République Algérienne Démocratique et Populaire
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABOUBEKR BELKAID
- TLEMCEN -

FACULTE DE TECHNOLOGIE

DEPARTEMENT DE GENIE BIOMEDICALE

MEMOIRE DE PROJET DE FIN D'ETUDE

pour l'obtention du diplôme de

MASTER en GENIE BIOMEDICAL

Spécialité : instrumentation biomédicale.

Présenté par : SLIMANI HICHEM et NEKROUF MED SAID

ETUDE ET REALISATION D'UN SYSTEM POUR LA MESURER LA PERSISTANCE RETINIENNE A L'AIDE D'UN PIC16F84A

Soutenu le 24 mai 2015 devant le Jury

Mr.	DBALE. R	*MCB*	Université de Tlemcen	Président
Mr.	DIB. N	*MAA*	Université de Tlemcen	Examinateur
Mr.	HAMZA CHERIF.L	M.A.A	Université de Tlemcen	Encadreur

Année universitaire : 2014 -2015

Remerciements

Je remercie Dieu tout puissant de nous avoir donnés le courage et la patience pour mener à bien ce travail, qu'il soit béni et glorifié.

Nous tenons expressément et chaleureusement à remercier nos parents pour leur soutien et leurs encouragements tout au long de nos études.

Nous tenons à exprimer nos plus sincères remerciements à Mr. Hamza Cherif, notre encadreur. Nous le remercions pour ses précieux conseils; sa confiance, qui ont permis une progression concrète dans ce projet.

Nous tenons particulièrement à remercier ***Mr. Benmoulai hadj Mohamed*** *que nous remercions vivement de nous avoir honoré de diriger ce travail ; ses conseils et ses motivations ont été pour nous un précieux encouragement.*

Je tiens à exprimer mes plus sincères remerciements à mon Co cadreur Mr M'hamed saadi bachir.

Je remercie tout d'abord les membres du jury qui me font honneur en jugeant ce travail.

Nous tenons aussi à remercier tout le corps pédagogique ;enseignants ,administrateurs, Employés du département de génie biomédicale.

Dédicaces

Je dédie ce travail à mes chers parents pour m'avoir soigné par leur parrainage.

M*a grand-mère pour leur aide et leur soutien durant toute la période de mes études.*

A toute ma grande famille SLIMANI, en particulier mes oncles.

A tous ceux qui m'ont, de près ou de loin, aidé à élaborer ce travail

Enfin je le dédie à tous mes amis que je n'ai pas cités et à tous ceux qui me connaissent.

Hichem

Sommaire

Introduction générale

L'ophtalmologie est la branche de la médecine chargée du traitement des maladies de l'œil et de ses annexes. C'est une spécialité médico-chirurgicale.

La représentation visuelle que nous avons du monde extérieur est le fruit d'une construction cérébrale réalisée à partir des signaux lumineux captés par notre œil.

Comme tout organe du corps humain, l'œil a toujours étonné par sa richesse les scientifiques et biologistes. Sa fonction est de transformer l'information lumineuse en influx nerveux transmis au cerveau et n'est toujours pas parfaitement connu, même si on essaye de le mimer dans de nombreux projets technologiques modernes.

Toutefois, les récentes avancées technologiques, nous on permit de mieux comprendre certaines parties de son fonctionnement et de combattre ainsi quelques défauts pouvant apparaître chez cet organe comme certains troubles visuels.

L'électrophysiologie visuelle consiste à enregistrer les signaux bioélectriques émis par l'œil suite à une stimulation lumineuse. L'électrorétinogramme ou ERG, enregistre l'activité des photorécepteurs et des couches rétiniennes. Notre travail dans ce projet de fin d'étude, consiste à étudié et réalisé un système pour la mesure de la persistance rétinienne. Le principe de notre système est basé sur des stimulations temporisées à l'aide d'un microcontrôleur (16 f 84) pour déterminé la persistance rétinienne de nos yeux. L'utilisation des résultats obtenus qui présent différents aspects de la vision permet d'avoir des informations portant sur l'acuité visuelle.

A l'aide d'observation de l'organisation de zones cessibles « photorécepteurs » on peut avoir aussi des informations sur le trajet des messages nerveux issus de différentes zones de la rétine est les conséquences prévisible d'une section accidentelle d'un nerf optique.

Notre mémoire est répartie en quatre chapitres :

- Le premier chapitre présente une brève étude sur le système visuel et la persistance rétinienne.
- Les systèmes à base de microcontrôleur sont discutés dans le deuxième chapitre.
- Le troisième chapitre a été consacré à l'étude et la réalisation de notre système pour la mesure de la persistance rétinienne.
- Le dernier chapitre présente l'application pratique et les résultats obtenus.

Chapitre I :

Le système visuel et la persistance rétinienne

1.1. Introduction

Le système visuel nous donne des informations sur la position, la taille, la forme, la texture et la couleur des objets qui nous entourent. Les mouvements des objets avec leur direction et leur vitesse relative sont eux aussi détectés. Ces derniers sont enfin identifiés en fonction de l'expérience de l'individu. Un grand nombre de ces informations peuvent être obtenues aussi bien à la lumière des étoiles qu'en pleine lumière du jour.

Nous introduisant dans ce chapitre quelques notions sur l'anatomie de l'œil, le système visuel et la persistance rétinienne sujette de notre travail. Elle consiste à la capacité du système visuel à superposer une image déjà vue aux images que l'on est en train de voir. Elle résulte du temps de traitement biochimique des signaux optiques par la rétine et le cerveau. Elle est plus forte et plus longue si l'image observée est lumineuse.

1.2. Anatomie de l'œil humain [1]

L'œil est l'organe de la vision. Il est de faible volume (6.5 cm^3), pèse 7 grammes, et a la forme d'une sphère d'environ 24 mm de diamètre, complétée vers l'avant par une autre demi-sphère de 8 mm de rayon, la cornée (figure 1.1).

L'œil est composé de plusieurs éléments qui ont chacun une influence dans le cheminement de la lumière et la compréhension du signal optique par le cerveau. La figure suivante représente l'anatomie de l'œil humain.

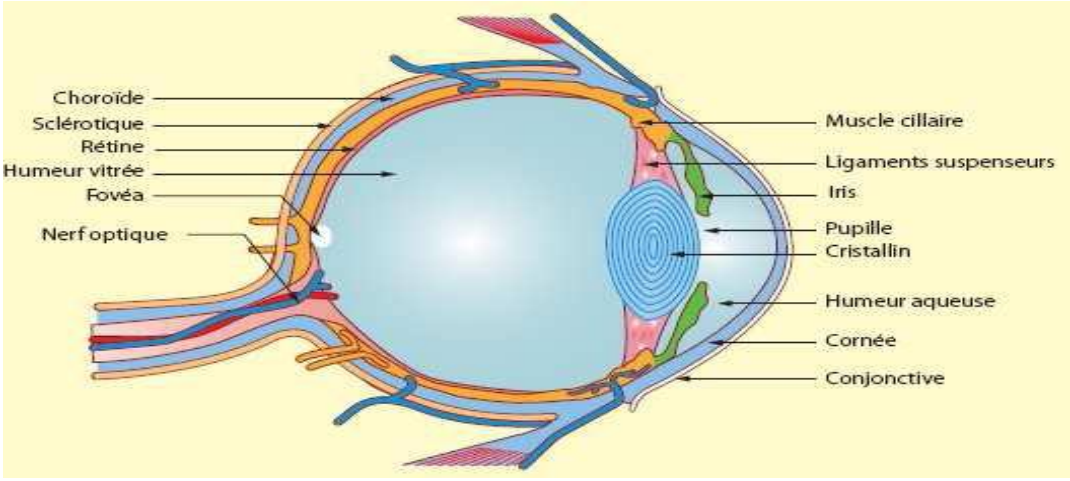

Figure 1.1 : Anatomie de l'œil humain.

L’œil humain est limité par trois enveloppes emboîtées : la sclérotique, la choroïde, et la **rétine** qui se prolonge par le **nerf optique**. Il comprend des **milieux transparents** qui sont traversés par les rayons lumineux avant d’atteindre le fond de l’œil : **cornée**, **humeur aqueuse, cristallin, humeur vitrée.**

a. La cornée

C’est une ouche transparente et résistante placée sur la face avant de l'œil. Son rôle est de protéger l’avant du globe oculaire. Sa courbure dépend des individus et varie aussi avec l'âge.

b. L'humeur aqueuse

C’est un liquide salin et alcalin sous pression qui maintient ainsi la rigidité du globe oculaire.
L'humeur aqueuse est un liquide transparent constamment renouvelé responsable du maintien de la pression intraoculaire. Elle est produite par les procès ciliaires et passe de la chambre postérieure vers la chambre antérieure à travers la pupille.
L’humeur aqueuse est composée de 99,6 % d’eau, mais aussi de vitamine C, de glucose, d’acide lactique, de Na et de Cl en majorité et elle est pauvre en protéines et en acides aminés. Elle se renouvelle constamment toute les 2 à 3 heures. [2]

c. L’iris

Membrane colorée qui fonctionne comme un diaphragme en contrôlant la quantité de lumière qui pénètre dans l'œil. Couleur donnée par un pigment, la mélanine. L’iris est également capable, en s’ouvrant et en se fermant, de régler la quantité de lumière entrant dans l’œil « le réflexe pupillaire » figure 1.2 :

➢Lorsque la luminosité est faible, l’iris s’ouvre pour laisser rentrer un maximum de lumière.

➢Lorsque la luminosité est élevée, l’iris se ferme pour éviter les éblouissements.

a)  **b)** 

Figure 1.2 : Le réflexe pupillaire : a) En lumière faible, b) En lumière forte.

d. La pupille

Elle a une apparence de disque noir, est l'ouverture centrale de l'iris. Sa taille varie pour laisser passer plus ou moins de lumière (diamètre=2mm en pleine lumière, 8mm dans l'obscurité). La dimension maximale de la pupille est affectée par le vieillissement. Les pupilles d'une personne âgée s'ouvrent moins que celles d'un jeune (en vieillissant on à besoin d'un éclairage plus intense pour être à l'aise).

e. Le cristallin

C'est une Lentille transparente convergente placée derrière l'iris. Il permet d'avoir une vision nette de ce que nous observons grâce à sa capacité à modifier sa courbure (mise au point). La cornée et le cristallin font converger la lumière qui entre dans l'œil en la focalisant sur la rétine (membrane recouverte de photorécepteurs). La perte de transparence des cellules cristallines, entraînant leur mort, s'appelle la cataracte.

f. L'humeur vitrée

Il constitue les 4/5 du volume de l'œil. Il est composé d'un liquide parfaitement transparent continuellement sécrété et absorbé, dont le rôle est d'assurer la structure autonome de l'œil.

g. Le nerf optique

La transmission des informations vers le cerveau est opérée par le nerf optique. Toutes les fibres optiques issues des cellules visuelles convergent vers un point précis de la rétine : **la papille**. Ce point ne contient donc pas de cellules visuelles mais seulement les fibres nerveuses. La papille est donc un point de l'œil qui ne voit pas. On l'appelle aussi **la tache aveugle**. En ce point débouche aussi le réseau veineux et artériel de la rétine. Les fibres optiques se rejoignent toutes là pour former un câble appelé le nerf optique. Il mesure 4 mm de diamètre et 5 cm de long. Il y a un nerf optique par œil, donc 2 nerfs optiques en tout. Ces 2 nerfs se croisent dans une zone appelée **chiasma** optique. A cet endroit s'entrecroise une partie seulement des fibres et plus précisément provenant de la rétine nasale.

h. Le point aveugle

On l'appel également la tache aveugle ; endroit où le nerf optique vient se raccorder à la rétine avec un trou dans la vision appelé scotome.

i. La rétine :

C'est une membrane qui tapisse le fond de l'œil (figure 1.3). Elle reçoit les signaux lumineux, et assure leur transmission au cerveau par l'intermédiaire du nerf optique.

