseur en rassemblant ces éléments essentiels dans un seul circuit intégré. On parle alors de "système sur une puce" (en anglais : "System On chip"). Les microcontrôleurs sont plutôt dédiés aux applications qui ne nécessitent pas une grande quantité de calculs complexes, mais qui demandent beaucoup de manipulations d'entrées/sorties. C'est le cas de contrôle de processus. Les systèmes à microprocesseur sont plutôt réservés pour les applications demandant beaucoup de traitement de l'information et assez peu de gestion d'entrées / sorties. Les ordinateurs sont réalisés avec des systèmes à microprocesseur. Il existe plusieurs famille de microcontrôleurs dont les plus connues sont :

Atmel : AT ; familles AT89Sxxxx, AT90xxxx, …

Motorolla : famille 68HCxxx, …

Microchip : PIC ; familles 12Cxxx, 16Cxxx, 16Fxxx, 18Fxxx, …

Intel : famille 80C186XX

STMicroelectronics : famille STX

Analog Devices : famille ADuC

Nous allons nous intéresser dans le cadre de ce cours à la famille Microchip PIC (Programmable Integrater Circuit) de moyenne gamme (MIDRANGE). [**1**]

1.2.1 Définition le microcontrôleurs :

Un microcontrôleur (C) est une unité de traitement de l'information de type microprocesseur, c'est un ordinateur monté dans un circuit intégré rassemblant dans un même boitier une unité arithmétique et logique. Les avancées technologiques en matière d'intégration, ont permis d'implanter sur une puce de silicium de quelques millimètres carrés la totalité des composants qui forment la structure de base d'un ordinateur. Comme tout ordinateur, on peut décomposer la structure interne d'un microprocesseur en trois parties :

- Les mémoires, -Le processeur,
- Les périphériques
- Les mémoires sont chargées de stocker le programme qui sera exécuté ainsi que les données nécessaires et les résultats obtenus.
- Le processeur est le cœur du système puisqu'il est chargé d'interpréter les instructions du programme en cours d'exécution et de réaliser les opérations qu'elles contiennent .Au sein du processeur, l'unité arithmétique et logique ALU interprète, traduit et exécute les instructions de calcul.
- Les périphériques ont pour tâche de connecter le processeur avec le monde extérieur dans les deux sens. Soit le processeur fournit des informations vers l'extérieur (périphérique de sortie), soit il en reçoit (périphérique d'entrée). Les microcontrôleurs sont de taille tellement réduite qu'ils peuvent être sans difficulté implantés sur l'application même qu'ils sont censés piloter. Leur prix et leurs performances simplifient énormément la conception de système électronique et informatique

[**18**]

1.2.2 l'architecteur de microcontrôleurs :

Le microcontrôleur est un dérivé du microprocesseur. Sa structure est celle des systèmes à Base de microprocesseurs. Il est donc composé d'une unité centrale de traitement et de commande (équivalente au microprocesseur), de mémoires et de ports d'entrées/sorties. En plus de cette configuration minimale, les microcontrôleurs sont dotés d'un ou plusieurs systèmes de comptage (TIMER). Quelques-uns sont dotés d'un convertisseur analogique/numérique (CAN) intégré. Ces atouts supplémentaires permettent de nombreuses applications telles que :

- acquisition et traitement de données analogiques (CAN)
- comptage d'événements (TIMER)
- mesure de fréquence ou de période (TIMER)
- génération d'impulsions (TIMER)
- les programmes peuvent être différents (gestion d'un thermostat intelligent, d'une photocopieuse..)
- les programmes ont en commun peut de calculs complexes contrairement à un système informatique)

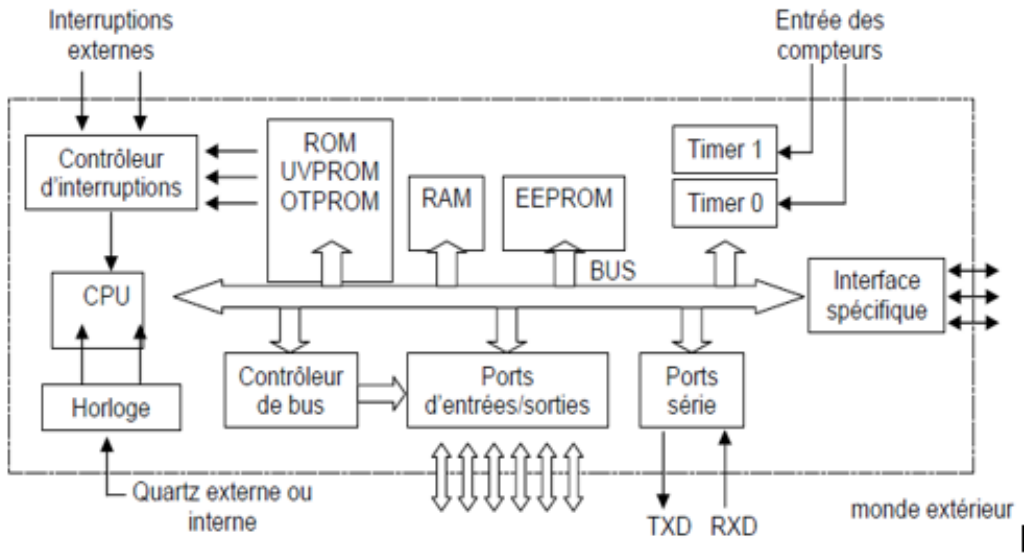


FIG. 1.2 : l'architecteur de microcontrôleurs

[**3**]

Un système minimal programmable pour fonctionner à besoin :

- d'une unité centrale
- de mémoires morte pour le programme (PROM, EPROM,….)
- de mémoires vives, pour les calculs, pour stocker les données,
- de circuits interfaces, pour connecter les périphériques qui vont permettre la communication avec l'extérieur d'où l'apparition des microcontrôleurs (ou Monochip) Dans un seul circuit on va trouver :
- Une Horloge (oscillateur)
- Un processeur (Unité centrale)
- De la mémoire vive (RAM)
- De la mémoire morte (PROM, EPROM, EEPROM)
- Des interfaces qui vont dépendre du type de microcontrôleurs choisi

Compteurs/Timer

Convertisseurs Analogiques/numériques (C.A.N.)

Chien de garde (« Watch Dog »)

Gestion d'un port parallèle (d'entrée/sortie)

Gestion d'une liaison série RS232

Gestion des interruptions

Gestion des moteurs en PWM (pulse w modulation)

Gestion d'écran LCD

Gestion de bus I2C Etc… Il suffit de choisir le microcontrôleur le mieux adapté à l'application que l'on doit réaliser. !

***La ROM contient le programme à exécuter :**

Contrairement à un système informatique classique, il n'est pas question de charger un programme en mémoire vive à partir d'un disque ou d'une disquette car l'application doit démarrer dès la mise sous tension et ne possède pas d'organe de ce type.

*** La RAM ou mémoire vive (Random Access memory : mémoire à accès aléatoire) :**

On les appelle comme ça de façon impropre LES ROM sont aussi a accès aléatoire Ces mémoires perdent l'information lorsqu'elles ne sont plus alimentées. Pour pouvoir travailler normalement le µcontrôleur doit pouvoir a souvent besoin de stocker des données temporaire que part et c'est la qu'intervient la RAM. Contrairement a un système informatique classique la RAM d'un µcontrôleur est de petite taille.

***Les entrées sorties :**

Les circuits d'interfaces peuvent piloter des matériels très différents, (moteur pas à as, afficheur LCD, thermistance, communication avec des pc ou d'autres µcontrôleurs, etc…)

***Le bus système :**

L'unité centrale doit pouvoir communiquer avec les mémoires et les périphériques.

Exemple :

pour écrire une donnée en mémoire, l'UC doit d'abord spécifier l'adresse de la mémoire, puis envoyer la donnée, et en dernier lieu, envoyer un signal qui validera la mémorisation de la donnée. Tous ces signaux seront véhiculés par les « bus », ensembles de « conducteurs », sur lesquels viennent se brancher les mémoires, les interfaces des périphériques. On distingue 3 types de bus
- le bus d'adresses
- le bus de données
- le bus de contrôle (pour les signaux de service)

***Les différents types de mémoires dans les microcontrôleurs :**

Le prix du microcontrôleur dépend fortement du type de mémoire qu'il contient. Outre les différents périphériques possibles, les différents types de mémoire constituent les différentes gammes de microcontrôleurs de même architecture.

***La ROM qui contient le programme à exécuter (plusieurs kilo octets) :**

Ne s'efface pas hors tension

PROM :

(Programmable Read Only memory) On les appelle aussi mémoire fusibles ou OTP (One Time Programmable) La programmation de ce type de mémoires consiste à faire « claquer » des fusibles (qui sont en fait des jonctions semi-conductrices)

EPROM :

(Erasable Programmable Read Only Memory) ou UVPROM Ce sont des mémoires programmables électriquement et effaçable par UV donc Réutilisables. Il faut noter que l'effacement par UV (environ 15 min) et l'écriture (que minutes) sont des opérations relativement longues. Nécessite une petite fenêtre en quartz pour laisser passer les UV (principe utilisé :utilisation de transistors FAMOS : Floating Gâte Avalanche Injection Métal Oxide Silicium)
- apparition d'OTPROM qui sont des UVPROM sans fenêtre et donc non réutilisable mais pas chère.

Autres :

EPROM :

(Erasable Programmable Read Only Memory) Ce sont des mémoires programmables et effaçables électriquement, ou aussi mémoires

FLASH :

Procédé beaucoup plus rapide que lors d'une exposition UV.

la RAM :

(mémoire des données, variables, piles, etc...) (Quelques octets) S'efface hors tension.[**3**]

1.2.3 caractéristique principale des microcontrôleurs :

La tendance actuelle est de s'éloigner de plus en plus des microcontrôleurs 8 bits afin de s'approcher des microcontrôleurs 32 bits. Les processeurs 8 bits sont utilisés dans les applications qui ont pour critères principaux la consommation et le coût. Les processeurs 16 bits sont relativement peu répandus. Il existe néanmoins quelques exemples comme le MSP430 de TI. Le prix des processeurs 32 bits actuels, comme le Cortex-M3, est devenu tellement intéressant que ces derniers remplacent de plus en plus les processeurs 8 bits Les processeurs 32 bits les moins chers coûtent environ un eur les microcontrôleurs 8 bits présentent les avantages suivants faces aux microcontrôleurs 32 bit
- Faible consommation
- Bas coût

- Dimension réduite Les microcontrôleurs 32 bits présentent les avantages suivants faces aux microcontrôleurs 8 bits
- Puissance et vitesse de calcul supérieures
- Convient pour l'utilisation d'un système d'exploitation. En effet les performances de la CPU et la taille de la mémoire sont suffisantes
- Espace d'adressage plus grand
- Taille de mémoire requise pour les microcontrôleurs dans les systèmes embarqués La taille de la mémoire programme s'étend typiquement de quelques kilooctets à quelques centaines de kilooctets. Cette règle est valable pour les microcontrôleurs 8 et 32 bits La taille de la mémoire pour les données s'étend de quelques centaines d'octets à quelques centaines de kilooctets Le portage d'un système d'exploitation embarqué comme celui de Linux ou de Windows CE augmente considérablement l'exigence concernant la mémoire programme ou de données à quelques mégaoctets.[**4**]

1.2.4 déférence famille de microcontrôleurs :

- la famille Atmel AT91 ;
- les familles ARM Cortex-M et ARM Cortex-R
- la famille Atmel AVR (utilisée par des cartes Wiringet Arduino) ;
- le C167 de Siemens/Infineon ;
- la famille des Infineon AURIX TC3x, Infineon AURIX TC2x, Infineon TriCore TC1x, Infineon XMC, XC2000 de Infineon Technologies ;
- la famille Hitachi H8 ;
- la famille Intel 8051, qui ne cesse de grandir ; de plus, certains processeurs récents utilisent un cœur 8051, qui est complété par divers périphériques (ports d'E/S, compteurs/temporisateurs, convertisseurs A/N et N/A, chien de garde, superviseur de tension, etc.) ;
- l'Intel 8085, à l'origine conçu pour être un microprocesseur, a en pratique souvent été utilisé en tant que microcontrôleur ;
- le Freescale 68HC11 ;
- la famille Freescale 68HC08 ;
- la famille Freescale 68HC12 ;
- la famille Freescale Qorivva MPC5XXX ;
- la famille des PIC de Microchip ;
- la famille des dsPIC de Microchip ;
- la famille ADuC d'Analog Devices ;
- la famille PICBASIC de Comfile Technology ;
- la famille MSP430 de Texas Instruments ;
- la famille 8080, dont les héritiers sont le microprocesseur Zilog Z80 (désormais utilisé en tant que contrôleur dans l'embarqué) et le microcontrôleur Rabbit ;
- la famille PSoC de Cypress Semiconductor ;

- la famille LPC21xx ARM7-TDMI de Philips ;
- la famille V800 de NEC ;
- la famille K0 de NEC ;
- la famille des ST6, ST7, ST10, STR7, STR9, de STMicroelectronics ;
- la famille STM32 de STMicroelectronics ;
- la famille STM8 de STMicroelectronics.[**5**]

1.3 Etat de l'art sur les microcontrôleurs :

1.3.1 Définition de PIC :

Le PIC (Programmable Interface Contrôler) est une unité de traitement de l'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 composant dits RISC (Reduced Instructions Set Computer), ou encore composant à jeu D'instructions réduites. Le microcontrôleur ce trouve, dans plusieurs appareils telle que : les Téléphones portables, machines à laver, télévisions vidéo .... Etc.[**6**]

1.3.2 Identification des PIC :

Un PIC est généralement identifié par une référence de la forme suivante : xx(L)XXyy-zz

Xx : famille du composant, actuellement « 12, 14, 16, 17 et 18 ».

L : tolérance plus importante de la plage de tension.

XX : type de mémoire programme

C : EPROM ou EEPROM

CR : PROM

F : Flash

Yy : Identification.