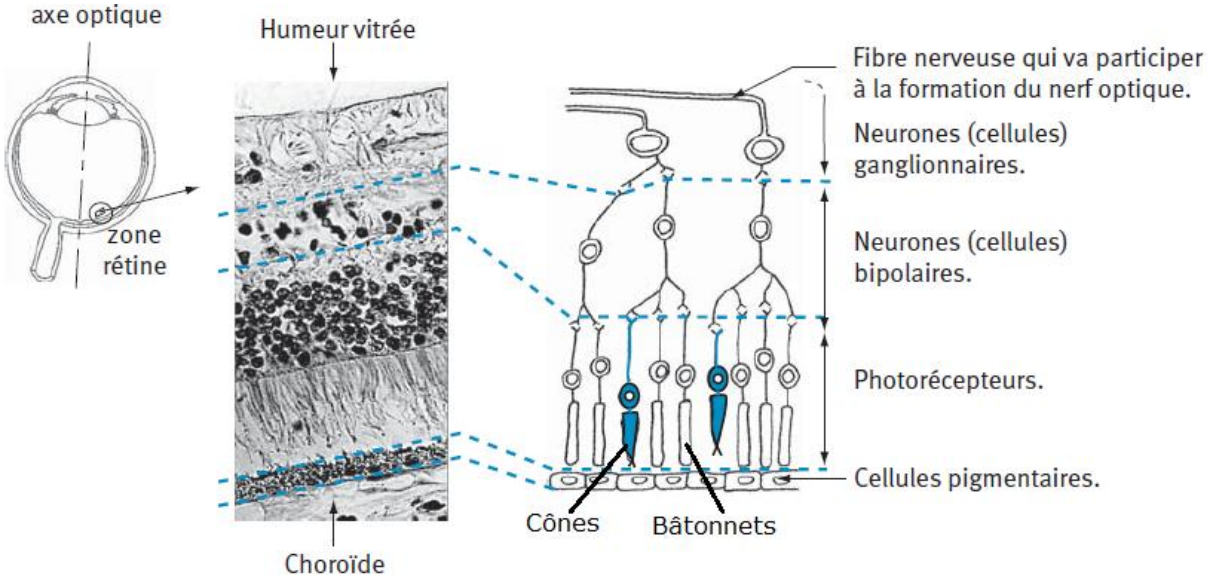

Figure 1.3 : Coupe de rétine au microscope optique (x 400) après coloration, associée à un schéma simplifié de l'organisation observée.

L'épaisseur de la rétine est d'environ 250µm sauf dans la zone centrale appelée fovéa (diamètre = 1,5 mm) où elle s'amincit pour atteindre une centaine de microns. La rétine contient deux types de photorécepteurs (figure 1.3) dont la répartition spatiale est très différente :

►**Les cônes :** Ils sont seuls présents dans la fovéa et leur densité diminue en allant vers la périphérie. Ils sont en petit nombre (~5 millions par œil), de sensibilité moyenne, de Grande vitesse de réponse et de Sensibles à la couleur

►**Les bâtonnets** : Au contraire des cônes ils sont absents de la zone centrale mais réparties sur tout le reste de la rétine. Ils sont très nombreux (~110 millions par œil), très sensibles, avec une insensibles à la couleur et lents à l'adaptation.

Ces photorécepteurs, cônes et bâtonnets, fonctionnent selon trois modes:

– **Photopique** : vision nette et colorée, due aux cônes.
– **Scotopique** : vision achromatique et moins nette, due aux bâtonnets.
– **Mésopique**, vision mixte où tous les photorécepteurs sont actifs.

Les deux types de cellules de la rétine ont une fonction différente : les bâtonnets donnent la vision à faible intensité, tandis que les cônes sont sensibles aux couleurs, sous forte lumière.

► **Fovéa :** Région rétinienne, située au niveau de l'axe optique de l'œil, en forme de petite dépression de la partie centrale (de 2 mm de diamètre) de la macula, Elle comporte essentiellement des cônes. Elle est caractérisée par une acuité visuelle élevée.

►**Macula ou tache jaune :** Portion centrale de la rétine apparaissant comme une fine excavation uniquement composée de cônes. N'occupant que 2 à 3% de la surface de la rétine, elle transmet 90% de l'information visuelle traitée par le cerveau. Ceci s'explique par sa position, en plein dans l'axe optique de l'œil.

1.3. Naissance de messages nerveux dans la rétine

La rétine se trouve au fond de l'œil. Elle agit comme un écran de cinéma au fond d'une salle noire. Elle réceptionne l'image formée par la cornée et le cristallin. La rétine a pour tâche de transmettre l'image (la lumière) au cerveau via le nerf optique. Diverses cellules rétiniennes vont transformer l'image en impulsions électriques que le cerveau pourra comprendre (figure 1.4).L'image que la rétine transmet au cerveau est étonnante : elle est inversée, presque entièrement monochrome (en noir et blanc), très floue, elle présente une zone noire qui correspond à l'extrémité du nerf optique (la tache aveugle), elle est sans cesse en mouvement, l'œil n'est jamais immobile et elle est nette seulement dans une zone centrale appelée : la macula (la macula présente une zone centrale qui analyse la couleur : la fovéa)

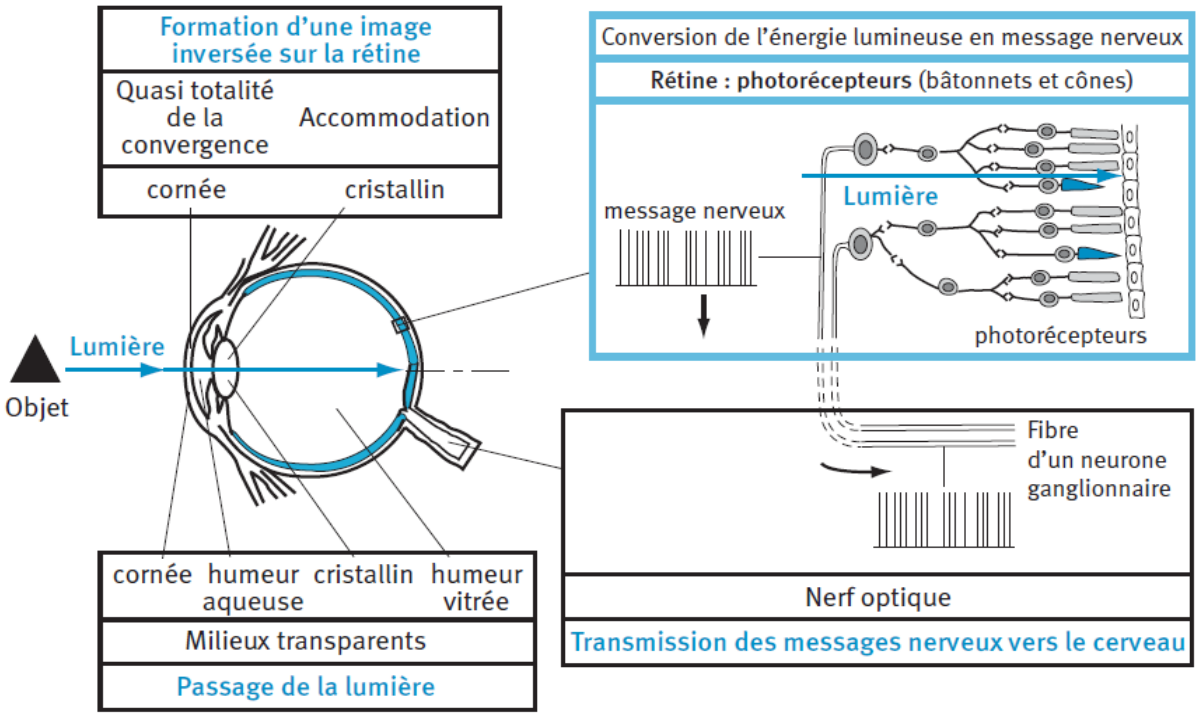

Figure 1.4 : L'œil : de la lumière aux messages nerveux.

1.4. Les électrorétinogrammes

L'enregistrement de l'activité électrique des cellules rétiniennes est l'électrorétinogramme. L'électrophysiologie visuelle consiste à enregistrer les signaux bioélectriques émis par l'œil suite à une stimulation lumineuse. L'électrorétinogramme ou ERG, enregistre l'activité des photorécepteurs et des couches rétiniennes plus internes (à l'exception des cellules ganglionnaires). Il est fonction du nombre et du niveau d'implication de chaque catégorie de cellules rétiniennes.

On réalise cet examen, afin de tester le fonctionnement des cellules visuelles, sur des yeux, dont les pupilles ont été dilatées au préalable. Le sujet est placé dans l'obscurité afin que ses rétines soient adaptées à celle-ci. On pose ensuite des électrodes externes (elles sont, par exemple, collées sur la peau au niveau de la paupière inférieure) qui vont permettre

l'enregistrement, pour chaque œil, de la réponse électrique des rétines. Les flashes lumineux ont une durée inférieure à 5ms.

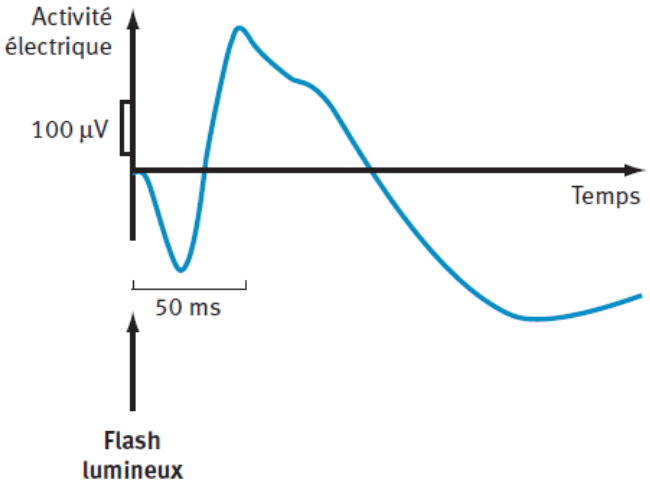

Figure 1.5 : Exemple d'un électrorétinogramme.

Des chercheurs ont enregistré l'activité électrique d'une fibre d'une cellule ganglionnaire dans l'obscurité, puis suite à un flash lumineux. Les messages nerveux (cours de seconde) sont des trains de signaux de nature électrique. Chaque trait vertical dans la figure correspond à un de ces signaux. C'est la variation de fréquence de ces signaux qui a valeur de message.

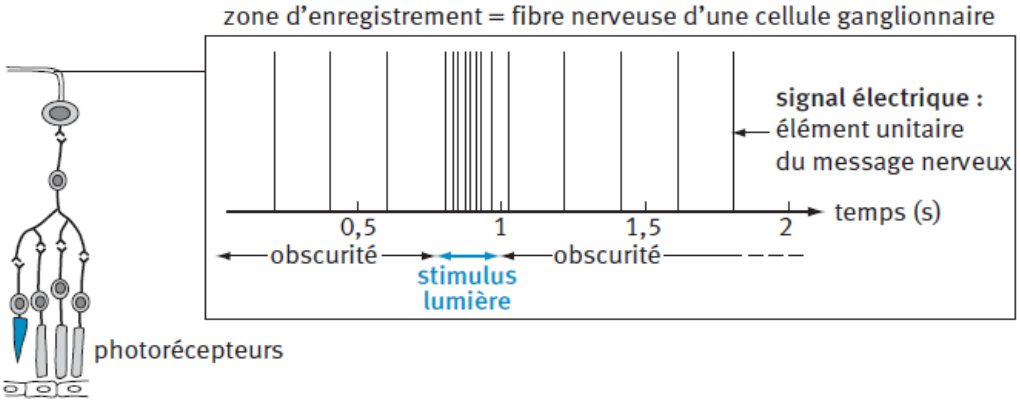

Figure 1.6 : La variation de la fréquence des signaux des messages nerveux.

1.5. Persistance rétinienne

Toutes les images se forment sur la rétine sont « gardées en mémoire » par la rétine. Ce phénomène dure environ 50 ms (figure 1.5). Cela correspond au temps nécessaire pour que les photorécepteurs redeviennent à nouveau sensibles à la lumière. Ce phénomène s'appelle la persistance rétinienne.

La persistance rétinienne fut observée pour la première fois par Léonard de Vinci pendant la Renaissance. Ce célèbre savant ne put cependant prouver ce phénomène, et ce fut le chimiste et physicien britannique Michael Faraday qui le démontra en 1825. [3]

La persistance rétinienne est la capacité ou défaut de l'œil à conserver une image vue superposée aux images que l'on est en train de voir (c'est la durée du traitement du signal

biologique). Cette capacité explique le phénomène d'animation lorsqu'on regarde un dessin animé par exemple : une séquence d'images fixes qui défilent très rapidement nous donne la sensation de mouvement. On distingue deux types de persistances rétiniennes : la persistance positive, très rapide (environ 50 ms) et la persistance négative, résultant d'une exposition prolongée à la lumière. [4]

►**La persistance positive** : Elle laisse sur la rétine une image lorsqu'on ferme les yeux. Elle dure 50 à 100 millisecondes, et est responsable de la fluidité des images d'un film qui montre paradoxalement des images de façon saccadée.

►**La persistance négative** Elle dure plus longtemps (plusieurs secondes) et apparaît avec les très fortes illuminations. En fermant les yeux, une image inversée apparaît alors.

Si l'on regarde fixement un point coloré pendant une vingtaine de secondes, et qu'ensuite on porte son regard sur un fond uni ; la persistance de l'image dure pendant un bref instant. On constate que la force de l'habitude reste souvent, pour le cerveau, le moteur le plus puissant lors de la perception des illusions d'optiques. Cette propriété de l'œil est utilisée par le cinéma et la télévision pour donner l'impression d'un mouvement continu à partir d'une séquence d'images.

1.6. Origine de la persistance rétinienne

La persistance rétinienne est due au fait que les cellules photoréceptrices, sur la rétine, saturent si la quantité de lumière est donc la charge électrique reçue est trop importante (chacun son seuil).

Il est donc normales qu'un flash d'appareil photo par exemple, qui contient plus d'énergie, "persiste" plus longtemps car les cellules photoréceprices mettent plus de temps à se décharger et continuent à envoyer des informations au cerveau même si les paupières sont fermées.

En outre, ce phénomène est amplifié quand la luminosité ambiante est faible: seules les cellules "noir et blanc" réagissent bien dans la pénombre (on voit moins bien les couleurs dans la nuit) et les cellules "couleur" passent d'un état quasi au repos à un état de saturation.

1.7. Mystères de la persistance rétinienne

a. Le flip book (folioscope ou feuilletoscope)

En effet, lorsqu'on présente rapidement au regard une succession de deux images, on a l'impression qu'il n'y a plus deux images distinctes mais une seule, issue de la superposition des deux autres. [5] Le folioscope est un livret de dessins qui s'anime en feuilletant rapidement les pages. En 1868 l'anglais John Barnes Linnett a déposé un brevet de cette invention sous le nom de Kinéographe. [6]

b. Le thaumatrope

Il s'agit d'un jouet qui produit en tournant une illusion d'optique. Il est basé sur le principe bien connu de la persistance des impressions lumineuses sur la rétine, même lorsque l'objet qui a provoqué ces impressions a disparu. On fait tourner rapidement un disque en carton autour d'un diamètre comme axe. Les figures dessinées sur chacune de ses faces se succèdent si

rapidement qu'elles se superposent pour l'œil qui les perçoit en même temps : ce qui produit un effet curieux presque magique. [7] par exemple, Ainsi si l'on montre très rapidement l'image isolée d'une cage, puis d'un oiseau, dessinés sur chaque face d'un disque de carton on obtient l'illusion que l'oiseau est entré dans la cage. La rétine de l'œil a gardé en mémoire un bref instant chaque image, le cerveau ayant superposé les images. L'illusion du mouvement est née. [5]

c. Le phenakistiscope.

Il permet d'animer des images en les regardant défiler à travers une série de fentes. La version la plus simple nécessite un miroir pour réfléchir les images à travers les fentes de disque. [8]

d. Le zootrope

Contrairement au phénakistiscope, le zootrope permet a plusieurs personnes d'observé simultanément la même scène. Le praxinoscope est une amélioration du zootrope dans lequel les fentes sont remplacées par un jeu de miroirs qui facilitent l'observation de scène. [8]

e. Les illusions d'optiques

L'image physique formée au fond de l'œil sur la rétine, analysée point par point, est transmise fidèlement au cerveau sous forme de messages codés. Ceci est en principe pareil pour tous. Mais ce sont les zones visuelles du cerveau qui analysent ces signaux et nous donnent une représentation de l'objet perçu.

L'interprétation qu'en fait le cerveau peut parfois être ambiguë. Ces "erreurs d'interprétation sont des illusions d'optique, qui ne sont pas perçues de la même façon par chacun d'entre nous (nous n'avons pas tous le même "vécu", ni les mêmes images en mémoire). Les illusions sont les témoins des mécanismes de la vision. Elles confirment que notre perception du monde est assez éloignée de la photographie.

Elle est le résultat d'une stimulation des cônes et des bâtonnets, qui se trouvent dans la rétine, pouvant subir des phénomènes de fatigue.

1.8. Phosphènes causés par la persistance rétinienne [9]

Un phosphène est un phénomène interne à l'œil qui se traduit par la sensation de voir une lumière ou par l'apparition de tâches dans le champ visuel. Les phosphènes peuvent être causés par une stimulation mécanique, électrique, ou magnétique de la rétine ou du cortex visuel mais aussi par une destruction de cellules dans le système visuel.

C'est un phénomène optique dû à la persistance rétinienne. Lorsque ce phénomène bien connu aveugle momentanément, on parle d'éblouissement (au volant par exemple).

Ce phénomène survient normalement après fixation d'une source lumineuse ponctuelle, mais peut intervenir lors de troubles psychologiques, comme l'épilepsie ou la migraine ophtalmique, ou suite à la prise d'un psychotrope (médicament ou drogue de type hallucinogène). Les phosphènes peuvent être un signe précurseur d'un décollement de rétine.

1.9. pathologie lié à la persistance rétinienne

La rétine est fragile. Un fort éblouissement (par ex: observation directe du soleil) peut l'endommager, avec perte irrémédiable de cellules de la rétine et ça implique une dégradation des performances de la vue.

1.9.1 La palinopsie [4]

a. Définition

La palinopsie est un trouble de la vision caractérisé par des hallucinations. Le sujet voit des objets anormalement longtemps, même lorsque ceux-ci sont hors de son champ de vision. Ces objets apparaissent très lumineux et les images se superposent.

b. Les causes

Ce trouble peut être provoqué par un médicament (antidépresseur), le fait de se présenter devant un écran d'ordinateur des heures par jours, une drogue ou une lésion cérébrale au niveau du lobe occipital (tumeur, AVC). Lorsque la palinopsie est d'origine médicamenteuse, elle peut persister plusieurs années après l'interruption du traitement.

c. Les symptômes

Le symptôme est caractérisé par une persistance rétinienne anormale, une baisse de vision dans la pénombre, et un écran "neigeux" permanent.

1.9.2 Décollement de rétine [4]

a. Définition :

Le décollement de rétine est une affection grave sur le plan fonctionnel qui, non traitée, aboutit à la cécité. Il peut être primitif ou secondaire :

Le décollement primitif est plus fréquent chez le myope à cause des lésions dégénératives périphériques.

Le décollement secondaire peut être dû à un traumatisme ou favorisé par certaines affections : rétinite diabétique, tumeurs sous-jacentes, etc.

b. Symptômes :

Les phosphènes sont des éclairs de lumière, survenant toujours au même endroit. Sensation de voile, baisse d'acuité visuelle si le décollement de rétine intéresse la région maculaire.

1.10. Conclusion

Dans ce chapitre, nous avons définis la persistance rétinienne et décrit brièvement l'œil et ses éléments principaux, et cité les différents mystères et les phénomènes responsables de la persistance rétinienne. Ainsi que les différentes pathologies liées à ce phénomène.

On à pu constater dans ce premier chapitre que la stimulation lumineuse de l'œil peut s'avérer intéressante de point de vu diagnostic de l'état fonctionnel de la rétine et de ça persistance à traiter les différentes informations.

Le chapitre suivant est consacré à la présentation des microcontrôleurs PIC de la société MICROCHIP, et particulier le PIC 16f84a, que nous avons utilisé pour réaliser les stimulations temporiser de notre système.

Chapitre II :
Système à base de microcontrôleur

2.1 Introduction

Un microcontrôleur se présente comme étant une unité de traitement de l'information de type microprocesseur contenant tous les composants d'un système informatique, à savoir microprocesseur, des mémoires et des périphériques (ports, timers, convertisseurs).
Chaque fabricant a sa ou ses familles de microcontrôleur. Une famille se caractérise par un noyau commun (le microprocesseur, le jeu d'instruction). Ainsi les fabricants peuvent présenter un grand nombre de pins qui s'adaptent plus au moins à certaines tâches. Mais un programmeur connaissant une famille n'a pas besoin d'apprendre à utiliser chaque membre, il lui faut connaître juste ces différences par rapport au père de la famille. Ces différences sont souvent, la taille des mémoires, la présenceou l'absence des périphériques et leurs nombres.