Zz : vitesse maximale du quartz de pilotage.[**2**]

Exemple :

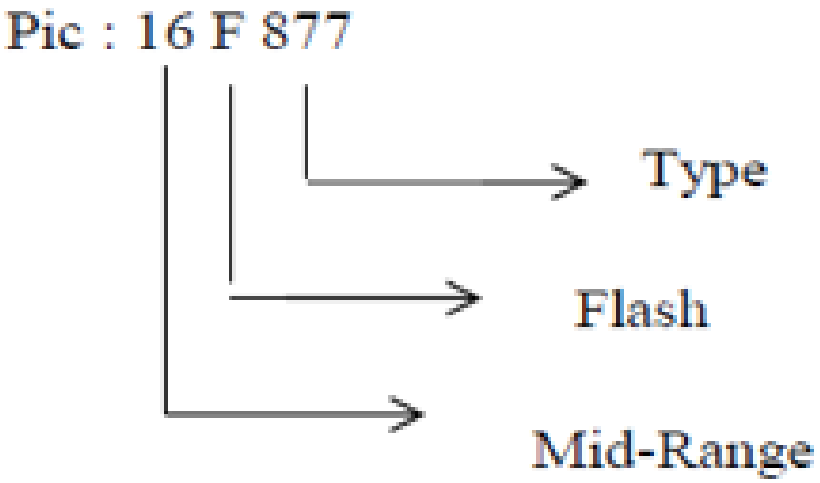


FIG. 1.3 : Exemple de l'identification PIC16F877

[7]

1.3.3 Choix et Présentation du PIC :

En premier lieu, ce modèle possédant une mémoire EEPROM effaçable électriquement, s'impose comme la solution idéale pour ceux qui veulent apprendre à utiliser un microcontrôleur PIC, du fait qu'il est reprogrammable jusqu'à plus de 1000 fois (selon les spécifications du fabricant). Associé à de simples organes périphériques, il représente l'outil d'apprentissage par excellence, car le lecteur peut tester tous les programmes avec le même microcontrôleur et revenir sur les erreurs, les corriger rapidement et de retenter l'application. Ce microcontrôleur possède un fusible interne, accessible par programmation, qu'il faudra se garder de laisser intact. Car lorsque ce fusible brûle, le microcontrôleur, s'il peut encore être effacé et reprogrammé, ne peut plus être lu correctement, car sa mémoire est restituée, complètement désorganisée. Bien utile pour ceux qui mettent au point une application commerciale qu'ils veulent protéger et mettre à l'abri de copies sauvages, ce fusible doit être ignoré pendant la durée de l'étude. Cette accessibilité permanente de la mémoire représente l'aspect le plus original de tous les microcontrôleurs à mémoire flash, parmi lesquels trône le 16F84 et 16F84A. De plus, la capacité mémoire de ce modèle (ni trop petite ni trop grande). Un 16F est un PIC Mid-Range dont la mémoire programme est de type FLASH de référence 84 pour les PIC 16F84 et 84A, capable d'accepter une fréquence d'horloge de 4MHz. Le PIC16F84 est un microcontrôleur 8 bits. Il dispose donc d'un bus de données de huit bits. Puisqu'il traite des données de huit bits, il dispose d'une mémoire de donnée dans laquelle chaque emplacement (défini par une adresse) possède huit cases pouvant contenir chacune un bit. L'organisation générale du PIC16F84 et 84A sont composées par 4 blocs principaux comme le montre la figure 1.4

- Mémoire de programme

- Mémoire de données
- Processeur
- Ressources auxiliaires (périphériques)

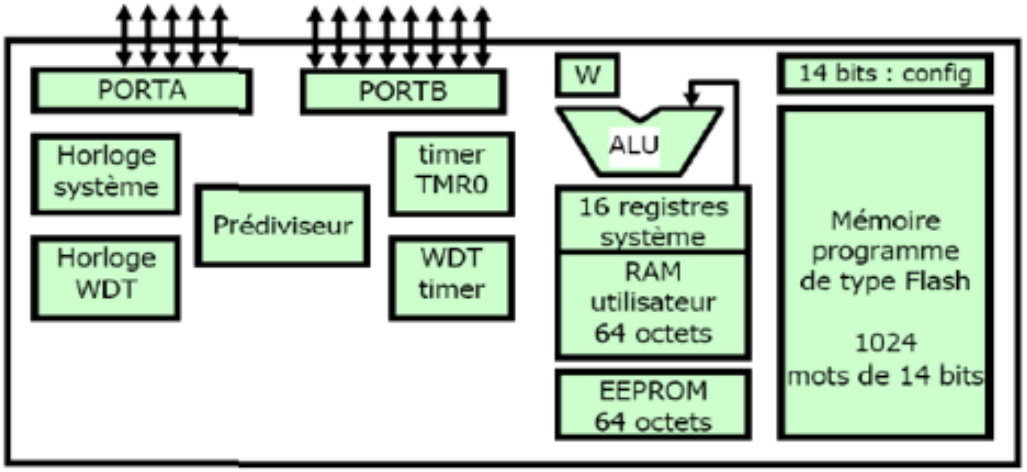


FIG. 1.4 : Structure de l'architecture interne et diagramme de blocs des PIC16F84A et 16F847 [**2**]

- La mémoire de programme : Il s'agit d'une mémoire non volatile de type FLASH programmable et effaçable. Pour le PIC 16F84 cette mémoire est d'une taille de 1024*14 bits, les instructions sont codées sur 14 bits. On peut donc stocker 1024 instructions.
- La mémoire de donnée est séparée en deux parties :

-une mémoire RAM de 68 octets puisque le bus de donnée est de huit bits. Cette RAM est volatile (les données sont perdues à chaque coupure de courant). On peut y lire et écrire des données.

-une mémoire EEPROM de 64 octets dans laquelle on peut lire et écrire des données (de huit bits soit un octet) et qui possède l'avantage d'être non volatile (les données sont conservées même en l'absence de tension). La lecture et l'écriture dans cette mémoire de données sont beaucoup plus lentes que dans la mémoire de données RAM.

- Le processeur est formé de deux parties :

-une unité arithmétique et logique (ALU) chargée de faire des calculs.

-un registre de travail noté W sur lequel travail l'ALU.

- Les ressources auxiliaires qui sont dans le cas du PIC16F84

- ports d'entrées et de sorties ;
- temporisateur ;
- interruptions ;
- chien de garde ;
- mode sommeil.[**2**]

1.3.4 Caractéristiques Principales :

Le PIC16F84 est un circuit intégré de 18 broches (Fige 1.5), ses principales caractéristiques sont :
- 13 lignes d'entrées/sorties, réparties en un port de 5 lignes (Port A) et un port de 8 lignes (Port B) ;
-Une alimentation sous 5 Volts ;
-Une architecture interne révolutionnaire lui conférant une extraordinaire rapidité ;
-Une mémoire de programme pouvant contenir 1024 instructions de 14 bits chacune (allant de 000h à 3FFh) ;
-Une mémoire RAM utilisateur de 68 emplacements à 8 bits (de l'adresse 0C à l'adresse 4F) ;
-Une mémoire ROM de 2x12 emplacements réservée aux registres spéciaux ;
-Une mémoire EEPROM de 64 emplacements ;
-Une horloge interne, avec pré diviseur et chien de garde, possibilité d'être programmé in-circuit, c'est à dire sans qu'il soit nécessaire de le retirer du support de l'application – vecteur de Reste situé à l'adresse 000 ;
-Un vecteur d'interruption, situé à l'adresse 004.
-Un bus d'adresses de 13 lignes.
-Présence d'un code de protection permettant d'en empêcher la duplication.
- Facilité de la programmation- simplicité..

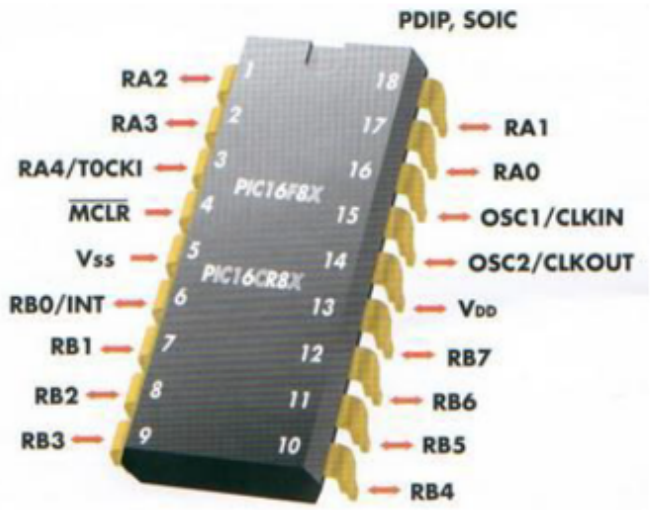


FIG. 1.5 : Architecture externe et brochage des pins du PIC16F8X.

[**8**]

-Les ports d'entrées / sorties :

Permettent de dialoguer avec l'extérieur du microcontrôleur, le PIC 16F84 possède 13 lignes d'entrées / sorties : RB0 à RB7 et RA0 à RA4.

-RA0 à RA4 :

Constituent le (PORTA) du microcontrôleur, ce port est bidirectionnel. La ligne RA4 de type drain ouvert en sortie peut aussi être utilisée comme entrée d'horloge du compteur (Timer), TMR0.

-RB0 et RB7 :

Constituent le PORTB du microcontrôleur, ce port est bidirectionnel. La ligne RB0 peut servir d'entrée de demande d'interruption externe.

-VDD et VSS :

Ce sont les connexions d'alimentation du circuit .Il est alimenté avec une tension de 5 volts VSS = 0v, VDD = +5v.

-MCLR :

Cette connexion active au niveau 0, est l'entrée de Reset (Master CLear Reset), elle permet aussi le branchement de la haute tension VPP nécessaire à la programmation du composant. Le pin MCLR peut être simplement relié à VDD si on n'a pas besoin de RESET externe. Par contre si on souhaite implanter un bouton de remise à zéro, on pourra câbler un simple réseau RC sur la broche MCLR.

-OSC1 et OSC2 :

Ces connexions permettent l'entrée des signaux nécessaires au fonctionnement de l'horloge. L'horloge est un système qui peut être réalisée soit avec un QUARTZ(a), soit avec un circuit RC(b), soit avec une horloge extérieure (c), la fréquence maximale d'utilisation va dépendre de Microcontrôleur utilisé. Le suffixe indiqué sur le boîtier donne la nature de l'horloge à utiliser et sa fréquence maximale (figure 1.6).[**8**]

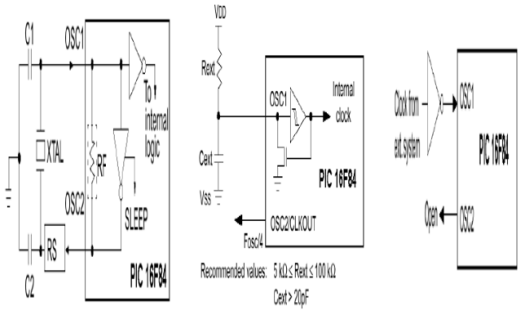


FIG. 1.6 : Conception des différentes configurations de l'oscillateur des PIC16F

[**8**]

1.4 Conclusion

Nous avons vu dans ce chapitre une étude générale sur les microcontrôleurs et en ce qui concerne les microcontrôleurs PIC, car ce dernier présente des caractéristiques d'utilisabilité très intéressantes car il réduit l'utilisation de nombreux composants électroniques tels que les circuits logiques et les oscillateurs

Chapitre 2

les langages de programmation de microcontrôleurs

2.1 Introduction

Les microcontrôleurs sont des composants programmables qui font ce que le programme exige d'eux, car le programme est une série d'instructions et de commandes, qui sont codées en binaire, ce qui signifie qu'il fonctionne à 0 ou à 1, de sorte qu'il peut être exécuté par le microcontrôleur.

Pour écrire un programme, il existe plusieurs solutions, mais chacune d'elles est une forme de traduction .Le programme peut également être écrit en langage de programmation d'assemblage ou en langage de programmation C.

Dans ce chapitre, nous nous concentrerons sur la programmation en langage assembleur ainsi qu'en langage C.

2.2 Principaux langages de programmation :

Le processeur d'un microcontrôleur ne traite que des 1 et des 0 (langage machine). Exemple : pour incrémenter le registre à l'adresse Oxof sur un PIC 16F737 : 00 1010 1 + Adresse du registre Soit : 00 1010 1000 1111 Une instruction simple peut alors devenir compliquer à mettre en œuvre et à lire. Pour pallier à ce problème, des langages de programmation sont mis en place afin de faciliter leur utilisation. Les deux principaux langages que nous allons évoquer sont : L'assembleur Le langage C.[**9**]

2.2.1 Nécessaire à la programmation :

Quand on veut programmer un PIC, il est nécessaire d'avoir un Minimum de documents avec soi. Cependant, si ces derniers sont Importants pour programmer en langage assembleur, nous verrons plus Tard qu'en langage C, certains deviennent quasi inutiles.

Ces documents, peu nombreux, traitent des tableaux de pages de Programmation, de la répartition des ports en entrées/sorties et du type D'oscillateur utilisé (en effet, dans le cas par exemple d'un oscillateur RC, l'instruction SLEEP est alors utilisable) Dans un premier

Dans ce tableau, nous ne nous réfèrerons qu'à la ligne : STATUS TRISA TRISB Les autres lignes entrant dans des modes spécifiques du PIC, ou Servant juste à indiquer dans quel état se trouve le PIC, nous ne les Détaillerons pas.

STATUS :

: Ce registre contient tous les bits qui servent pour nous renseigner sur l'état du PIC, et pour une partie de la programmation. Seul trois bits nous intéressent : le C, le RP1 et le RP0. Le bit C (pour carry, retenue en français), est à un ou zéro selon que l'opération effectuée nécessitait une retenue ou non. Le RP0 et le RP1 sont les bits permettant d'avoir accès au tableau de programmation, vu précédemment (et donc au bit système des registres) et aux différents espaces mémoires. Ils respectent le tableau suivant :

ADRESSES HEXA	NOM	Bit7	Bit6	Bit 5	Bit4	Bit3	Bit2	Bitl	Bit0
00	INDF	Utilise le contenu de FSR pour adressera mémoire							
01	TMR0	Horloge/compteur en temps réel							
02	PLC	O :tet de moindre poids du FC							
03	STATUS	IRP	IRPI	RP0	TO#	PD#	z	DC	C
04	FSR	Adressage indirect avec INDF							
05	PORTA				RA4 1Tock	RA3	RA2	RAI	RAO
06	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0 IINT
07		Non implérrenté, lu comme étant à 0							
08	EEDATA	Registre de données E'PROM							
09	EEDATA	Registre d'adresses E'PROM							
0A	PCLATH				S'écrit surl,s 5 bts de PCH				
OB	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

TAB. 2.1 : tableau de programmation 0

RPI	RP0	PAGE	ADRESSES Hexa
0	0	0	00 à 7F
0	1	1	80 à FF
1	0	2	100 à 17F
1	1	3	180à IFF

TAB. 2.2 : tableau registre STATUS

TRISA :

Le TRISA est le registre permettant de définir les entrées/sorties du Port A. En remplaçant les RAx par leur valeur correspondante (1 pour une entrée, 0 pour une sortie), on obtient alors le mot binaire à rentrer dans TRISA. Quand au bit 7, 6 et 5, on peut leur donner la valeur que L'on veut, selon que ça nous arrange ou non.

TRISB :

Le TRISB fonctionne de la même manière que le TRISA, mais avec 2 Bits supplémentaires Les registres qui vont nous intéresser ici sont : EECON OPTION

INTCON :

Nous ne nous attarderons pas, mais ces registres EECON servent à Contrôler l'écriture et la lecture dans l'E²PROM. Ils doivent être utilisés avec les registres EEDATA et EEADR du tableau de program-

ADRESSES HEXA	NOM	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
80	INDF	Utilise le contenu de FSR pour adresser la mémoire							
81	OPTION	RBPU#	INTDG	TOCS	TOSE	PSA	PS2	PSI	PS0
82	PLC	Octet de moindre poids du PC							
83	STATIJS	IRP	IRPI	RP0	TO#	PD#	Z	DC	C
84	FSR	Adressage indirect avec INDF							
85	TRISA				Configuration du PORT A				
86	TRISB	Configuration du PORT 8							
87		Nonimplémenté, lu comme étant à 0							
88	EECONI				EEIF	WRERR	WREN	WRRD	
89	EECON2	Re st,e de commande E'PROM							
0A	PCLATH				Configuration du PORT A				
OB	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

Tab. 2.3 : tableau de programmation 1

mation 0.

OPTION :

Ce registre permet de définir le choix d'un diviseur et son affectation au Watch dog ou au RTCC Le choix de ce diviseur, se fait par l'intermédiaire de PS2, PS1, et PS0, Selon le tableau suivant :

PS2	PSI	PSO	Rapport de division	
			WDT	RTCC
0	0	0	1	2
0	0	1	2	4
0	1	0	4	8
0	1	1	8	16
1	0	0	16	32
1	0	1	32	64
1	1	0	64	128
1	1	1	128	256

Tab. 2.4 : tableau registre OPTION

L'affectation au Watch dog (WDT) ou à l'horloge temps réel (RTCC) se fait par l'intermédiaire du bit PSA. Dans le cas de l'affectation du diviseur au RTCC, le bit TOSE permet de choisir entre

un déclenchement sur front montant ou sur front descendant. Le bit INTEDG permet lui, de choisir le déclenchement de l'interruption (sur front montant ou sur front descendant). Enfin, le bit /RBPU, permet de choisir de la mise en place ou non d'une résistance de rappel entre RB4 et RB7. Tout ceci est résumé dans le tableau suivant :

Bit du regisn-e INTCON	NOM	Fonrtion
Bit 7	GIE (Global lnterrupt Enabled Bit)	Bit de validation général des interruptions (à 1 pour activer les interruptions)
Bit6	EEIE (E'PROM Write Complète lnterrupt Enabled Bit)	Interruption de la fin d'écritureen E2 PROM
Bit5	TOIE (Tuner 0 lnterrupt Enabled Bit)	Interruption pour le Tuner 0
Bit4	INTE (INTerrupt pin Enabled Bit)	Interruption sur la patte RB0
Bit 3	RBIE(RB port change lnterrupt *Enabled bit)	Interruption sur les pattes RB4 à RB7
Bit2	TOIF (Tuner 0 lnterrupt Flag bit)	Signal que l'interruption vient du débordement du Tuner 0
Bit 1	INTF (INTerrupt pin Flag bit)	Signal que l'interruption provient d'un changement d'état de RB0
Bit0	RBIF (port B lnterrupt F1ag bit)	Signal quel'interruption provient des pattes RB4 àRB7

Tab. 2.5 : tableau Activez et vérifiez les interruptions active

Ce tableau résume la façon dont on peut activer et vérifier les Interruptions actives.[**10**]

2.2.2 langages de programmation :

Quand on choisit de programmer un microcontrôleur, il faut savoir que sauf exceptions, cas rare, nous avons le choix entre plusieurs langages de programmation. En ce qui concerne le PIC, les programmes se développent avec le logiciel MPLAB (dont nous verrons le fonctionnement plus loin dans ce livre), qui nous permet de programmer le PIC dans divers langages. Parmi ceux-ci, les principaux sont le C et le BASIC pour les langages hauts niveaux, et l'assembleur pour les langages bas niveaux.

Les langages bas niveaux sont les plus proches de la machine, généralement l'équivalent de la « langue

natale » du système. L'assembleur en fait partie. Proche de la machine, ce langage permet d'optimiser le programme. Il reste cependant difficile à mettre en oeuvre pour des programmes complexes, mais abordable pour des programmes de base. Adopté par nombre d'industriels comme langage de référence,il reste cependant difficilement réparable pour une tierce personne.

Les langages hauts niveaux sont des langages faciles à comprendre et à mettre en œuvre, avec un minimum de connaissance. Ce type de Langage facilite la maintenance lors de problèmes, car il est aisé pour Quelqu'un, pris au hasard, de comprendre un programme haut niveau. Ce type de langage est, en effet, plus proche de l'utilisateur que de la machine. Ces langages utilisent ce qu'on appelle des « compilateurs »,sorte de convertisseurs, dictionnaires automatiques. Ces compilateurs Font la conversion directe entre le langage haut niveau et le langage Assembleur, nous épargnant ainsi une difficulté supplémentaire.

En comparant ces deux types de langage, et dans la mesure où les Compilateurs sont, de nos jours, performant, il apparaît comme plus Pratique de choisir le langage C, plutôt que le langage assembleur, tant pour la simplicité de programmation, que pour la facilité d'utilisation.

Ainsi, nous verrons dans ce livre, les bases de l'assembleur pour PIC (Instructions de commande) à fin d'informations, et éventuellement de comprendre certains programmes.

Puis nous nous attarderons sur le langage C, langage de Programmation choisi. Nous commencerons par voir les bases classiques du C général, puis nous verrons ensuite le C PIC, spécifique au PIC.[**10**]

2.2.3 le langage assembleur :

Le langage assembleur est ce qu'on appelle un langage « bas niveau », c'est-à-dire que c'est un langage extrêmement proche du fonctionnement du système. Très difficile à appréhender, il est cependant nécessaire d'avoir des bases en assembleur, car, cela permet d'optimiser les programmes et de gagner de la place, chose importante dans certains cas, car celle-ci est précieuse sur le PIC. Il faut savoir que quand, sur les documentations techniques, on lit le mot « instruction », celles-ci font référence à une instruction assembleur. C'est le langage par défaut du PIC, sa « langue natale ». Cependant, nous nous attarderons bien plus sur le langage C, successeur à l'assembleur, et bien plus compréhensible. Nous ne verrons donc en assembleur, que les 35 instructions du PIC avec quelques explications sur chacune d'entre elles. Voici donc pour commencer les instructions du PIC. Il n'y en a que 35, et le PIC ne se programme qu'avec elles. Les registres (W et f) contiennent des littérals (k), un peu comme un dossier (registres) contient des fichiers (littérals), mais où ces derniers seraient du texte.[**10**]

Mnémonique	Opérande 1 (source)	Opérande 2 (cible)	Opération réalisée	Nombre de cycle
Addlw	k		Ajoute k à w	1
Addwf	f	d	Ajoute f à w. Si d=0, le résultat est mis dans w ; si d=1, le résultat est mis dans f	1
andlw	k		Réalise un ET logique entre w et k	1
andwf	f	d	f&w, si d=0, le résultat est mis dans w, dans f sinon	1
bcf	f	b	Mise à O du bit b de f	1
bsf	f	b	Mise à 1 du bit b de f	1
btfsc	f	b	Si le bit b de f est nul, l'instruction suivante est ignorée	1-2
btfss	f	b	Si le bit b de f est à 1, l'instruction suivante est ignorée	1-2
call	k		Appel le sous programme d'étiquette k. A utiliser avec le mnémonique return	2
clrf		f	Charge la valeur 00 dans f	1
clrw			Charge la valeur 00 dans w	1
clrwdt			Efface la valeur du chien de garde	1
comf	f	d	Complément à 1 de f. Si d=0, le résultat est mis dans w ; dans f sinon	1
decf	f	d	Décrémente f d'une unité. Si d=0, le résultat est mis dans w ; si d=1, le résultat est mis dans f	1
decsfz	f	d	Décrémente f d'une unité. Si le résultat est 0, l'instruction suivante est ignorée. Si d=0,	1-2
			le résultat est placé dans w, dans f si d=1	
goto	K		Branchement inconditionnelle à l'étiquette k	2
call	k		Appel le sous programme d'étiquette k. A utiliser avec le mnémonique return	2
clrf		f	Charge la valeur 00 dans f	1
clrw			Charge la valeur 00 dans w	1
clrwdt			Efface la valeur du chien de garde	1
comf	f	d	Complément à 1 de f. Si d=0, le résultat est mis dans w ; dans f sinon	1

Tab. 2.6 : tableau le langage assembleur

[**10**]

decf	f	d	Décrémente f d'une unité. Si d=0, le résultat est mis dans w ; si d=1, le résultat est mis dans f	1
decsfz	f	d	Décrémente f d'une unité. Si le résultat est 0, l'instruction suivante est ignorée. Si d=0,	1-2
			le résultat est placé dans w, dans f si d=1	
goto	K		Branchement inconditionnelle à l'étiquette k	2
incf	F	d	Incrémente f d'une unité. Si d=0, le résultat est mis dans w ; si d=1, le résultat est mis dans f	1
incsfz	F	d	Incrémente f d'une unité. Si le résultat est 0, l'instruction suivante est ignorée. Si d=0, le résultat est mis dans w ; dans f sinon	1-2
iorlw	K		OU logique entre k et w	1
iorwf	F	d	f +w, si d=0, le résultat est mis dans w, dans f sinon	1
movf	F	d	Déplace f vers w si d=0, vers f si d=1	1
movlw	K		Déplace k dans w	1
movwf		f	Déplace w dans f	1
nop			Instruction vide	1
retfie			Retour de sous programme d'interruption	2
retlw	K		Retour de Sous programme, avec chargement de k dans w	2
return			Retour de sous programe	2
rlf	F	d	Rotation d'un bit vers la gauche de f. Si d=0, le résultat est mis dans w, dans f sinon	1
rrf	F	d	Rotation d'un bit vers la droite de f. si d=0, le résultat est mis dans w, dans f sinon	1
sleep			Mise en sommeil du pic (basse consommation)	1
sublw	K		Soubstrait w à k (k-w)	1
subwf	F	d	f-w, si d=0, le résultat est mis dans w ; dans f sinon	1
swapf	F	d	Inversion des 4 bits de poids forts, avec les 4 bits de poids faibles	1
xorlw	F	d	f+w exclusif, si d=0, le résultat est mis dans w, dans f sinon	1
xorlw	K		OU exclusif entre k et w	1

TAB. 2.7 : tableau le langage assembleur[**10**]

2.2.4 le langage C :

Dans cette partie, nous verrons les bases du langage C pour ordinateur, en nous limitant toutefois aux points communs avec le langage C PIC, c'est-à-dire aux commandes communes. Il est cependant à spécifier, que pour des raisons de commodité, nous verrons quelques points en plus, afin d'expliquer clairement le langage C ; points existant en C PIC, mais sous formes différentes. Gardons donc à l'esprit que les bases du C sont ici données, afin d'apprendre à maîtriser ce langage haut niveau afin de savoir, plus tard, programmer les PICs avec le plus de facilité possible. A noter tout de même, que la meilleure école pour maîtriser un langage reste la pratique de celui-ci et l'étude de programme. Un programme en langage C se compose de plusieurs parties :
-Un appel de librairies
-Une fonction « main », correspondant au programme principal
-Eventuellement de sous-programmes (auxquels cas, à déclarer) [**10**]

Les librairies :

Elles constituent une sorte d'encyclopédie pour le programme principal. En effet, elles contiennent toutes les infos et directives dont a besoin le programme principal : le main. Le main a accès à ces directives par l'intermédiaire d'instructions spécifiques aux librairies en question. Pour résumer, et faire simple, le main appelle les fonctions dans les librairies par leur nom. On les appelles avec la commande « include<nom.h> ».

La fonction main :

Il s'agit du programme principal. C'est dans ce programme qu'on écrira notre code informatique. Cette fonction peut faire appel à des sous fonctions, lesquelles peuvent être situées dans des librairies ou derrièrele main. Il s'agit de programmes secondaires, sensés aider le main dans son travail. L'utilité de créer des sous fonction apparaît surtout lorsqu'une tache se répète plusieurs fois dans le même programme : plutôt que d'écrire par exemple 3 fois le même code, on définit une seule fois le code, dans un sous programme, et on l'appelle trois fois. On économise ainsi de la place (optimisation du code), chose très importante pour un microcontrôleur. La création possible de sous programme doit être une des choses à avoir à l'esprit lorsqu'on cherche à optimiser le code. Lorsque la possibilité se présente, il faut alors juger s'il est plus judicieux de créer un sous programme, ou alors de laisser le code tel quel, en se référençant à la longueur du programme. [**10**]

Les bases du langage C :

Quelques règles :
-Une ligne se finit toujours (sauf cas particulier que nous verrons) par un point virgule « ; »
-Il est jugé appréciable d'écrire avec l'aide des tabulations, pour avoir un code lisible. Les commentaires :

Lorsque l'on désire faire des commentaires en langage C, il y a deux solutions :
-Pour une ligne seule, on utilise un double slash : « // » Il n'est pas nécessaire de mettre un point virgule à la fin du Commentaire.
- Pour un paragraphe, ou plusieurs lignes consécutives, on utilise Un « /* » pour indiquer le début du commentaire, et un « */ » pour en indiquer la fin. [**10**]

Exemple :

a=b ; //voici un commentaire sur une ligne /*Voici un commentaire sur plusieurs lignes*/

Les variables :

Dans un programme, on utilise souvent des variables pour effectuer des opérations (ex : a=b). Il faut savoir que le langage C fait la différence entre majuscule et minuscule ; et que les variables sont la première chose à définir dans un programme. Elles peuvent principalement être de trois types : int (pour les entiers), float (pour les chiffres à virgule), char (pour les caractères). Les variables doivent être obligatoirement définies au début du bloc. [**10**]

Exemple :

int a, b ;

float c ;

Les formes d'écriture :

Les deux utiles sont :

-décimal (format par défaut)

-hexadécimal (il faut marquer 0x devant la valeur pour spécifier le type)

Les opérateurs :

Les opérateurs sont :

+ : addition

- : soustraction

* : multiplication

/ : division

= : égale

!= : différent

>,< : supérieur et inférieur

Il existe certaines subtilités sur les opérateurs. Ainsi mettre ++ ou – équivaut à, respectivement, ajouter ou soustraire '1' à la valeur ou variable. De même, marquer a=b, équivaut à dire que a est égal à b ; mais marquer a==b, consiste à comparer a à b, pour savoir s'ils sont égaux. Les opérateurs logiques : Ils servent à effectuer tous types d'opérations logiques. Il y a le ET (), le OU inclusif (||) et le NON (!).