L'utilisation des microcontrôleurs pour les circuits programmables à plusieurs points forts est bien réelle. Il suffit pour s'en persuader, d'examiner la spectaculaire évolution de l'offre des fabricants des circuits intégrés en ce domaine depuis quelques années. En effet, le microcontrôleur est moins cher que les autres composants qu'il remplace. D'une autre part, le microcontrôleur diminue les coûts de main d'œuvre. [10]

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. L'utilisation des microcontrôleurs ne connaît de limite que l'ingéniosité des concepteurs, on les trouve dans nos cafetières, les magnétoscopes, les radios …..Une étude menée en l'an 2004 montre qu'en moyenne, un foyer américain héberge environ 240 microcontrôleurs.

2.2. Le choix du PIC 16F :

a. définition d'un microcontrôleur (µC):

Un PIC n'est rien d'autre qu'un microcontrôleur, c'est à dire une unité de traitement de l'information de type microprocesseur à laquelle on a ajouté des périphériques internes de communication avec l'extérieur permettant de réaliser des montages sans nécessiter l'ajout de composants externes, ou du moins avec un nombre restreint de composants. [11].
C'est un ordinateur monté dans un circuit intégré. 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

C'est ce qu'on peut voir sur la figure 1 :

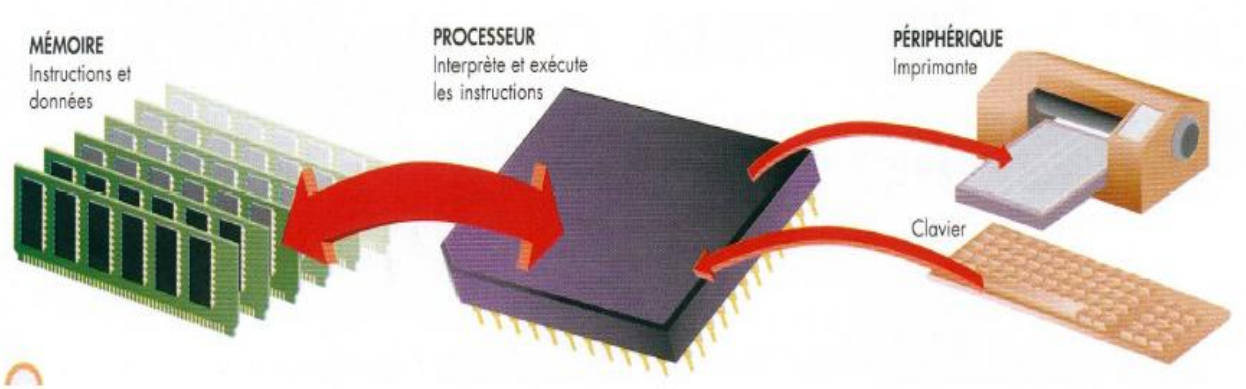

Figure 2.1

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

b. Les différentes familles de PIC : [12]

Les PICs sont subdivisée en 3 grandes familles :

- La famille **Base-Line**, qui utilise des mots d'instructions (nous verrons ce que c'est) de 12 bits pour certains PIC (12C508), de 14 pour d'autres (12F675),
- La famille **Mid-Range**, qui utilise des mots d'instruction de 14 bits (et dont font partie les 16F84 et 16F876).
- La famille **High-End**, qui utilise des mots d'instruction de 16 bits (18FXXX).

c. L'identification de PIC : [12]

Pour identifier un PIC, on utilise simplement son numéro :

- Les 2 premiers chiffres indiquent la catégorie du PIC, 16 indique un PIC Mid-Range.

- Vient ensuite parfois une lettre L, celle-ci indique que le PIC peut fonctionner avec une plage de tension beaucoup plus tolérante.
- Vient ensuite une ou deux lettres pour indiquer le type de mémoire programme :
  - C indique que la mémoire programme est une EPROM ou plus rarement une EEPROM
  - CR pour indiquer une mémoire de type ROM
  - F pour indiquer une mémoire de type FLASH.

- On trouve ensuite un nombre qui constitue la référence du PIC.
- On trouve ensuite un tiret suivi de deux chiffres indiquant la fréquence d'horloge maximale que le PIC peut recevoir.

d. Architecture mémoire

La donnée de base de PIC (16FXXX) est l'octet. L'octet comporte 8 bits. Le bit 0 est le bit de poids faible et le bit 7 est le bit de poids fort.

e. Le choix d'un PIC16F84A est directement lié à l'application envisagée.

Le choix d'un microcontrôleur est primordial car c'est de lui que dépendront en grande partie les performances, la taille et la facilité d'utilisation de l'application envisagée. En fait ce choix est imposé dans le cahier de charge. La société Microchip offre une vaste gamme de microcontrôleur. Afin de choisir un PIC adéquat à notre projet, nous avons pensée à l'utilisation du PIC 16f84A, qui a un nombre de ports entrée/sortie suffisant. Il possède 16 broches et une mémoire de programme 1024 × 14 bits, une fréquence de 20 MHz, ce qui dans notre cas est largement suffisant.

De plus le PIC 16f84Apossède un jeu instructions puissant, et permet donc le développement programme simple et réduit.

Le choix dépend aussi aux :

- Leur disponibilité
- Leur prix
- Il faut dans d'un première temps déterminer le nombre d'entrées/sorties nécessaires pour l'application.
  Ce nombre d'entrées/sorties nous donne une première famille de PIC.
- Il faut ensuite déterminer si l'application nécessite un convertisseur Analogique/Numérique ce qui va centrer un peu plus vers le choix d'une famille de PIC.
- La rapidité d'exécution est un élément important, il faut consulter les DATA-BOOK pour vérifier la compatibilité entre la vitesse maximale du PIC choisi et la vitesse max nécessaire au montage.
- La taille de la RAM interne et la présence ou non d'une EEPROM pour mémoriser des données est également important pour l'application souhaitée, de la mémoire EEPROM (64 octets) et peuvent donc stocker des paramètres.
- La longueur de programme de l'application détermine la taille de la mémoire programme du PIC recherché.

- Leur technologie : la mémoire FLASH permet un effacement et une reprogrammation multiple du circuit.
- La documentation et l'assembleur sont largement disponibles sur les sites Web des constructeurs.

f. Brochage et fonction des pattes [13]

Figure 2.1 montre le brochage du circuit. Les fonctions des pattes sont les suivantes :

- VSS, VDD : 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

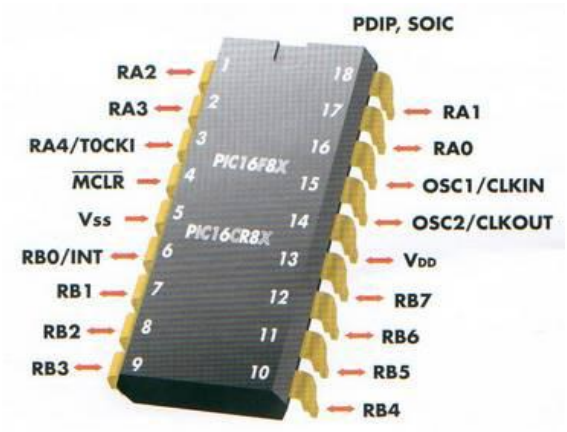

Figure 2.2 : Brochage du circuit.

L'alimentation du circuit est assurée par les pattes VDD et VSS. Elles permettent à l'ensemble des composants électroniques du PIC de fonctionner.

Pour cela on relie VSS (patte 5) à la masse (0 Volt) et VDD (patte 14) à la borne positive de l'alimentation qui doit délivrer une tension continue comprise entre 3 et 6 Volts.

Le microcontrôleur est un système qui exécute des instructions les unes après les autres à une vitesse (fréquence) qui est fixée par une horloge interne au circuit. Cette horloge doit être stabilisée de manière externe au moyen d'un cristal de quartz connecté aux pattes OSC1/CLKIN (patte 16) et OSC2/CLKOUT (patte 15).

La patte 4 est appelée MCLR. Elle permet lorsque la tension appliquée est égale à 0V de réinitialiser le microcontrôleur. C'est à dire que si un niveau bas (0 Volt) est appliqué sur MCLR le microcontrôleur s'arrête, place tout ses registres dans un état connu et se redirige vers le début de la mémoire de programme pour recommencer le programme au début (adresse dans la mémoire de programme : 0000).

A la mise sous tension, la patte MCLR étant à zéro, le programme démarre donc à l'adresse 0000,(MCLR=Master CLear Reset).

Les broches RB0 à RB7 et RA0 à RA4 sont les lignes d'entrées/sorties numériques. Elles sont au nombre de 13 et peuvent être configurées en entrée ou en sortie. Ce sont elles qui permettent au microcontrôleur de dialoguer avec le monde extérieur (périphériques).

L'ensemble des lignes RB0 à RB7 forme le port B et les lignes RA0 à RA4 forment le port A.

Certaines de ces broches ont aussi d'autres fonctions (interruption, timer)

g. Les différents registres de PIC 16F84A :

►Les registres fondamentaux :

- **Le registre (W) :**

Ce registre de travail que l'on peut appeler accumulateur est un registre utilisé pour réaliser des calculs, le résultat d'un calcul peut être sauvegardé dans un emplacement RAM (F) ou dans le registre de travail (W). (Il n'y a pas d'adresse). [14]

- **Le registre pointeur de pile :** (PP ou SP en anglais).