Les opérateurs conditionnels

Ex : si X=Y alors A=B, sinon A=C

Se traduit par :

IF(X=Y)

A=B ;

ELSE

A=C ;

Ex : Surveille A ; si A=1, alors B=1 ; Si A=2, alors B=2…

Se traduit par :

SWITCH (A)

CASE '1' : B=1;

BREAK ;

CASE '2' : B=2;

BREAK ;

Ex : fait A=B*C, tant que A supérieur à 3

Se traduit par :

DO

A=B*C ;

WHILE(A>3) ;

Ex : La boucle FOR, étant difficile à traduire, nous allons la présenter, et expliquer son fonctionnement.
FOR(i=0 ;i<4 ;i++)

A=B*C ;

La boucle FOR présente trois paramètres p1, p2, et p3 : FOR(p1 ;p2 ;p3) p1 ne sert que lorsque l'on rentre dans la boucle. Il sert à initialiser des valeurs. P peut être vide s'il ne sert à rien. p2 est la condition qui permettra de mettre fin à la boucle. p3 est un paramètre où l'on effectue des opérations sur la variable de condition. Ces opérations peuvent être de différents types. La fonction FOR fonctionne ainsi. On rentre dans la boucle, le paramètre p1 est effectué. Puis, on vérifie p2. Si elle est fausse, on effectue les action entre incollades ; puis on effectue p3, et on retourne vérifier p2. Si p2 est vérifiée, alors, on sort de la boucle FOR, sinon, on recommence les actions, et p3, puis p2, jusqu'à ce que p2 soit vérifiée. Exemple

de programme pour résumer tout ce qu'on a vu :

```
/* Création d'un programme qui attribue une lettre à une variable selon le résultat d'une opération*/
//déclaration des librairies
include<stdio.h>
//programme principal
Void main(void)
//déclaration de variable
Int A, b, c=3 ;
Char car ;
//boucle FOR
For(b=0 ;b<3 ;b++)
A=b* c ; Swit ch(A)
Ca se '0' : car='A' ;
Break ;
Case '3' : car='B' ;
Break ;
Case '6' : car='C' ;
Break ;
```

Le C pour PIC :

Le C pour PIC est quelque peu différent de celui pour ordinateur. Ces différences viennent du compilateur C. En effet, puisque ce dernier fait la conversion C-assembleur et que les instructions dans ce dernier sont limitées (processeur RISC), le langage C est grandement simplifié, mais les concepteurs du compilateur ont légèrement modifié les instructions,qui sont heureusement assez intuitives, et rapidement assimilées. [**10**]

Les nouveautés :

La différence entre un ordinateur et un PIC est dans les E/S. De ce fait, en langage C PIC, on donnera, si on le souhaite et c'est préférable, des noms aux pattes, ou aux ports complets, entre l'appel des librairies et la déclaration de sous-fonctions et/ou main.

Les règles :

Il est nécessaire d'écrire les instructions de C en minuscule, et celles propres au PIC (comme les TRISA et TRISB) en majuscule.

L'appel des librairies :

En C PIC, les librairies s'appellent ainsi : include ''nom.H''

La déclaration des pattes et ports :

En C PIC, une patte se désigne par son port, et son numéro (lesquels commence à 0 jusqu'à 7). Par exemple, la patte 3 (en fait la quatrième) du port B sera : PORTB.3 Une patte seule se définit par la commande ''bit'' : bit fname@PORTB.3, affecte le nom fnam à la patte 3 du port B. Un port complet se définit par la commande ''char'' : char sname@PORTA, affecte le nom snam au port A. Ainsi, dans le main et les sous-programmes, on utilisera snam à la place de port A, et fnam, à la place de PORTB.3 [**10**]

Les affectations :

La première chose à faire dans le main, en C PIC, est de définir les valeurs dans TRISA et TRISB. Si leurs déclarations étaient compliquées en assembleur, elles sont grandement simples en C PIC. Ainsi il suffit de marquer : TRISA=0x00 (par exemple, en hexadécimal, la valeur étant à déterminer). On peut également rentrer la valeur en décimal (ex : TRISA=27), ou en binaire (ex :TRISA=0b00100110). En ce qui concerne, la façon dont on utilise les pattes, elle est très simple. Comme vu précédemment, elles se désigne par le port auxquelles elles appartiennent, et leur numéro de patte. De plus elles peuvent être affectées à une valeur logiques (en sortie, ex : PORTA.2=1 ;), ou être lues (en entrées, ex : c=PORTB.7). [**10**]

2.2.5 Exigences de programmation :

Programmer un PIC Programmer un microcontrôleur, c'est écrire le programme compiler dans la mémoire programme. 3 méthodes les plus répandus : Utiliser un programmateur simple. Utiliser un programmateur In-circuit. Utiliser un Boot loader et un bus de communication avec un ordinateur (USB, RS232,..).[**11**]

.Programmateurs simple :

Ces programmateurs à faible coût nécessite d'isoler le microcontrôleur de son circuit d'utilisation pour le programmer. Certains comme le PICSTART sont commercialisés par micro-chip, mais il est très facile de s'en fabriquer un avec peu de matériel. Attention : Les programmateurs maison ne fonctionneront pas tous directement sous MPLAB. Il est alors possible d'utiliser d'autres logiciels comme ICPROG. Anthony DUVAL.[**11**]

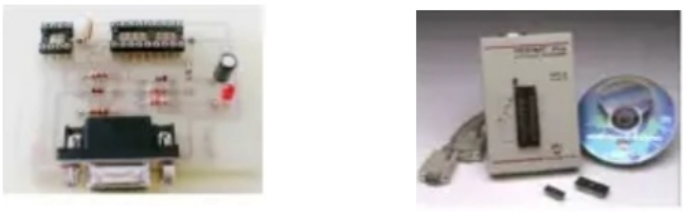


FIG. 2.1 : Programmateurs simple

[**11**]

.Programmateurs In-circuit :

Ce types de programmateurs permet de programmer le PIC directement dans son élément (carte final d'utilisation). Il évite ainsi les manipulations répétées qui risquent d'endommager le microcontrôleur : Décharges électrostatiques et casses des pattes de connexion. Ils permettent également de programmer les micros soudés en surface (CMS ou SMD) qui ne sont pas démontables Ces programmateurs permettent souvent le débogage de programme en situation réelle.

FIG. 2.2 : Programmateurs In-circuit

[**11**]

.Programme Notion de temps :

A droite, un exemple de déroulement d'un programme simple permettant de faire clignoter une LED la seconde. Un microcontrôleur ne peut effectuer qu'une seule instruction en même temps. Les sous

programmes de temporisation sont composés de suffisa mment d'instructions qui ne réalise aucune action pour faire « perdre du temps » au processus et générer ainsi une attente. Pendant la durée d'exécution d'un tel sous -programme, aucune autres actions ou programmes ne peut être effectués. Si Tcy=2us, 500000 instructions auraient pu être réalisés pendant chaque tempo.

Fig. 2.3 : Programmateurs Notion de temps

[**11**]

Programme : Notion de temps :

Regardons maintenant un autre exemple :

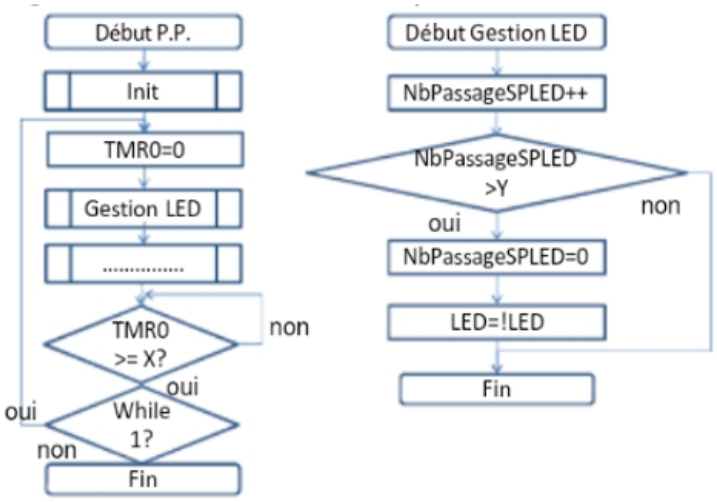


Fig. 2.4 : Exemple Programmateurs Notion de temps

[**11**]

. Utilisation d'un Boot loader :

Un boot loader est un petit programme qui est déjà implanter dans le microcontrôleur. Celui-ci permet de gérer une communication avec un ordinateur et d'écraser une partie de la mémoire programme par ce qu'il reçoit. On utilise alors une liaison série (RS232 ou USB) pour envoyer le programme. L'envoi du programme dans le PIC se fait cette fois encore In-Circuit. Inconvénients : le boot loader doit être implanté dans le PIC par l'intermédiaire d'un programmateur. Cette opération est néanmoins à ne faire qu'une seule fois. Tous les microcontrôleurs ne peuvent pas fonctionner avec un Boot loader. Il doit être capable d'auto écrire dans leur mémoire programme (Self- write) ou bien de lancer un programme depuis une mémoire externe.[**11**]

2.3 Conclusion

Dans ce chapitre, nous avons traité des langages de programmation les plus importants utilisés pour la programmation des microcontrôleurs PIC, car le langage d'assemblage est principalement utilisé pour améliorer les fonctions individuellement ou les algorithmes qui nécessitent des calculs intensifs, contrairement au langage C, qui est presque parfait en termes de la taille de la mémoire et la vitesse d'exécution du programme.

Chapitre 3

Description et rédaction du programme

3.1 Introduction

Tout au long de ce chapitre, nous parlerons d'une étude générale de PIC16F84 et également des programmes de programmation, au fur et à mesure de l'introduction du programme ISIS, ainsi que du programme LOGIPIC qui est utilisé pour préparer les algorithmes et les relier au programme de conversion en langage machine.

3.2 Etude de PIC16F84A :

3.2.1 Caractéristiques du PIC 16F84

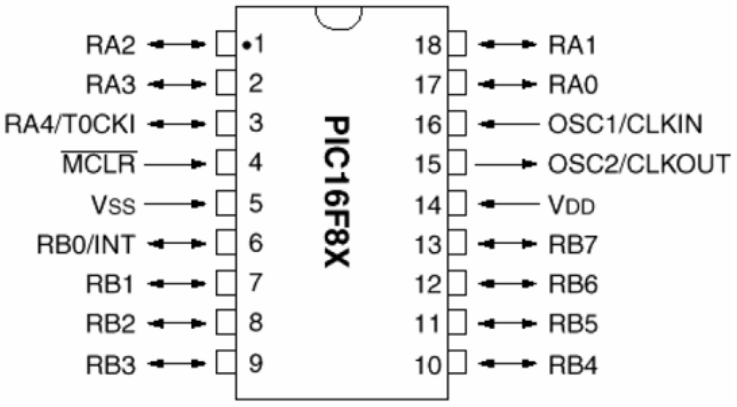


Fig. 3.1 : Architecteur PIC16F84A

[**11**]

Ce microcontrôleur, que nous surnommerons « le pic » pour des raisons de commodité, dans ce livre, possède 13 broches configurables, réparties sur deux ports : le port A et le port B. Le port A possède 5 broches (nommées RA1 à RA4), mais la quatrième, également appelée TOCKI peut servir pour une éventuelle temporisation externe. Le port B, lui, possède 8 broches (de RB0 à RB7) ; mais la broche RB0 peut également servir comme interruption éventuelle (un peu comme un garde sur un évènement) La broche 4, le MCLR barre, sert à indiquer au PIC s'il est en fonctionnement normal (un '1' logique) ou alors s'il est en cours de programmation (un '0' logique). Cette broche sert également à un éventuel Reset du PIC. Ne reste que 4 broches : l'alimentation (la 5 (0 volts) et la 14(+5 volts)) l'oscillateur(pattes 16 et 15) Outre ces caractéristiques, il faut savoir que le PIC 16F84 n'utilise pas de signaux analogiques, mais uniquement numériques (logique). De plus, il faut savoir, qu'il ne fait pas la différence entre les niveaux logiques TTL et CMOS. Il est donc alors préférable d'utiliser les deux extrêmes de signaux logiques TTL (0 et 5 volts), pour éviter tout problème possible. Voici enfin, les caractéristiques du PIC 16F84 fournit par Microchip :

.Mémoire de programme : 1KO, type Flash

.Mémoire de données RAM : 68 octets

.Mémoire de données E²PROM : 64 octets
.Niveau de la pile : 8
.Jeux d'instruction RISC : 35 de 14 bits
.Temps d'exécution des instructions normales : 4*Tosc
.Temps d'exécution des instructions de saut : 8*Tosc
.Cause d'interruption : 4
.Fréquence max de travail : 10 MHz
.Lignes E/S numérique : 13
.Temporisateur : un pour l'utilisateur, un pour le Watchdog
.Tension d'alimentation : 2 à 6 V continu
.Tension de programmation : 12 à 14 V continu
.Boîtier : DIL 18

Enfin, autre point fort du PIC : la consommation, car en tant que système embarqué, il faut que ce microcontrôleur consomme peu. Ainsi, des 2 mA de consommation en fonctionnement normal, il peut passer à 10 A en fonction veille ou sommeil, comme par exemple sur un téléphone portable, quand on ne s'en sert pas, la lumière reste éteinte : il est en veille. En revanche, au moindre appui sur un bouton (interruption externe, broche RB0 ici), il se remet en marche. Le Pic fonctionne de la même manière.[**11**]