Est essentiellement utilisé lorsque l'on réalise un sous-programme .le pointeur de pile est chargé de mémoriser l'adresse courante que contient le compteur de programme avant le saut à l'adresse du sous-programme .lorsque le sous-programme est terminé, le pointeur restitue l'adresse sauvegardée vers le compteur de programme. [15]

- **Le registre d'instruction :**

Contient tous les codes binaires correspondant aux instructions à réaliser par le microcontrôleur, le PIC 16F84A comporte **35** instructions. [15]

►Les registres spéciaux :(Les registres internes)

Tableau des registres :

Adresse	Registre	Registre	Adresse
00	INDF	INDF	80
01	TMR0	OPTION	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	FSR	FSR	84
05	PORTA	TRISA	85
06	PORTB	TRISB	86
07			87
08	EEDATA	EECON1	88
09	EEADR	EECON2	89
0A	PCLATH	PCLATH	8A
0B	INTCON	INTCON	8B
0C à 4F	68 mémoires disponibles	Accès aux même Mémoires que page 0	8C à CF
Page 0			Page 1

Tableau : 2.1 : Les registres internes du PIC 16F84 [14]

- **Les registres PCL et PCLATH :**

Le compteur de programme ou compteur ordinal nommé PC permet de pointer sur la prochaine instruction a exécuter, le PC complet étant d'office sur 13 bits .Il faut 2 registres, le premier accessible en lecture et écriture contient l'adresse basse du PC sur 8 bits, il est appelé PCL (PC Low), le deuxième s'appelle : PCLATH (PC LATch counter High). [14]

- **Le registre INTCON (INTerrupt CONtrol) :**

Ce registre se situe à l'adresse 0x0B, dans les 2 banques. Il est donc toujours Accessible. C'est un registre de bits, donc, chaque bit a une fonction particulière. [11] Voici le détail de ces bits :

b7 : GIE

Global **I**nterrupt **E**nable bit. Il permet de valider ou d'invalider toutes les interruptions d'une seule fois. Ce bit correspond donc à notre interrupteur de validation générale.

b6: EEIE

EEprom write complete **I**nterrupt **E**nable bit. Ce bit permet de valider l'interruption de fin d'écriture en eeprom (nous étudierons plus tard le mécanisme d'écriture eeprom).

b5 : T0IE

Tmr**0** **I**nterrupt **E**nable bit : Valide l'interruption générée par le débordement du timer0. Attention, c'est le chiffre zéro et non la lettre « O ».

b4 : INTE

INTerrupt pin **E**nable bit : Valide l'interruption dans le cas d'une modification de niveau de la pin RB0.

b3 : RBIE

RB Interrupt **E**nable bit : Valide les interruptions si on a changement de niveau sur une des entrées RB4 à RB7.

b2 : T0IF

Tmr**0 I**nterrupt **F**lag bit. C'est un Flag, donc il signale. Ici c'est le débordement du timer0. Tous les bits terminés par F dans ce registre sont des flags (drapeaux, dans le sens « signalisation »).

b1 : INTF

INTerrupt pin **F**lag bit : signale une transition sur la pin RB0 dans le sens déterminé par INTEDG du registre OPTION (b6).

b0 : RBIF

RB Interrupt **F**lag bit : signale qu'une des entrées RB4 à RB7 a été modifiée.

- **Les registres INDF et FSR :**

Le registre INDF ainsi que FSR permettent l'adressage INDIRECTE. Le principe est d'utiliser deux registres intermédiaires pour accéder aux données: -Dans le premier appelé FSR, on inscrit l'adresse (en fait, il est utilisé comme un pointeur).L'autre registre INDF contient la valeur qui se trouve à l'adresse définie dans le FSR. [16]

- **Le registre OPTION :**

Ce registre se trouve à l'adresse 0x81, donc dans la banque1. Dans les fichiers « include » de MPLAB, ce registre est déclaré avec le nom **OPTION_REG**[11] .
Ce registre est un registre de bits, c'est à dire que chaque bit a un rôle particulier :

b7 : RBPU

Quand ce bit est mis à 0 (actif niveau bas), une résistance de rappel au +5 volts est placée sur chaque pin du PORTB. Nous verrons dans cette leçon le fonctionnement du PORTB.

b6 : INTEDG

Donne, dans le cas où on utilise les interruptions sur RB0, le sens de déclenchement de l'interruption. Si b6 = 1, on a interruption si le niveau sur RB0 passe de 0 vers 1. Si b6 = 0, l'interruption s'effectuera lors de la transition de 1 vers 0. Comme nous n'utilisons pas les interruptions dans cette leçon, nous pouvons laisser **b6 = 0**.

b5 : TOCS

Notre PIC ne souffre pas de troubles obsessionnels compulsifs. Ce bit détermine simplement le fonctionnement du timer0, que nous verrons bientôt. Retenez que le timer0 est incrémenté soit en fonction de l'horloge interne (synchronisé au programme), dans ce cas b5 = 0, soit il compte les impulsions reçues sur la pin RA4, dans ce cas b5=1.

b4 : TOSE

Donne, pour le cas ou le bit 5 serait 1, le sens de la transition qui détermine le comptage de tmr0. Si b4 = 1, on a comptage si le signal passe de 5V à 0V sur RA4, si on a b4 = 0, ce sera le contraire.

Comme nous avons placé b5=0, b4 est alors inutilisé. Nous laisserons donc b4 = 0.

b3 : PSA

A ce niveau, sachez simplement que ce prédiviseur peut servir à une des deux fonctions suivantes (et pas les deux) : soit il effectue une prédivision au niveau du timer du watchdog (b3 = 1), soit il effectue une prédivision au niveau du timer 0 (tmr0) (b3=0).

b2, b1, b0 : PS2, PS1, PS0 :

Ces trois bits déterminent la valeur de prédivision pour le registre déterminé ci-dessus. Il y a donc 8 valeurs possibles.

Si vous désirez ne pas utiliser de prédiviseur du tout, la seule méthode est de mettre b3=1 (prédiviseur sur watchdog) et PS2 à PS0 à 0. Dans ce cas : pas de prédiviseur sur tmr0, et prédiviseur à 1 sur watchdog, ce qui correspond à pas de prédiviseur non plus. Nous mettrons donc **b2=b1=b0= 0**.

- **Le registre PORTA :**

Ce registre est un peu particulier, puisqu'il donne directement accès au monde extérieur. C'est en effet ce registre qui représente l'image (numérique 0/1) des pins RA0 à RA4, soit 5 pins. [11]

Si vous suivez toujours, c'est ce registre qui va servir à allumer la LED.
Ce registre se situe à l'adresse 05H, dans la banque 0. Chaque bit de ce registre représente une pin.

Pour écrire sur une pin en sortie, on place le bit correspondant à 1 ou à 0, selon le niveau souhaité.

Par exemple :

```
bsf PORTA , 1 ; envoyer niveau 1 sur RA1
```

- **Le registre TRISA :**

Ce registre est situé à la même adresse que PORTA, mais dans la banque 1. Son adresse complète sur 8 bits est donc 0x85.

Ce registre est d'un fonctionnement très simple et il lié au fonctionnement du PORTA.

Chaque bit positionné à 1 configure la pin correspondante en entrée. Chaque bit à 0 configure la pin en sortie.

- **Les registres PORTB et TRISB :**

Ces registres fonctionnent exactement de la même manière que PORTA et TRISA, mais concernent bien entendu les 8 pins RB.

Voyons maintenant les particularités du PORTB. Nous en avons déjà vu une, puisque les entrées du PORTB peuvent être connectées à une résistance de rappel au +5V de manière interne. [11]

- **Le registre « STATUS » :**

C'est un registre dont chaque bit a une signification particulière. Il est principalement utilisé pour tout ce qui concerne les tests. Il est donc également d'une importance fondamentale. [11]

Voici les différents bits qui le composent :

b0 : C Carry (report) :

Ce bit est en fait le 9ème bit d'une opération.

Par exemple, si une addition de 2 octets donne une valeur >255 (0xFF), ce bit sera positionné à 1. C'est également l'emprunt lorsque l'opération est une soustraction, auquel cas il est positionné si le résultat de la soustraction n'est pas négatif.

b1 : DC Digit Carry :

Ce bit est utilisé principalement lorsque l'on travaille avec des nombres BCD : il indique un report du bit 3 vers le bit 4.

Pour info, un nombre BCD est un nombre dont chaque quartet représente un chiffre décimal.

b2 : Z Zéro

Ce bit est positionné à 1 si le résultat de la dernière opération vaut 0. Rappelez-vous cependant que ces flags ne sont positionnés que pour les instructions qui le précisent (Status bit affected).

b3 : PD Power Down

Indique quel événement a entraîné le dernier arrêt du PIC

b4 : TO Time-Out

Ce bit indique (si 0), que la mise en service suit un arrêt provoqué par un dépassement de temps ou une mise en sommeil. Dans ce cas, PD effectue la distinction.

b5 : RP0 Register Bank Select0

Permet d'indiquer dans quelle banque de RAM on travaille.0 = banque 0.

b6 : RP1 Register Bank Select1

Permet la sélection des banques 2 et 3. Inutilisé pour le 16F84a, doit être laissé à 0 pour garantir la compatibilité ascendante (portabilité du programme).

b7 : IRP Indirect RP

Permet de décider quelle banque on adresse dans le cas de l'adressage indirect

- **Le registre tmr0 :**

Ce registre, qui se localise à l'adresse 0x01 en banque 0, contient tout simplement la valeur actuelle du timer0. Vous pouvez écrire ou lire tmr0.

- **Le registre EEDATA :**

Ce registre de 8 bits permet de lire ou d'écrire une donnée dans la mémoire non volatile (EEPROM). [16]

- **Le registre EEADR :**

Registre de 8 bits qui contient l'adresse de la donnée se trouvant dans l'EEPROM. [16]

- **Le registre EECON 1 :**

Registre de contrôle permettant de définir le mode de fonctionnement (écriture et l'écriture) de la mémoire de donnée EEPROM. [16]

h. Les interruptions :

Une interruption comme son nom l'indique, interrompt un programme en cours (programme principal) pour faire exécuter à le microcontrôleur un autre travail (suite d'instructions appelée aussi sous-programme). Celui-ci se termine par une instruction de retour d'interruption (RETFIE) qui permet à le microcontrôleur de reprendre le programme principale ou il a été quitté. [14]

a) Le mode d'interruption

C'est évidemment le mode principal d'utilisation du timer0. En effet, lorsque T0IE est positionné dans le registre INTCON, chaque fois que le flag T0IF passe à 1, une interruption est générée.

b) Les sources d'interruptions du 16F84a :

Le PIC16F84 est très pauvre à ce niveau, puisqu'il ne dispose que de 4 sources d'interruptions possibles (contre 13 pour le PIC16F876 par exemple).

Les événements susceptibles de déclencher une interruption sont les suivants :

►**TMR0** : Débordement du timer0 (tmr0). Une fois que le contenu du tmr0 passe de 0xff à 0x00, une interruption peut être générée.

►**EEPROM** : cette interruption peut être générée lorsque l'écriture dans une case EEPROM interne est terminée.

►**RB0/INT** : Une interruption peut être générée lorsque, la pin RB0, encore appelée **INT**errupt pin, étant configurée en entrée, le niveau qui est appliqué est modifié.

►**PORTB** : De la même manière, une interruption peut être générée lors du changement d'un niveau sur une des pins RB4 à RB7. Il n'est pas possible de limiter l'interruption à une seule de ces pins. L'interruption sera effective pour les 4 pins ou pour aucune.

Principe :

Lorsqu'une interruption est provoquée :

- Le programme termine l'instruction en cours
- Il arrête la procédure qu'il était en train d'exécuter
- Il passe à l'adresse 0x04
- À cette adresse doit se trouver l'instruction qui envoie sur la routine de traitement de l'interruption
- À la fin de cette routine de traitement, il reprend la procédure initiale là où elle était.

c) Le watchdog :

Le watchdog, ou chien de garde est un mécanisme de protection de votre programme. Il sert à surveiller si celui-ci s'exécute toujours dans l'espace et dans le temps que vous lui avez attribués.

Par exemple, si le programme attend le résultat d'un système extérieur (conversion analogique numérique par exemple) et qu'il n'y a pas de réponse, il peut rester bloqué. Pour en sortir, on utilise un chien de garde. Il s'agit d'un compteur qui, lorsqu'il arrive en fin de comptage, permet de redémarrer le programme. Il est lancé au début du programme. En fonctionnement normal, il est remis à zéro régulièrement dans une branche du programme qui s'exécute régulièrement. Si le programme est bloqué, il ne passe plus dans la branche de remise à zéro et le comptage va jusqu'au bout et déclenche le chien de garde qui relance le programme.

►Le principe de fonctionnement :

La mise en service ou l'arrêt du watchdog se décide au moment de la programmation de votre PIC, à l'aide de la directive _CONFIG. Si « _WDT_OFF » est précisé, le watchdog ne sera pas en service. Si au contraire vous précisez « _WDT_ON », le watchdog sera actif.

Le fonctionnement du watchdog est lié à un timer interne spécifique, qui n'est pas synchronisé au programme, ni à un événement extérieur, et qui ne dépend pas de la fréquence de l'horloge de votre PIC.

d) Le prédiviseur :

Le prédiviseur pouvait être affecté au tmr0 ou au watchdog, via le bit PSA du registre OPTION.

Réglage de la pré division. Ces trios bits déterminant la valeur de pré division pour le registre détermine par le bit PSA . La pré division est différente pour le compteur timer0 (TMR0) et pour chien de garde (Watch Dog). Voir tableau suivant (tableau :2.1) .

PSA	PS2	PS1	PS0	/tmr0	/WD	Temps tmr0 (cycles)
0	0	0	0	2	1	512
0	0	0	1	4	1	1024
0	0	1	0	8	1	2048
0	0	1	1	16	1	4096
0	1	0	0	32	1	8192
0	1	0	1	64	1	16384
0	1	1	0	128	1	32768
0	1	1	1	256	1	65536
1	0	0	0	1	1	256
1	0	0	1	1	2	256
1	0	1	0	1	4	256
1	0	1	1	1	8	256
1	1	0	0	1	16	256
1	1	0	1	1	32	256
1	1	1	0	1	64	256
1	1	1	1	1	128	256

Tableau 2.1 : les prédiviseurs du timer et du watchdog.

- **PSA à PS0 :** bits de configuration du prédiviseur.

- **/tmr0 :** valeur du prédiviseur résultante sur le timer0.
- **/WD :** valeur du prédiviseur résultante sur le Watchdog.
- **temps tmr0** : nombre de cycles d'instructions entraînant un débordement de tmr0.

2.3. Langages de programmation :

a. Définition :

Un langage de programmation est un moyen formel permettant de décrire des traitements (des tâches à effectuer) sous la forme de programmes (de séquences d'instructions et de données de « haut niveau », c.à.d. compréhensibles par le programmeur) pour lesquels il existe un traducteur pour en permettre l'exécution effective par un ordinateur. [13]

Un langage de programmation est un ensemble de suites (appelés mots) de symboles choisis dans un ensemble donné (appelé alphabet) qui vérifient certaines contraintes spécifiques au langage (syntaxe). [17]

Dans le cas des langages de programmation, on trouve dans l'alphabet plusieurs types de symboles :

– des mots-clés. Ex : main, int, if, for ...,
– des opérateurs. Ex : =, <, & ...,
– des chiffres et des lettres permettant d'identifier des variables ou est constantes,
– des caractères spéciaux : Ex : accolades, crochets, point-virgule, tiret bas ... permettant de structurer le tout...

Traducteur : programme qui convertit un programme quelconque écrit dans un langage source en un programme écrit dans un langage cible.
• Assembleur : traducteur de langage objet
• Compilateur : traducteur de langage source évolué
Interpréteur : programme qui traduit puis exécute un programme quelconque écrit dans un langage source.

►Traducteur :

1) Compilateur : [13]

Traduit une seule fois les programmes dans leur ensemble :
Tout le programme est fourni au compilateur pour la traduction et son résultat (code objet) peut être soumis au processeur pour traitement.
Un langage de programmation pour lequel un compilateur est disponible est appelé un **langage compilé.**

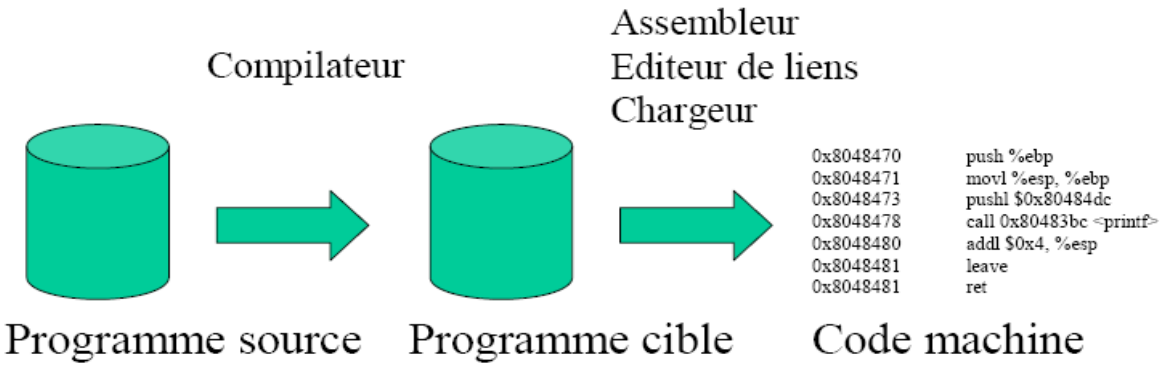

Figure 2.3 : langage compilé.

►Interpréteur : [13]

• Traduit les programmes instruction par instruction et soumet chaque instruction traduite au processeur pour exécution.

• Un langage de programmation pour lequel un interpréteur est disponible est appelé un **langage interprété.**

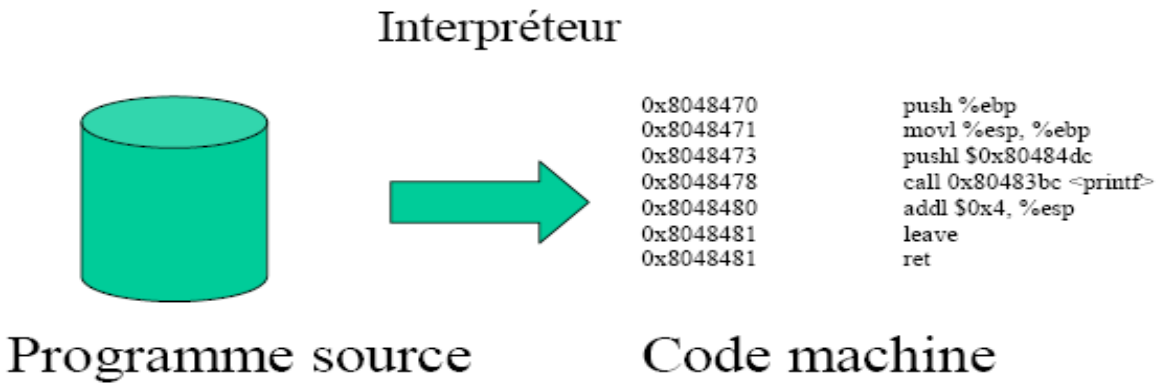

Figure 2.4 : langage interprété.

►Langages compilés, interprétés, semi-compilés :

• Certains langages (LISP par exemple) peuvent être indifféremment interprété ou compilé ; d'autres ne sont que compilé jamais interprété (C, C++, Pascal, ADA) ; certains ne sont qu'interprété (Prolog).

• Certains sont semi-compilés (par exemple Java, V-Pascal) : ils sont compilés en langage de type « assembleur » (byte-code) puis traitée par une « machine virtuelle » qui interprète le byte-code [l'idée sous-jacente est de rendre le programme indépendant de la plateforme].

b. Les différents langages de programmation :

Comme vous le savez déjà, un programme ne se fait pas comme ça ; il faut rentrer des instructions dans tel ou tel langage pour que l'ordinateur puisse les comprendre et les exécuter. La plupart des langages de programmation partage quelques bases communes. La plus évidente est que les instructions sont en anglais ou inspirées de l'anglais.

Tous les langages permettent de manipuler différents types de variables (qui peuvent contenir diverses formes de nombres, des caractères, du texte....) Les instructions sont lues et exécutées les unes après les autres mais il existe des instructions de branchement qui permettent de "sauter" à une position donnée et donc de faire des boucles (blocs d'instructions qui se répètent sous certaines conditions), des fonctions (sorte de sous-programme qui renvoie un résultat à partir d'une ou plusieurs variables).

Tous les langages n'offrent pas les mêmes possibilités ; certains sont plus faciles à transférer sur une autre plate-forme (Linux, Solaris, Mac...) que les autres, certains sont plus faciles à manipuler, certains gèrent mieux les périphériques, etc…

Voyons les principaux langages de programmation : au programme (pas de jeu de mot intentionnel) :l'Assembleur (ASM), le Cobol, le BASIC, le JAVA, le C/C++, le Pascal, le Visual Basic, le Delphi, les langages du web (HTML - CSS - JS - PHP - SQL), le Flash.

► Le langage C : [18]

Le langage de programmation C est la référence pour programmer les microcontrôleurs avec des exigences temps réel.

En effet, ce dernier présente l'avantage d'être un langage de programmation orienté matériel.

Le code machine généré par le compilateur est presque optimal en ce qui concerne d'une part la taille de la mémoire requise et d'autre part la vitesse d'exécution du programme. Par conséquence, environ 65% des applications embarquées sont programmées en C.

Toutefois, le langage C présente également des inconvénients.

► Le langage C++ : [18]

C ++ nécessite plus de ressources (utilisation de la mémoire, la puissance du processeur) que C (environ 20%).

Cela dépend essentiellement comment le programme a été défini en C++. Est-ce que ce dernier utilise le polymorphisme ou non. Le « C ++ embarqué » est une version plus dépouillée du C++.

C++ présente les avantages de la programmation orientée objet. C qui est particulièrement intéressant pour les applications plus complexes.
Environ 25% des applications embarquées sont programmés en C++.

►Assembler : [18]

Les applications programmées uniquement en assembleur sont relativement rares. Par contre, l'assembleur est souvent combiné avec du C ou du C++. Ce langage est

principalement utilisé pour optimiser individuellement les fonctions. Comme par exemple pour les drivers (accès direct au Hardware) ou les algorithmes qui nécessitent des calculs intensifs. Moins de 10% de tous les applications embarquées sont programmées (du moins partiellement) en l'assembleur.

►Java [18] :

Java est aujourd'hui rarement utilisée pour programmer les applications embarquées (moins de 3%). Java n'est pas adéquate pour les applications temps réels. En effet, de langage est interprété (lent) et le comportement temporel du « récupérateur de place » (anglais : Garbage-Collection) n'est pas reproductible.