3.2.2 Fonctionnement du PIC 16F84:

Dans cette partie du livre, nous traiterons des principaux systèmes du PIC, plus ou moins utiles. Nous illustrerons ces systèmes par des schémas, les plus clairs et les plus compréhensibles possible. Certains de ces systèmes, étant des paramètres configurables, devront faire l'objet d'attention selon que vous deciderez où non de les mettre en oeuvre. Nous verrons donc :
.les entrées/sorties
.les différents types d'oscillateurs possibles (pour l'horloge)
.le Reset
.la mémoire E²PROM
.la mémoire flash, utilisée dans le PIC
.les interruptions
.le TMR0
.le Watchdog
.le RTCC[**11**]

3.3 Présentation du logiciel ISIS :

Isis est un éditeur de schémas qui intègre un simulateur analogique / logique ou mixte. Toutes les opérations se passent dans cet environnement, aussi bien la configuration des différentes sources que le placement des sondes et le tracé des courbes. Dans cette section nous allons commencer par la présentation

de la fenêtre du logiciel ISIS[**7**]

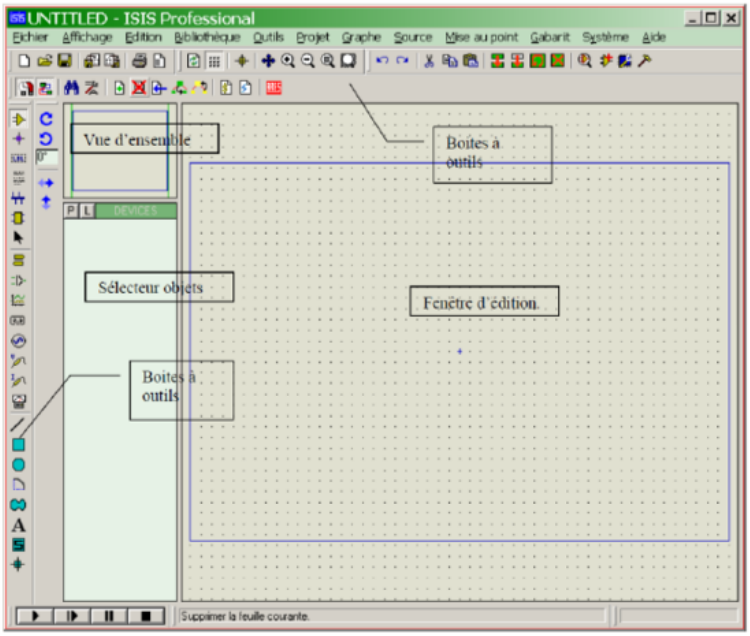


FIG. 3.2 : Interface du logiciel ISIS

[**7**]

3.3.1 Fenêtre d'ensemble (Vue d'ensemble) :

Le cadre en bleu délimite l'espace de travail tel qu'il a été défini par la commande 'Définir taille des feuilles' du menu 'système'. Le cadre en vert délimite La zone de travail, c'est à dire la partie du schéma visible dans la fenêtre principale. Vous pouvez déplacer cette zone de travail en pointant la souris sur la zone désirée de la fenêtre d'ensemble et en effectuant un clic gauche. Vous pouvez redéfinir la zone de travail dans la fenêtre d'ensemble en appuyant sur la touche majuscule 'shift 'du clavier, associée au déplacement de la souris en maintenant appuyé le bouton gauche.

3.3.2 Fenêtre d'édition :

La surface la plus grande de l'écran s'appelle "Fenêtre d'édition" et se comporte comme une fenêtre de dessin. C'est là que vous placez et câblez les composants.

3.3.3 La boite à outils :

Elle est composée d'un ensemble d'icônes dont les fonctions seront détaillées ultérieurement et d'un sélecteur d'objet utilisé pour choisir les boîtiers, le style des pastilles, des traces, des traversées, etc.[**7**]

3.3.4 Organisation de la boite à outils :

FIG. 3.3 : Organisation de la boite à outils ISIS

[7]

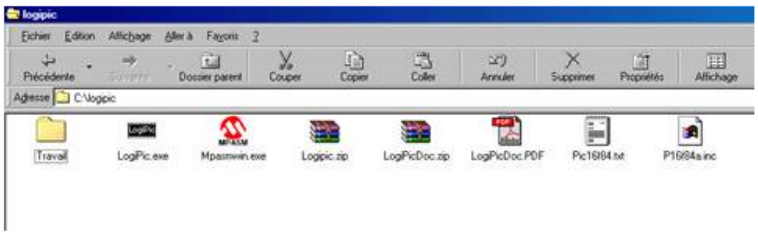
3.4 Présentation de l'ensemble des LOGIPIC :

C'est un programme qui nous permet de réserver l'algorithme et de le traduire en langage assembleur, pour obtenir le programme cible en langage machine nous avons besoin de MPASMWIN qui permet de transformer le langage assembleur en format compatible avec le microprocesseur.[**12**]

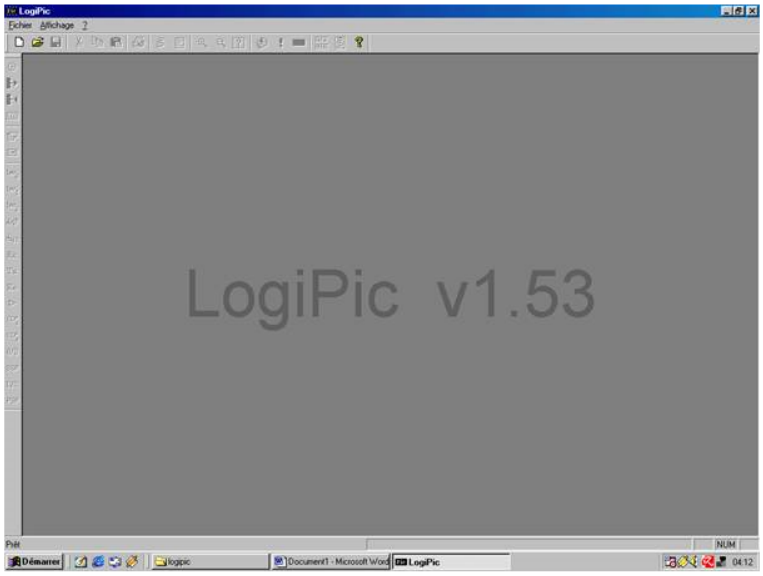
3.5 Tutorial LOGIPIC :

1-Copier les fichiers ci-dessous dans un dossier de votre choix (de préférence Logipic)

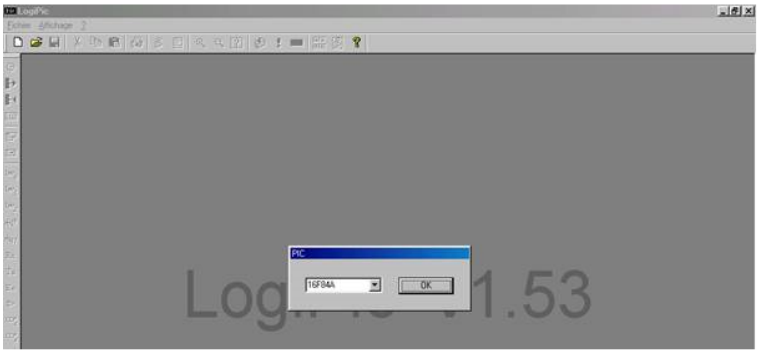
2-Créer dans Logipic un dossier de travail (Travail).



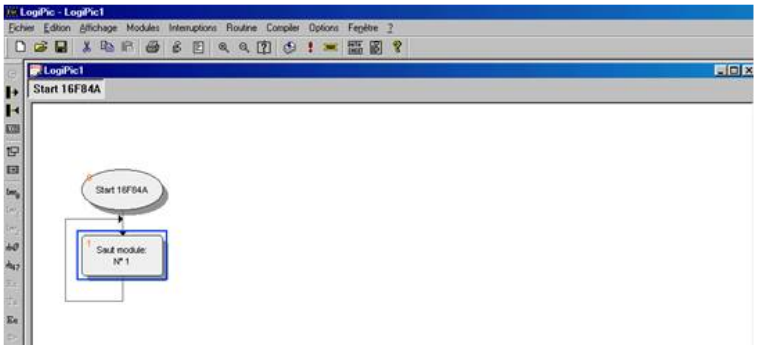
3-Lancer « logipic ». L'écran suivant apparaît.



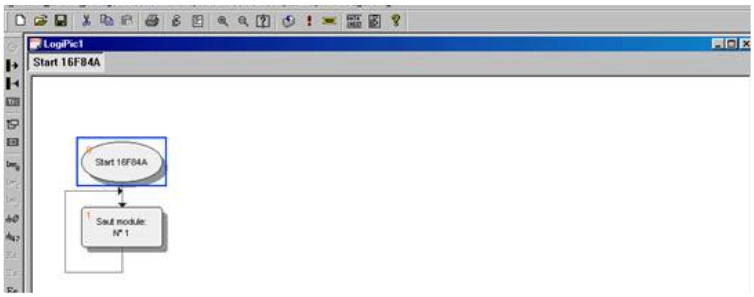
4-Pour créer un nouveau fichier : Fichier à Nouveau



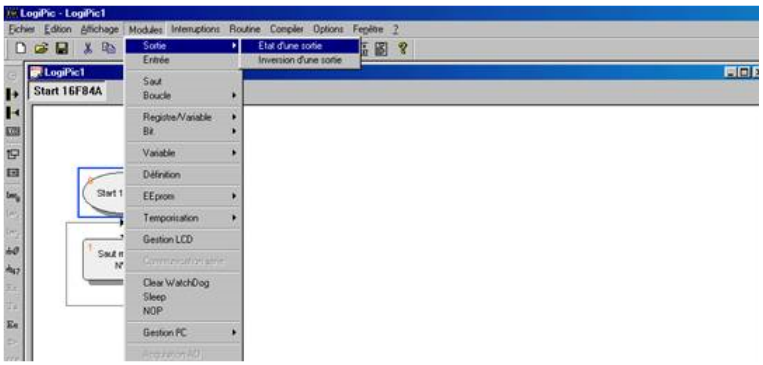
Choisir le type du pic puis validez par OK. On obtient :



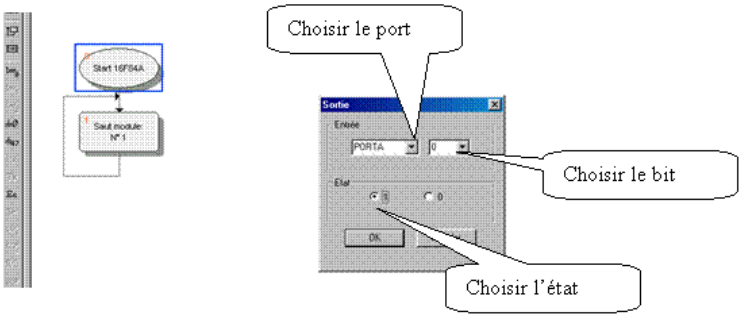
5-On se propose maintenant d'éditer un organigramme permettant de faire clignoter une branche sur la broche 0 du port A. On va appeler le fichier « led$_c lin$».$Commencerpar Apointerlebloc$«$Start16F84A$».



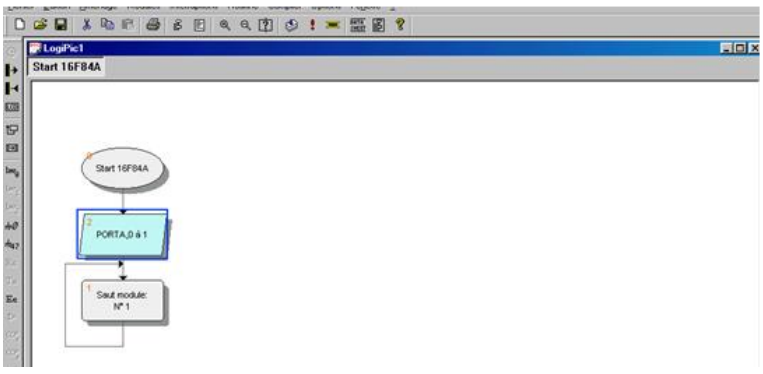
6-Insérer un bloc permettant d'allumer la led : Modules à Sortie à Etat d'une sortie



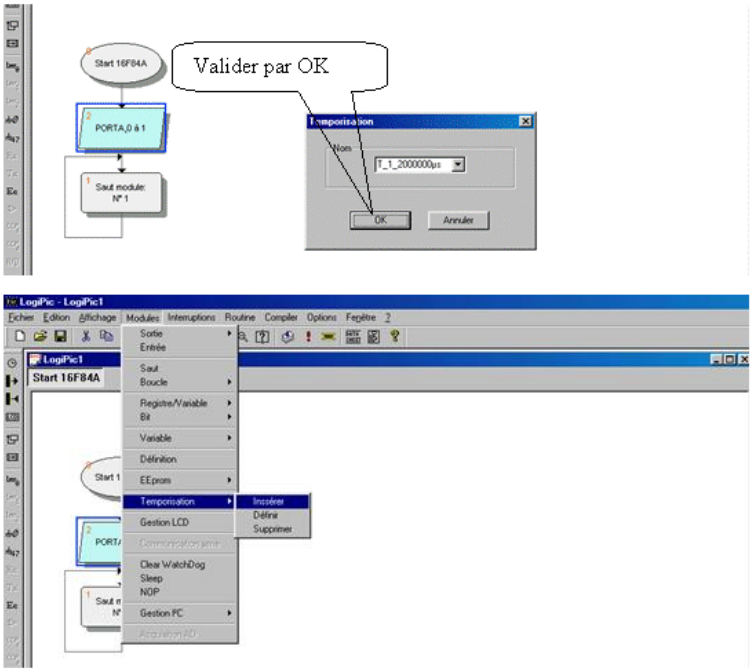
Une le choix est fait, valider par OK.