Langage	**domaine d'applications**	**Compilé / interprété**
ADA	Programmation temps réel	compilé
BASIC	Programmation à but éducatif	interprété
C	Programmation système	Compilé
C++	Programmation orientée objet	Compilé
Cobol	Gestion	Compilé
Fortran	Calcul	compilé
Java	Programmation internet	intermédiaire
MATLAB	Calcul mathématique	interprété
Mathematica	Calcul mathématique	interprété
LISP	Intelligence artificielle	intermédiaire
Pascal	Programmation a but éducatif	compilé
PHP	Développement sites Web dynamiques	interprété
Prolog	Intelligence artificielle	interprété
Perl	traitement chaines de caractères	interprété

Tableau 2.1 [17] – Quelques exemples de langages courants

2.1 Le choix de langage Assembleur

a) Définition :

(Abrégé ASM) est un langage de programmation de bas-niveau, qui fait la correspondance entre des instructions en langage machine (mots binaires) et des symboles appelés mnémoniques plus simples à utiliser.

b) Présentation du programme Assembleur :

Comme dans n'importe quel système à microprocesseur, il est nécessaire de préparer un programme qui permettra au PIC d'effectuer son travail. Un programme est constitué d'une liste d'instructions en séquence, chacune d'entre elles identifiant très précisément les fonctions de base que le PIC est capable d'effectuer.

Un programme en langage assembleur peut être écrit sur un PC en utilisant un éditeur de texte capable de générer des fichiers, Ce fichier texte est appelé fichier source assembleur. Une fois l'écriture du programme source assembleur terminée, le fichier doit être sauvegardé avec l'extension (.ASM).

La syntaxe utilisée ne doit pas contenir d'accent, d'espace inutile, de cédille, de parenthèse etc.

►L'identification

Les lignes sont précédées du symbole (;). Tout ce qui suit est une zone de commentaires, vous pouvez y inscrire ce que vous voulez. On utilisera cette zone comme zone d'identification dans laquelle on inscrira : le titre du programme, une petite description de ce qu'il fait, la date d'édition, le nom de l'auteur.

►Les directives

Les directives ne font pas partie du programme, elles ne sont pas traduites en OPCODE, elles servent à indiquer à l'assembleur de quelle manière il doit travailler. Ce sont donc des commandes **destinées à l'assembleur** lui-même**.**
Par contre, les instructions seront traduites en OPCODE et chargées dans le PIC. Il faut donc faire la distinction.

- La directive **"list"**

La directive "List" en bas de l'entête est destinée au compilateur pour lui indiquer quel type de processeur est utilisé (ici : Pic16F84a).

- Les directives **"__config**"

La ligne "__config" est aussi destinée au compilateur pour établir les bits de configuration de

- **Les bits 0 et 1 déterminent le type d'oscillateur utilisé :**

Il y a quatre (4) modes d'horloge possibles :

  - RC (résistance, capacité), 26.2Khz à 4610Khz .
  - LP (faible consommation) quartz ou oscillateur externe. 32 Khz, 200Khz .
  - XT (quartz ou résonateur céramique ou oscillateur externe. 100 Khz, 455Khz, 2 Mhz, 4Mhz).
  - HS (haute vitesse : 8Mhz, 20Mhz) quartz ou résonateur céramique ou oscillateur externe.

- **Le bit 2 autorise ou non le fonctionnement du watchdog timer :**

- La directive **« #define** » (définition) :

Une définition « #define » permet de remplacer un texte complexe par un nom particulier qu'on veut lui donner.
Dans le programme, on peut par exemple donner le nom "stat" au bit 5 du registre status (status,05 = stat). Il faut alors le définir dans le programme en utilisant la directive #define.
#define stat status,05
Pour mettre à 1 le bit 5 du registre status, on écrira alors :
bsf stat au lieu de bsf status,05

- La directive « **#include** »

Afin d'éviter l'écriture d'une multitude de lignes d'assignation au début de chaque programme, on peut les regrouper dans un fichier et les appeler par une seule commande

- La directive **« org** »

La directive "org" précise à quelle adresse sera placée l'instruction qui suit. Après un reset ou une mise sous tension, le PIC démarre toujours à l'adresse 0x00, c'est donc à cette adresse que doit débuter votre programme.
L'adresse 0x04 est utilisée par les interruptions. Nous devons donc sauter par-dessus vers le début de notre programme principal.

- La directive **« end** »

Cette directive indique l'endroit où doit cesser l'assemblage de votre programme. Elle est obligatoire sinon une erreur vous signalera que la fin de fichier a été atteinte sans rencontrer la directive "end" (End Of File). Les instructions après la directive "end" sont tout simplement ignorées.

►Les assignations ou directive « equ »
Une assignation est une simple substitution, elle associe un nombre à une étiquette. Au moment de l'assemblage, chaque assignation rencontrée sera remplacée par sa valeur.

►Les macros
Une macro remplace un morceau de code que nous utilisons souvent dans le programme. Elle fonctionne comme un simple traitement de texte.
La macro se compose d'un nom écrit en première colonne suivi de la directive "macro", d'une ou plusieurs instructions et de fin de macro "endm" (end of macro).

►Le programme principal :

Dans la mémoire programme, les instructions sont codées en suites de 0 et de 1, nous, nous préférons écrire des choses comme goto ou clrw, l'assembleur se charge de la traduction.
Chaque ligne peut contenir jusqu'à 4 types d'informations appelées champs.
Les quatre champs (ou colonnes) sont : l'étiquette, le mnémonique (OPCODE), l'opérande et le commentaire.

►Étiquette :

L'étiquette permet de donner un nom à un nombre (adresse, variable) ou à une ligne du programme. Elle commence par un caractère alphanumérique (ou un souligné), dans ce cas, le souligné ne doit pas être suivi par un chiffre. Les caractères utilisables sont lettres, chiffres, souligné et point d'interrogation. Leur longueur maximale est de 32 caractères. Par défaut il y a une différence entre une majuscule et une minuscule.

L'étiquette en début de ligne de programme commence en colonne 1; elle est suivie par un espace, une tabulation, par deux points (:) ou par un retour de chariot si elle est seule sur la ligne. Derrière un **call** ou un **goto**, elle est en colonne #3.

►Mnémonique :

C'est l'instruction elle-même qui doit commencer en colonne 2. S'il y a une étiquette en colonne 1, il doit y avoir un ou plusieurs espace, deux points (:) ou une tabulation avant le mnémonique.

►Opérande(s) :

L'opérande vient après le mnémonique, il complète l'instruction. Il doit être séparé de l'instruction par un ou plusieurs espace, par deux points (:) ou par une tabulation. S'il y a plusieurs opérandes, ils sont séparés par des virgules.

►Commentaire :

Tout texte commençant par un point-virgule est un commentaire. Il n'est pas pris en compte par la compilation jusqu'à la fin de la ligne sauf si le point-virgule fait partie d'une chaîne de caractères.

f- Reset : Il y a plusieurs possibilités de Reset :

- Reset à l'allumage (POR)
- Reset externe par mise à la masse de l'entrée MCLR
- Reset après Watchdog
- Le Power-up Timer (PWT) produit un délai de 72 ms à l'allumage.
- L'oscillateur Start-up Timer (OST) maintient le 16F84a en position reset jusqu'à ce que l'oscillateur à quartz se soit stabilisé.

c) Le langage Assembleur est un langage compilé, c'est à dire :

1. L'utilisateur écrit son programme en langage Assembleur. Ce fichier est assemblé pour traduire le programme en langage machine (avec éventuellement des améliorations).
2. Le programme en langage machine est alors utilisé pour programmer le microcontrôleur, c.à.d. qu'il est transféré dans la mémoire (programme) pour être exécuté.

d) Les avantages de l'assembleur :

L'assembleur peut vous donner la logique nécessaire pour programmer en n'importe quels langages.

On a la possibilité de faire tout et n'importe quoi avec la mémoire. L'ASM n'a pas de limite et de sécurité. Autant il peut être utile, autant il peut détruire ;)

C'est le seul langage permettant de modifier un programme compilé dont on n'a pas les sources.

L'assembleur se révèle être un langage bien à part, son apprentissage poursuit double objectif, d'une part avoir une compréhension plus profonde de la façon dont fonctionne l'ordinateur, d'autre part vous donnez des atouts considérables pour la compréhension et la maîtrise de tous les autres langages de programmation.

e) Le choix de MPLAB IDE :

Un langage compilé est préférable pour la réalisation d'applications efficaces et/ou de grande envergure :

• Programme plus efficace : le compilateur peut effectuer des optimisations plus facilement que l'interpréteur (puisqu'il possède une visibilité globale sur le programme)
• Traduction une unique fois.
• Meilleure détection des erreurs : structuration plus rigoureuse et typage.
• Protection de la propriété intellectuelle : la compilation permet de diffuser les programmes sous forme binaire sans en imposer sa diffusion sous forme lisible.

2.5 La carte de programmation :

a. définition de Programmateur du PIC :

Après avoir réalisé la programmation du PIC en langage quelconque **e**t le programme compilé, il faut le transférer dans la mémoire du microcontrôleur. Pour cela il vous faut, Une petite interface matérielle

Nous devrons avoir un programmateur de PIC, qui nous permet d'implémenter notre programme dans les registres de PIC.

Le programmateur est relié à l'ordinateur par une liaison série ou parallèle et n'a pas les supports pour tous les PICs.

b. Les différents programmateurs et leurs logiciels :

►Programmateur universel :

Ce programmateur permet de programmer une large gamme de PIC (12c508, 16f877….) aussi que la plus part des mémoires EEPROM série. Il se connecte simplement à l'interface parallèle d'un PC avec un port COM.

►Le programmateur PIC flash :

Le programmateur PICflash avec assistance mikroICD est outil très puissant destiné à la programmation des microcontrôleurs PIC 16F et PIC 18F de MICROCHIP.

Sa conception particulière et sa facilité d'utilisation en font un outil très populaire aussi bien chez les débutants que chez les professionnels.

Le programmateur PICflash communique avec le microcontrôleur via un câble USB servant également à alimenter le programmateur.

De plus c'est dispositif à faible consommation ce qui le rend idéal pour fonctionner avec des PCs portable.

En fin, notez que l'utilisation de ce programmateur nécessite l'installation de software présent sur le CD associé.

►Leur logicielle mikroICD :

Le micro ICD fait partie intègrent du programmateur embarqué son rôle est de test et déboguer les programme en temps réel se processus de test et débogage s'effectue en affichant l'état des registres et variables du microcontrôleur au cours de l'exécution du programme.

Le software mikro ICD est intégré dans tous les compilateurs pic micro- Electronika (mikroBASIC, mikroC et mikroPASCALl) aussitôt le débogueur mikroICD lancé, la fenêtre ci-dessous apparait à l'écran.

Le débogueur mikroICD communique avec le microcontrôleur par les broches de programmation.

Par conséquent, ces dernière ne peuvent pas être utilisé comme broches E /S lors de débogage.

►Un programmateur PIC kit2 : [19]

Le programmateur relie l'ordinateur au circuit électrique (donc au Pic BASIC). Ce programmateur est un PicKit2, il se branche sur le boitier directement à la carte électronique grâce aux 5 fiches qui dépassent et au bout du boitier se trouve un câble USB qu'on branche à l'ordinateur. D'autres modèles plus récents permettent d'autres possibilités tel qu'un débuggeur incorporé ou d'autres sorties pour pouvoir interagir en temps réel avec le dispositif, on pense notamment à l'ICD3.