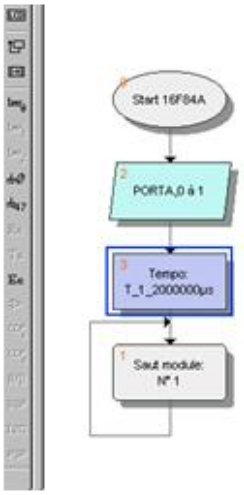
7-Définir un bloc de temporisation de 2 s (par exemple) : Modules à Temporisation à Définir



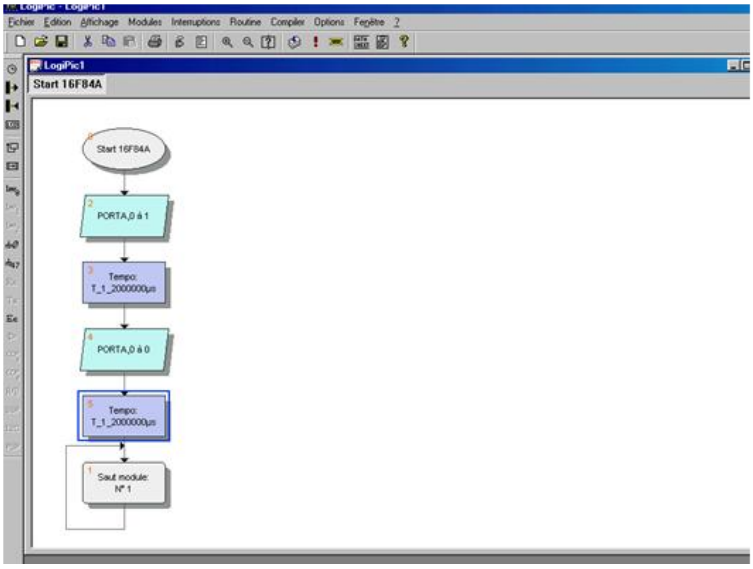
8-Insérer la temporisation : Modules à Temporisation à Insérer



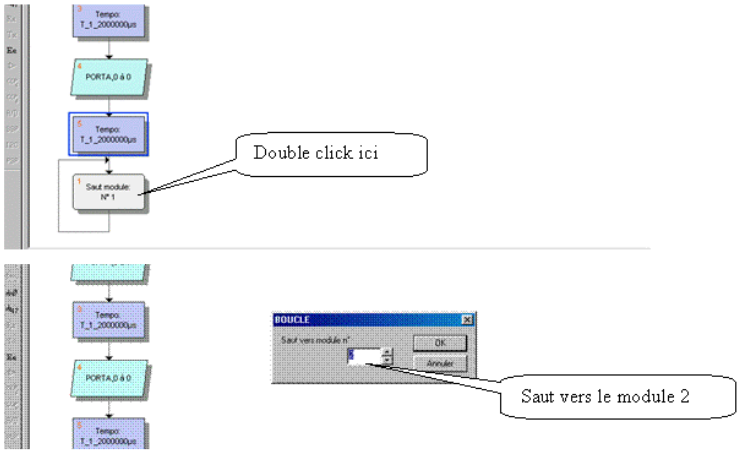
Le bloc de temporisation apparaît :



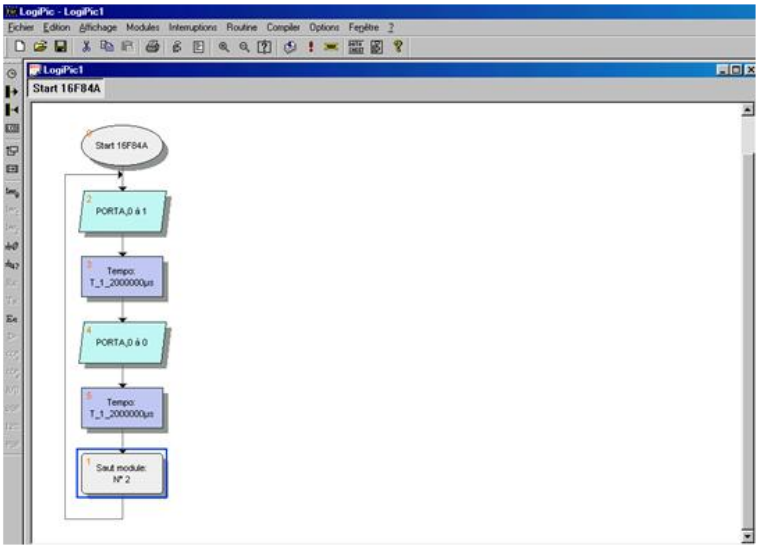
9-Procéder de la même façon pour insérer le module permettant d'éteindre la led (mise à 0 du bit 0 du port A), suivi d'une temporisation T_1.



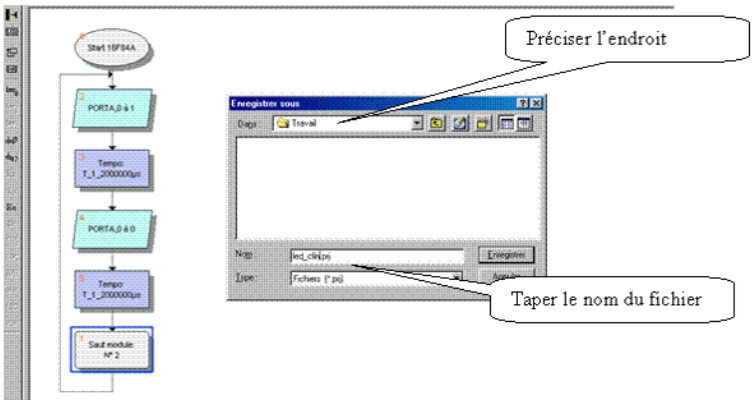
10-Modifier le branchement du saut par double click



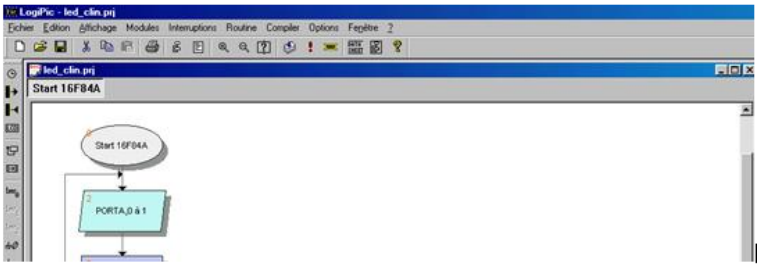
En validant par OK, on obtient l'organigramme final suivant :



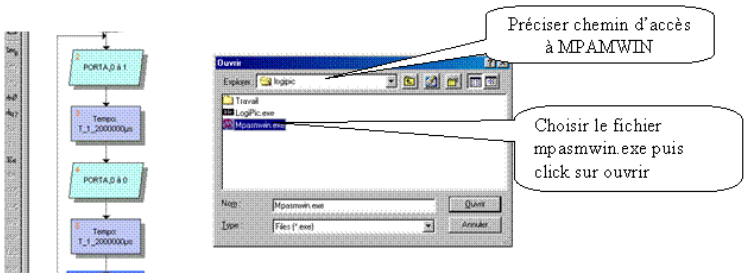
11- Enregistrement : Fichier à Enregistrer sous



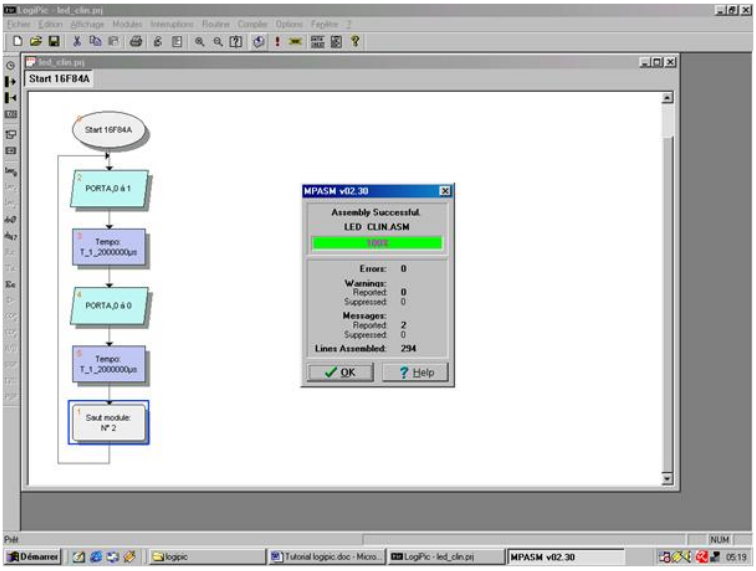
le nom du fichier en haut



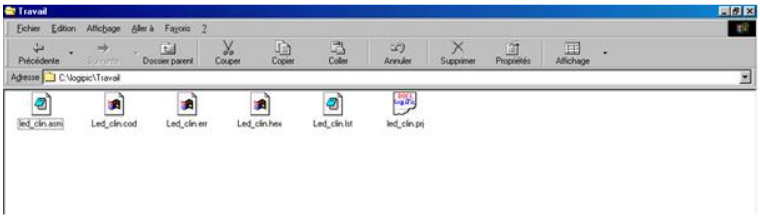
12- Configuration pour la compilation : Options à Chemins d'accès à « MPASMWIN »



13- Compilation : Compiler à Compiler



Si la compilation est réalisée, la bande 100s'affiche en vert et plusieurs fichiers sont ajoutés dans le dossier Travail parmi lesquels le fichier led clin.hex à transférer au microcontrôleur.[**13**]



3.6 Conclusion

Dans ce chapitre, nous avons expliqué les deux programmes LOGIPIC et ISIS utilisés dans la réalisation de la simulation d'irrigation intelligente, ainsi qu'une description du PIC16F84A et de ses accessoires, en plus de son principe de fonctionnement.

Chapitre 4

simulation de programme

4.1 Introduction

D'après ce que nous avons vu dans les chapitres précédents sur l'étude théorique des microcontrôleurs et des logiciels de programmation.

Dans cette partie, nous illustrerons les étapes de réalisation du système d'irrigation intelligent qui permet le fonctionnement de la pompe à eau et le contrôle du capteur d'humidité et de l'exposition au soleil, car tout cela sera contrôlé en utilisant (PIC16F84A), et en utilisant le Programme de simulation (ISIS).

4.2 Matériels utilisé :

4.2.1 Capteur d'humidité :

.Principe de fonctionnement :

La conductivité électrique de la terre dépend de l'humidité du sol, autrement dit la résistance électrique d'un sol augmente avec la sécheresse de celui-ci. Pour déterminer la concentration d'eau dans le sol, on exploite la propriété ci-dessus. Ainsi, pour mesurer cette résistance électrique on utilise deux électrodes qui sont fixées sur un support en forme de fourche que l'on plante verticalement dans le sol.

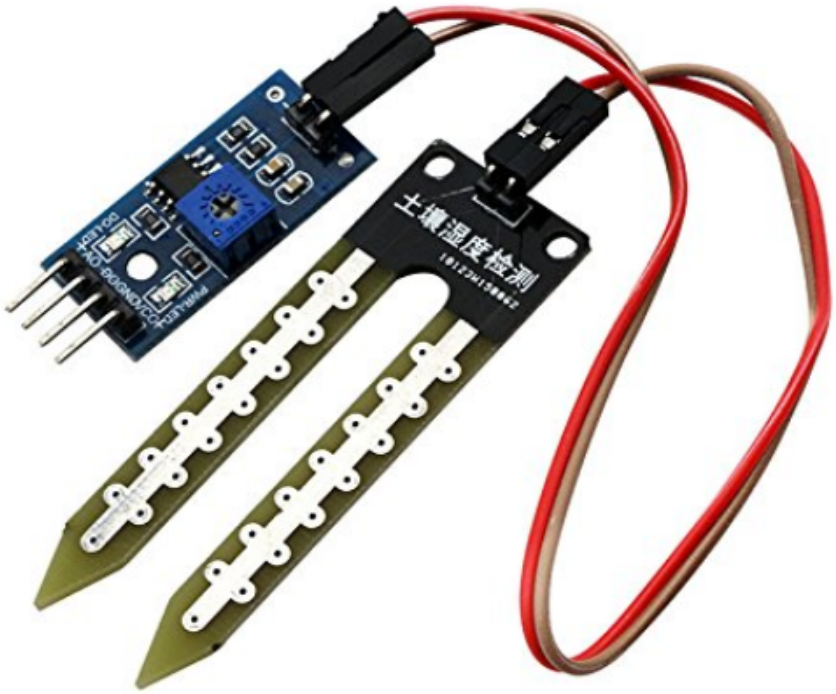


FIG. 4.1 : Capteur d'humidité

[**14**]

4.2.2 Capteur du solaire :

est un instrument de navigation qui équipe les plateformes de véhicules spatiaux tels que les satellites artificiels et sondes spatiales. Il permet, suivant les modèles, de détecter la présence ou la position du Soleil.

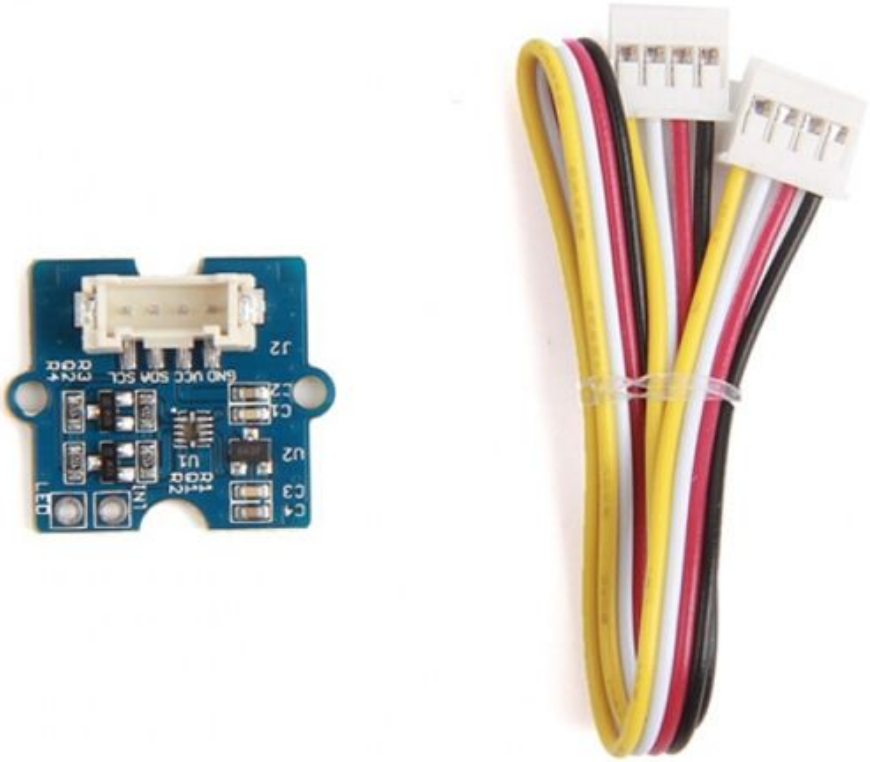


FIG. 4.2 : Capteur du solaire

[**15**]

4.2.3 La pompe :

Les pompes véhiculant des liquides se divisent en deux catégories principales :

.Les pompes centrifuges : le mouvement du liquide résulte de l'accroissement d'énergie qui lui est communiqué par la force centrifuge.

.Les pompes volumétriques : l'écoulement résulte de la variation d'une capacité occupée par le liquide.

FIG. 4.3 : La pompe

[**16**]

4.2.4 le microcontrôleurs PIC16F84:

Une carte PIC16F84 est une petite carte électronique équipée d'un microcontrôleur. Le microcontrôleur permet de programmer et commander des actionneurs à partir d'événement détectés par des capteurs.

FIG. 4.4 : PIC16F84
[**17**]

4.3 Matériels Unité de commande :

La salle de contrôle se compose de :
1- Appuyer sur les boutons pour contrôler (les portes d'exposition au soleil ainsi que la ventilation).
2- Témoin lumineux l'intention de contrôler (le capteur d'humidité et la source d'alimentation (jour / nuit))

4.4 Organigramme que représente la réalisation :

4.4.1 Principe Fonctionnement :

Le système étudié composé de plus de 50 serres et un système hybride assuré l'alimentation électrique du système, pour faciliter l'étude il sera fait sur une seule serre.

Chaque serre comporte par des éléments suivants :

- Une pompe assurer l'arrosage

- des Fenêtres d'aération pilotée par un vérin (vérin A)

- Des portes glissent (montée au plafond) commander par un vérin assuré l'exposition de soleil (vérin B)

- Capteur de l'humidité Le système comporte aussi par un capteur de lumière afin de faire le choix entre le système solaire ou le système conventionnel qui ont assurant la production d'électricité.

4.4.2 Organigramme de notre système :

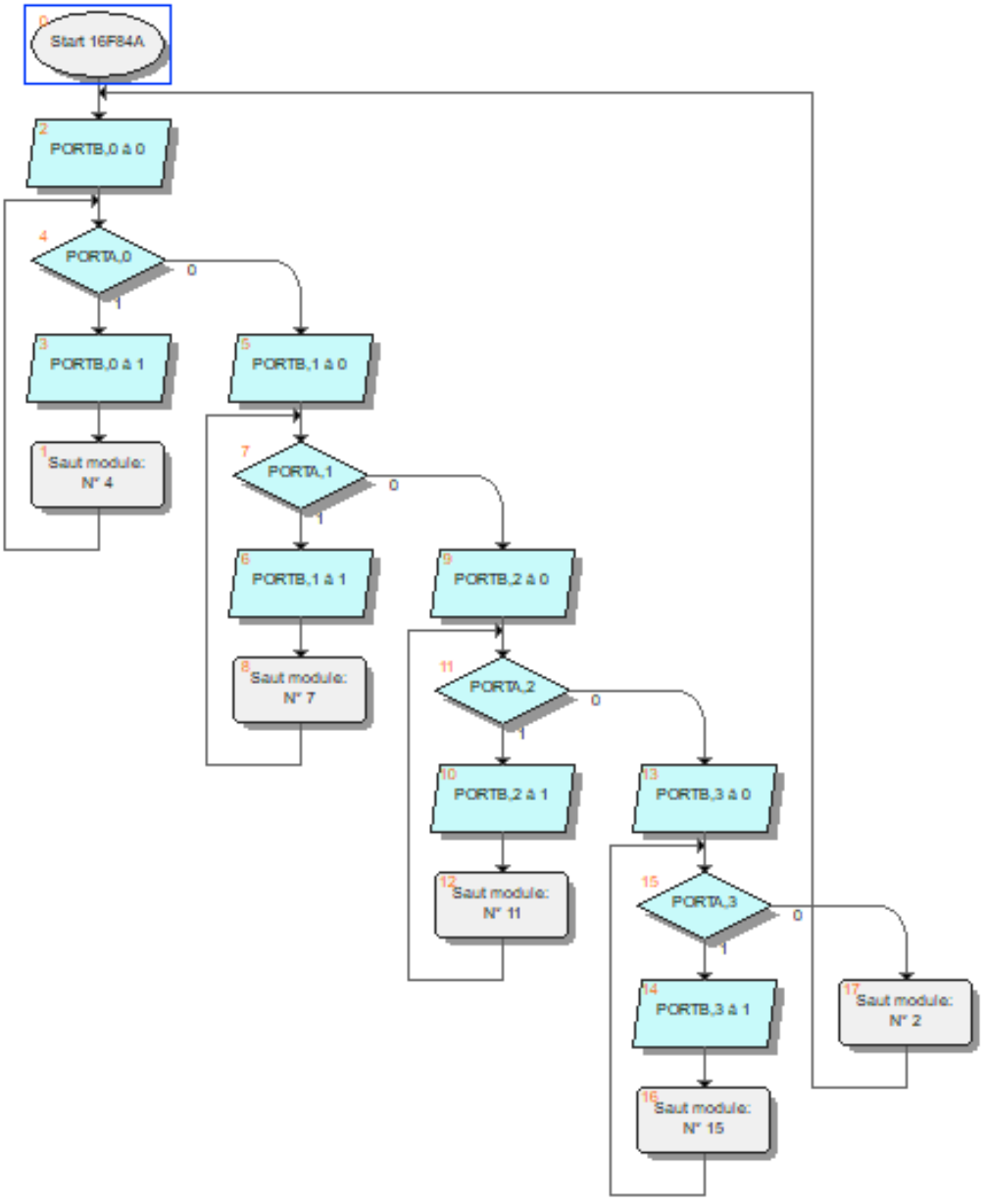


FIG. 4.5 : L'organigramme général de notre système

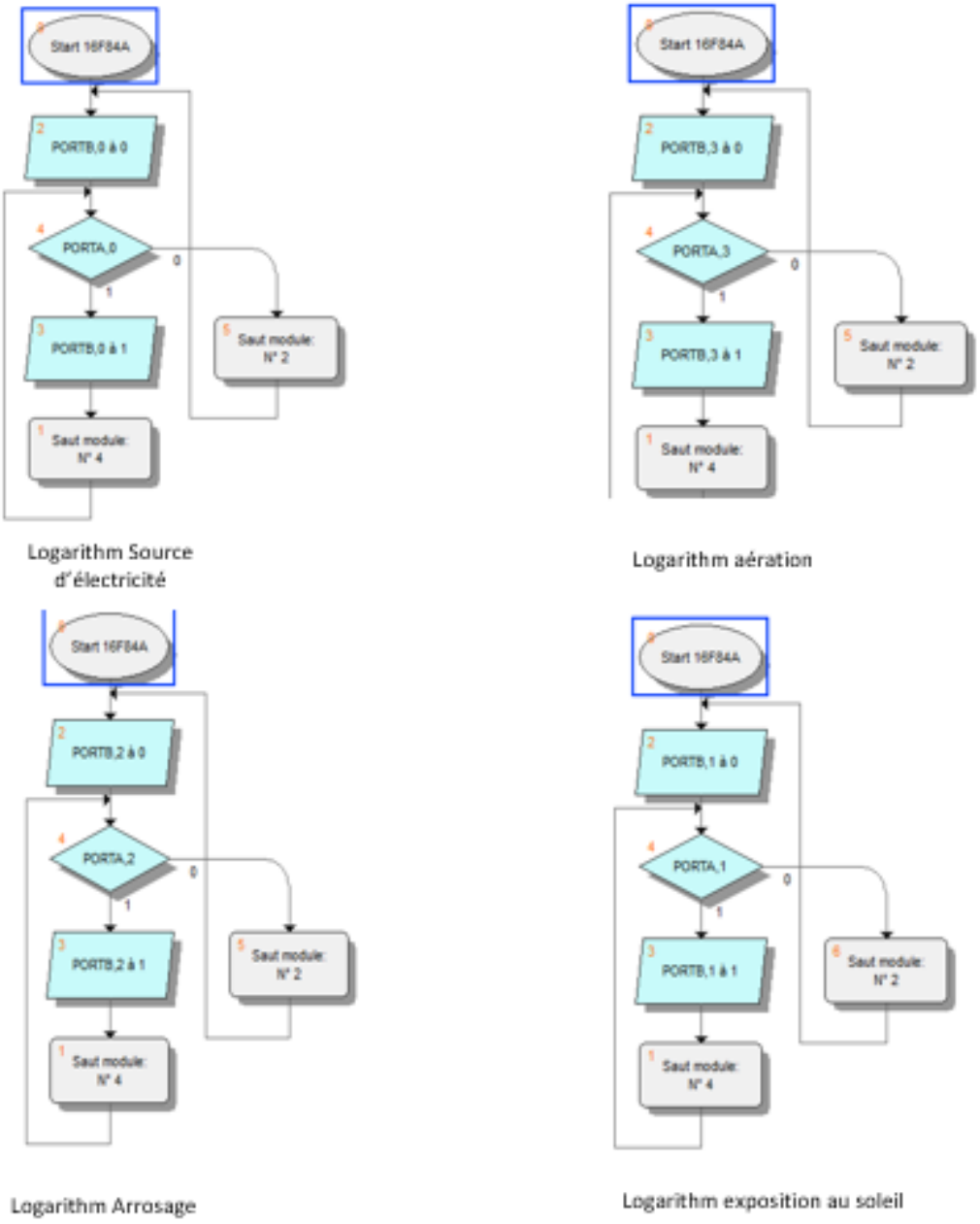


FIG. 4.6 : L'organigramme de chaque tâche de notre système

4.4.3 Schéma de la simulation :

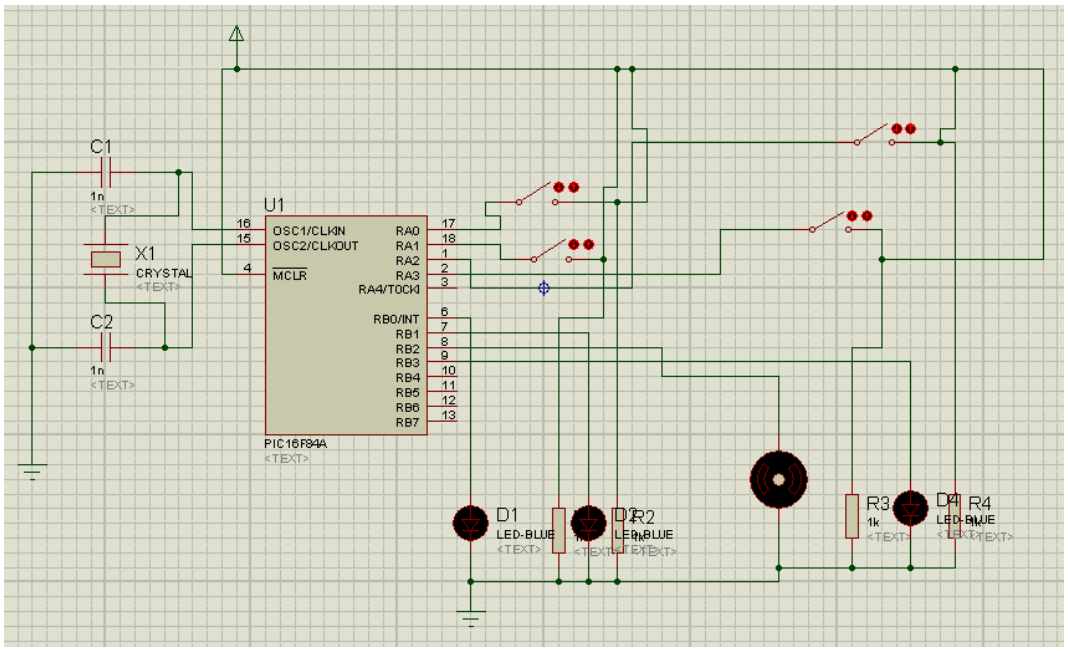


FIG. 4.7 : schéma général sur ISIS PROTEUS

Après avoir incorporé l'algorithme dans LOGIPIC et l'avoir inséré dans une simulation dans ISIS, nous obtenons les résultats suivants :

Tache 1:

source d'électricité (jour/nuit)

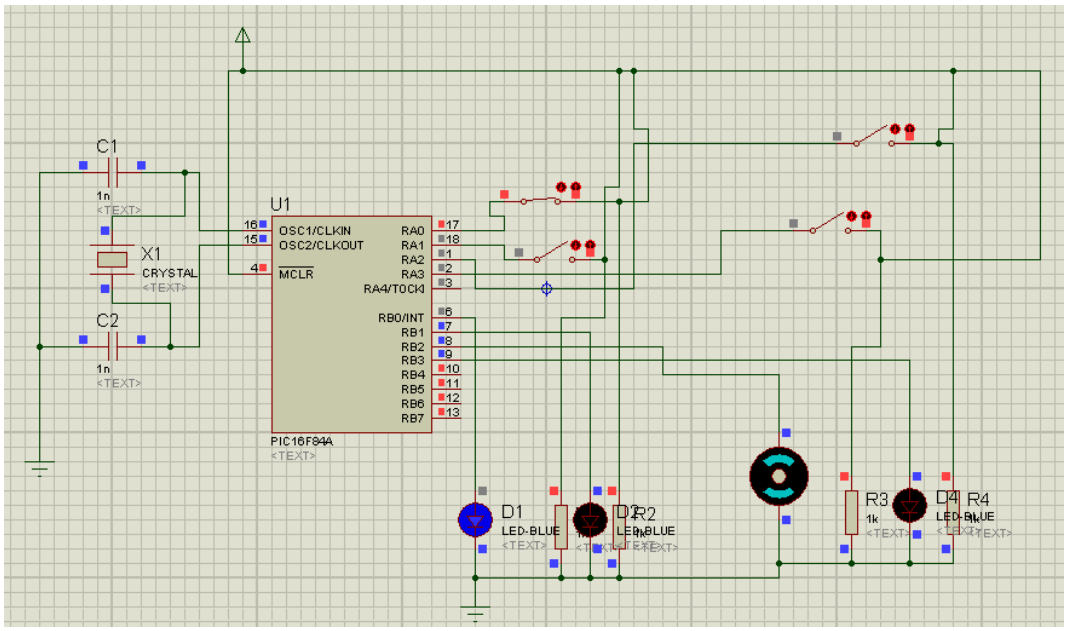


FIG. 4.8 : schéma Tache 1 sur ISIS PROTEUS

Lorsque le disjoncteur connecté à l'entrée RA0, qui est commandé par le compresseur en salle de commande, est fermé, le capteur optique détecte l'état du soleil (présence/absence). S'il fait jour, la source d'énergie proviendra du générateur, mais dans le cas de la nuit, la source d'énergie dépendra de l'énergie stockée des rayons du soleil.

Tache 2:

exposition au soleil

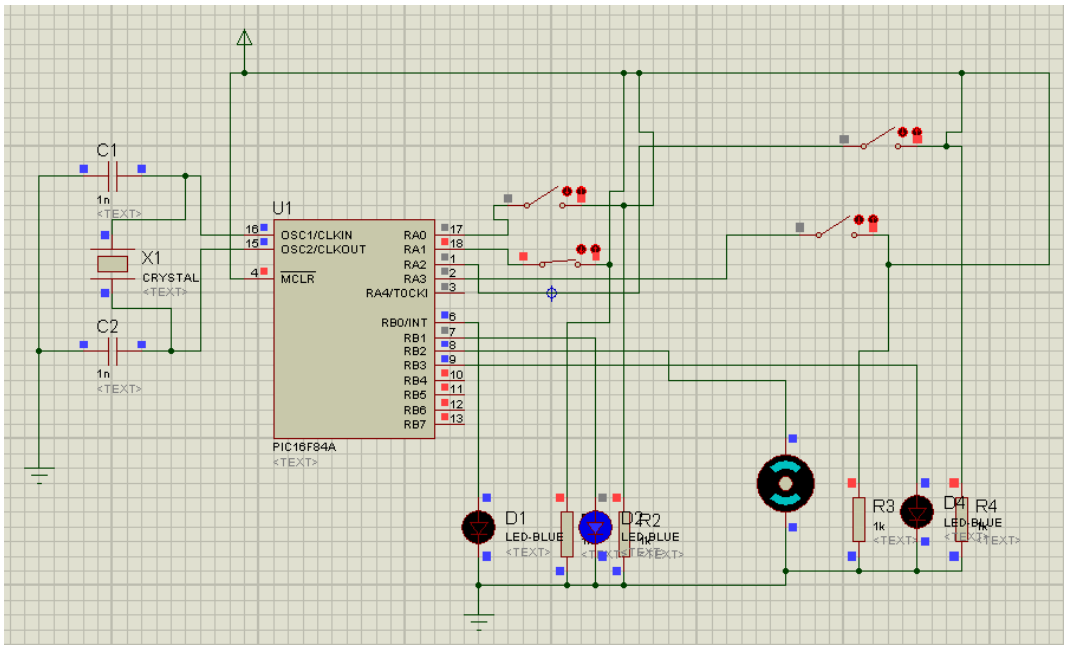


Fig. 4.9 : schéma Tache 2 sur ISIS PROTEUS

Pour assurer le besoin d'ensoleillement de l'usine, on appuie sur le compresseur de la salle de contrôle relié à l'entrée RA1 pour ouvrir les portes, et pour les fermer, le même compresseur est appliqué.

Tache 3:

arrosage

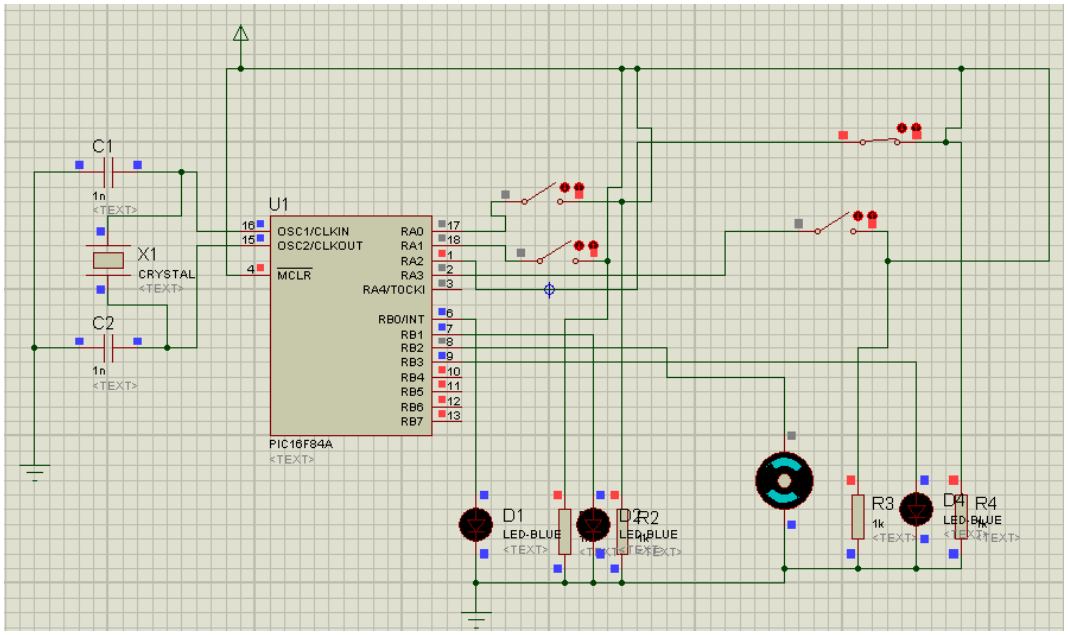


Fig. 4.10 : schéma Tache 3 sur ISIS PROTEUS

Pour arroser la plante, le capteur d'humidité détecte l'état du sol (sécheresse) et donne l'ordre de faire

fonctionner la pompe à eau pour alimenter la plante en eau en fermant le coupeur connecté à l'entrée RA2, et après que le sol soit saturé d'eau , la pompe à eau s'arrête automatiquement en ouvrant le cutte.

Tache 4:

aération

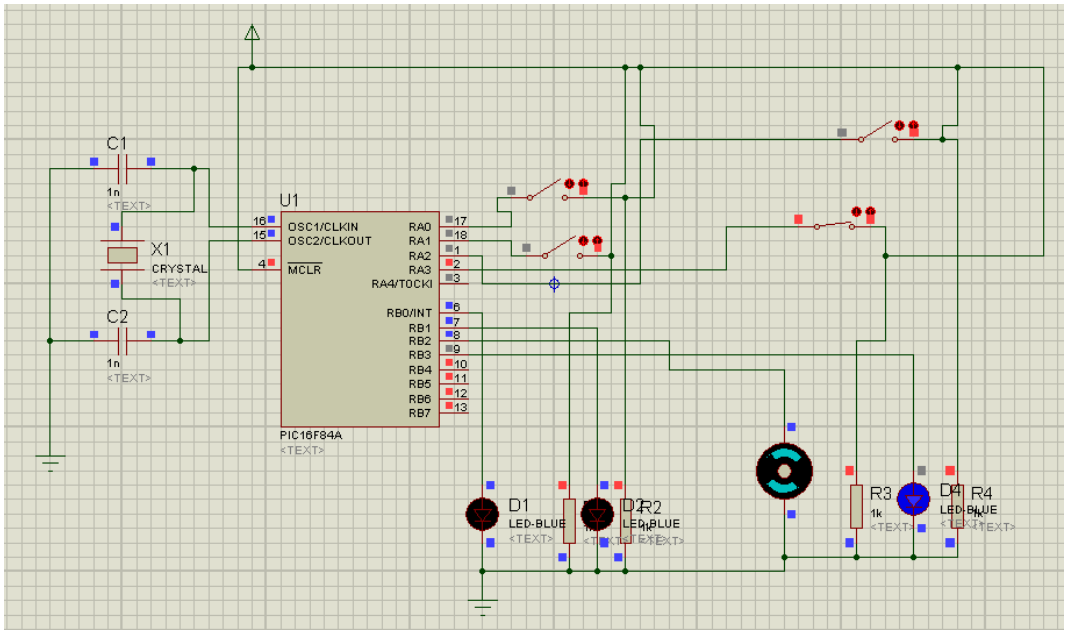


FIG. 4.11 : schéma Tache 4 sur ISIS PROTEUS

Pour ventiler, on appuie sur le compresseur dans la salle de contrôle, qui à son tour ferme le disjoncteur connecté à l'entrée RA3, où les fenêtres de ventilation sont ouvertes et pour les fermer, le même compresseur est à nouveau enfoncé.

4.5 Conclusion

A travers ce chapitre, nous avons brièvement défini les différents composants utilisés dans le système, et réalisé une simulation de notre système d'irrigation, qui comprend l'ouverture et la fermeture des fenêtres (ventilation), l'ouverture et la fermeture des portes (exposition des plantes au soleil) et la source d'alimentation par le programme ISIS et par le microcontrôleur PIC16F84A qui répond aux algorithmes Il a été bien programmé et a obtenu d'excellents résultats.

Conclusion générale

Conclusion générale

Notre travail représente une étude et une analyse d'un système d'irrigation intelligent qui effectue une auto-irrigation en mesurant l'humidité du sol, en ventilant les serres, en exposant la plante au soleil et en sélectionnant la source d'énergie appropriée en fonction de la situation temporelle (nuit / jour), et ce projet est implémenté par PIC16F84A.

Au début, nous avons lu quelques livres sur le microcontrôleur et le programme ISIS, et à travers eux, nous avons introduit une définition complète du microcontrôleur et de certains de ses éléments, de sa structure et de ses caractéristiques les plus importantes, et dans la deuxième étape, nous nous sommes penchés sur le PIC16F84A et son langage de programmation, et enfin nous avons simulé par ISIS et le programme LOGIPIC pour notre système d'irrigation et à partir des résultats obtenus à partir de la simulation logicielle et des tests pratiques, nous avons remarqué que le PIC16F84A est simple à utiliser, à programmer et à contrôler, et que le système d'irrigation est autonome car le capteur d'humidité contrôle la pompe à eau de manière coordonnée.

Ce projet était un défi pour nous, de pouvoir comprendre le processus de programmation de tels programmes, car nous ne les avions jamais abordés. Nous aurions pu rêver d'avoir des composantes selon le besoin attendu et matérialiser le projet, mais malgré le manque de moyens à ma disposition pour mettre en œuvre le projet sur le terrain, Mais la recherche a été formidable, malgré la présence de ces difficultés, nous avons toujours l'ambition d'incarner notre projet sur le terrain dans un futur proche.

Enfin, et à travers ce que nous avons atteint, nous suggérons des modifications simples, y compris l'ajout d'un capteur de température pour l'auto-exploitation du programme de ventilation, et l'ajout d'arroseurs pour fertiliser le sol qui fonctionnent automatiquement, et nous aspirons à l'avenir que le domaine de l'agriculture, de la cueillette des graines à la cueillette des fruits, être automatiquement sans intervention humaine, tout cela pourrait aider les chercheurs à l'avenir.

Bibliographiques

Bibliographe

[1] : Cours Microcontrôleurs- Auteur : Mme. Yosra Rkhissi Kammoun

[2] : Mebareke Abdennur et Kharroubi Larbi << Etude et Réalisation d'un cardio – fréquencemètre portable A l'aide du PIC16F84 >> Thèse représenter pour obtenir le diplôme de master en Génie Biomédical. Université de Tlemcen Année 2014-2015.

[3] : Jérôme VICENTE- Les_microcontrôleurs_Dpt_ME_Otion_SIIC_2_ème_année_2005_2006

[4] : les microcontrôleurs dans les systèmes embarqués

[5] : https://fr.wikipedia.org/wiki/Microcontr%C3%B4leur

[6] : Alibi El Mehdi/Jawadi Sami Rapport de projet de fin d'études Conception et réalisation d'un enregistreur de données -

(http://pfmh.uvt.rnu.tn/491/1/Conception_et_réalisation__d'un_enregistreur_de_données_.pd)

- 2010/2011 - 24/03/2013

[7] : Aberkane Mounir<< Mise en œuvre du protocole I2C Dans un environnement à microcontrôleur Micro chip (PIC16F877) >> Thèse représenter pour obtenir le diplôme de master en télécommunication. Université Mohamed Khider Biskra2012-2013.

[8] : OUDJEDAOUI HADJIRA << étude et réalisation d'un simulateur cardiaque >> Thèse représenté pour obtenir le diplôme de master en Génie Biomédical. Université de Tlemcen. Année2012-2013.

[9] : Cours 3A DEE MicroControleur.pdf

https://fr.scribd.com/document/217291951/Cours-3A-DEE-MicroControleur-pdf

[10] : Le PIC 16F84 L'essentiel - Alexandre GALODÉ
http://diablotronic.free.fr/Download_Knowledge/le_pic_16f84_livre.pdf

2004/11/11

[11] : Cours 3A DEE MicroControleur.pdf

https://fr.scribd.com/document/217291951/Cours-3A-DEE-MicroControleur-pdf

[12]: Microsoft PowerPoint - pic16F84

[13] : Tutorial logipic

https://www.technologuepro.com/microcontroleur/Tutorial_logipic.htm

[14] : Capteur d'humidité

https://www.4atoms.com/wp-content/uploads/2018/11/Soil-temperature-and-humidity-sensor.jpg

[15] : https://images.app.goo.gl/thYPo8pPygJCHuwj6

[16]:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.technipompe.fr%2Fpompes-de-surface-surpresseurs-c2.php&psig=AOvVaw0R-VuJJioFI2YDwzD75iXJ&ust=1622802230827000&source=images&cd=vfe&ved=0CAMQjB1qFwoTCOjWuZyS-_ACFQAAAAAdAAAAABAD

[17]:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.microchip.com%2Fpic16f84&psig=AOvVaw0fCkeoBsqjrETNbzbKEy1F&ust=1622803009236000&source=images&cd=vfe&ved=0CAMQjB1qFwoTCPifp4yV-_ACFQAAAAAdAAAAABAO

[18] : UNIVERSITE M'HAMED BOUGARA DE BOUMERDES FACULTE DES SCIENCES : V.TOURTCHINE ; MICROCONTROLEUR DE LA FAMILLE PIC Support de cours & Prise en main du logiciel MPLAB 2009