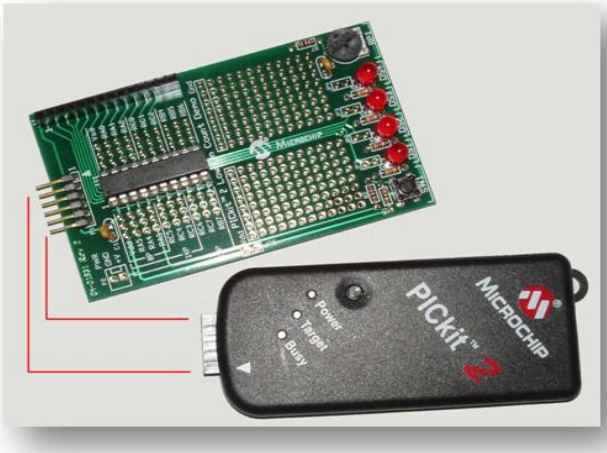

Figure 2.2 : Le programmateur PicKit2 et une carte électronique avec laquelle le programmateur peut interagir.

Figure 2.3 : Le programmateur ICD3 et une carte électronique plus évoluée avec laquelle le programmateur peut interagir.

►Kit MPLAB ICD 2 [20]

Ce kit permet de programmer et de déboguer un microcontrôleur PIC via le logiciel MPLAB IDE. Ce kit est composé :

- D'un module ICD 2 (circulaire) contenant l'électronique de communication,
- D'un câble USB permettant la communication entre le bloc circulaire et le PC,

Un petit câble RJ11 permettant la communication entre le bloc et le PIC.

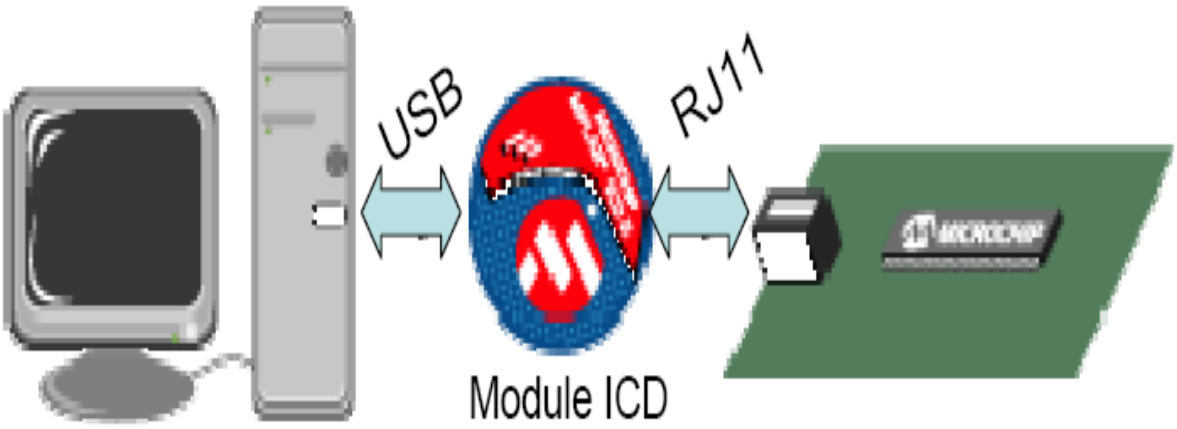

Figure 2.4 : Câblage du kit ICD 2

►Logiciel MPLAB de Microchip [20]

Il s'agit de l'environnement informatique de développement pour PIC fourni par le Constructeur Microchip. Il est gratuitement téléchargeable et utilisable.
MPLAB permet d'éditer le code assembleur (reconnaissance des instructions), compiler le code (le lier à des librairies si besoin), simuler le comportement d'un PIC (ceci permet de tester le bon fonctionnement d'un programme (débogage) par simulation).

2.6 Conclusion :

Dans ce chapitre nous avons présente le microcontrôleur PIC de manière générale et leur utilités et avantages.

On parle aussi sur l'identification et les différentes familles de PIC, puis la méthode de programmation.

On conclue que le choix de langage et de compilateur est important la programmation de PIC qui dépende de l'utilité

Ce nous a permis de nous familiariser avec la programmation de ce type de microcontrôleur et d'exploiter ce composant d'une manière correcte.

Le prochain chapitre on va voir la réalisation de système.

Chapitre III :
Étude et réalisation d'un système pour la mesure de la persistance rétinienne

3.1. Introduction :

Après avoir décrit la partie théorique de notre réalisation
Comme cela était précisé précédemment, ce projet consiste à faire l'étude et la réalisation d'un système permettant le diagnostic de la persistance rétinienne.

L'objectif de ce projet est donc d'utiliser un microcontrôleur pour centraliser les communications et le contrôle de différents matériels électroniques.

Le microcontrôleur est doté d'une faculté de traitement des informations qui lui parviennent et de nombreux moyens de communication ; nous souhaitons donc créer une carte électronique qui servira d'interface de contrôle et de communication entre plusieurs périphériques.

Réaliser une carte électronique à base de microcontrôleur permettant d'interfacer des périphériques tels que des LED et des buttons poussoirs

3.1. Présentation du système :

On cherche à réaliser un système permettant de diagnostiqué la persistance rétinienne et en va développer l'idée de clignotement d'une LED pour différentes fréquences à partir d'un pic microcontrôleur 16F84A et des boutons poussoir qui permet de changer la vitesse de clignotement de la LED.

3.2. Réalisation du système :

Notre système est composé de pic16f84A et une LED rouge, des buttons poussoirs, des résistances, oscillateur et 2 capacités.

3.2.1 Les diodes [10]

a. Description :

Les D.E.L (Diode Electro Luminescente) ou en Anglais : L.E.D (Light Emitting Diode) éclairent lorsqu' elles sont parcourues par un courant de l'anode vers la cathode.

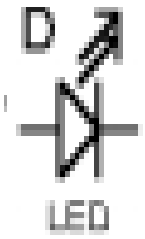

Figure 3.2 : Schéma interne de Diode

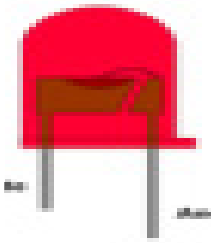

figure 3.3 : Schéma de Diode

b. Utilisations :

L'avantage d'utiliser des leds est qu'elles ne s'usent pas, elles sont moins chères que des voyants, elles consomment moins d'énergie .Mais l'inconvénient et qu'elles ne peuvent Fonctionner qu'avec une faible tension, et qu'elles n'éclairent pas beaucoup par rapport aux ampoules classiques.
La fonction LED gère aussi la persistance rétinienne.

3.2.2 L'oscillateur :

L'horloge système peut être réalisée soit avec un quartz (a), soit avec une horloge extérieure (b), soit avec un circuit RC(c). Dans ce dernier cas, la stabilité du montage est limitée.

La fréquence maximale d'utilisation va dépendre du microcontrôleur utilisé. Le suffixe indiqué sur le boîtier donne la nature de l'horloge à utiliser et sa fréquence maximale. [21]

Nous avons travaillé avec l'horloge interne de PIC donc on a besoin d'utilisé un quartz.

a. Le QUARTZ [10]

Le quartz est composé de silice SiO2, qui est une matière minérale, une fois taillé en fine lamelle. Il présente la particularité d'être piézo-électrique. Généralement incolore on peut le trouver dans la nature mais on l'obtient maintenant surtout par synthèse dans l'industrie.

Figure 3.4 : Schéma de QUARTZ

3.2.2 PIC 16F84A :

On a défini le pic dans le chapitre précédent avec leur identification et ses brochages.

3.4. Partie Programmation

3.4.1 Procédure de programmation :

Programmer les microcontrôleurs: préparations

a) **Ecrire ou modifier le code source**: d'habitude, le code source est écrit avec l'assembleur PIC.

b) **Compiler**: le code que vous avez écrit en langage clair est converti en langage machine. A cette fin nous utilisons le logiciel assembleur PIC de Microchip, MPLAB IDE.

c) **Programmer:** le code machine est programmé dans le processeur via votre PC et le programmateur **RS232** et à l'aide du logiciel **WinPIC 800**

d) **Effacer un appareil PIC:** Effacer le contenu d'un contrôleur

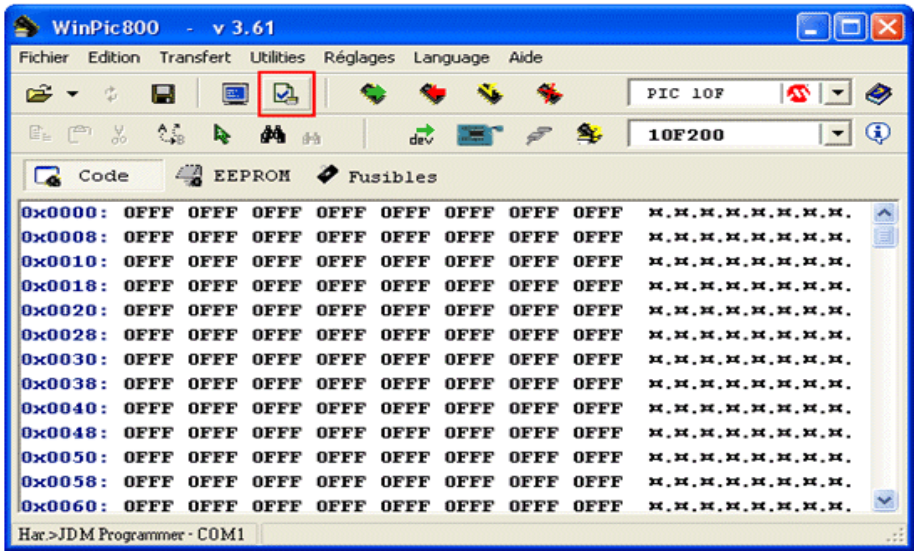

Figure 3.5 : l'interface de logiciel winpic800

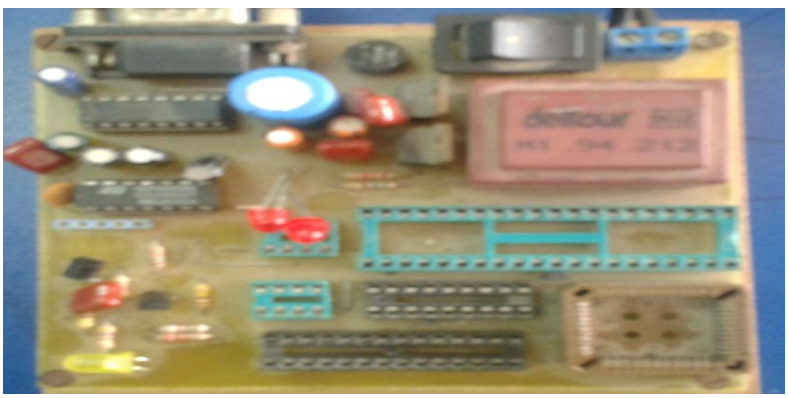

Figure 3.6 : Programmateur RS232

3.4.2 Organigramme

La programmation du circuit concerne d'une part plus importante le microcontrôleur.
L'organigramme du programme est le suivant :

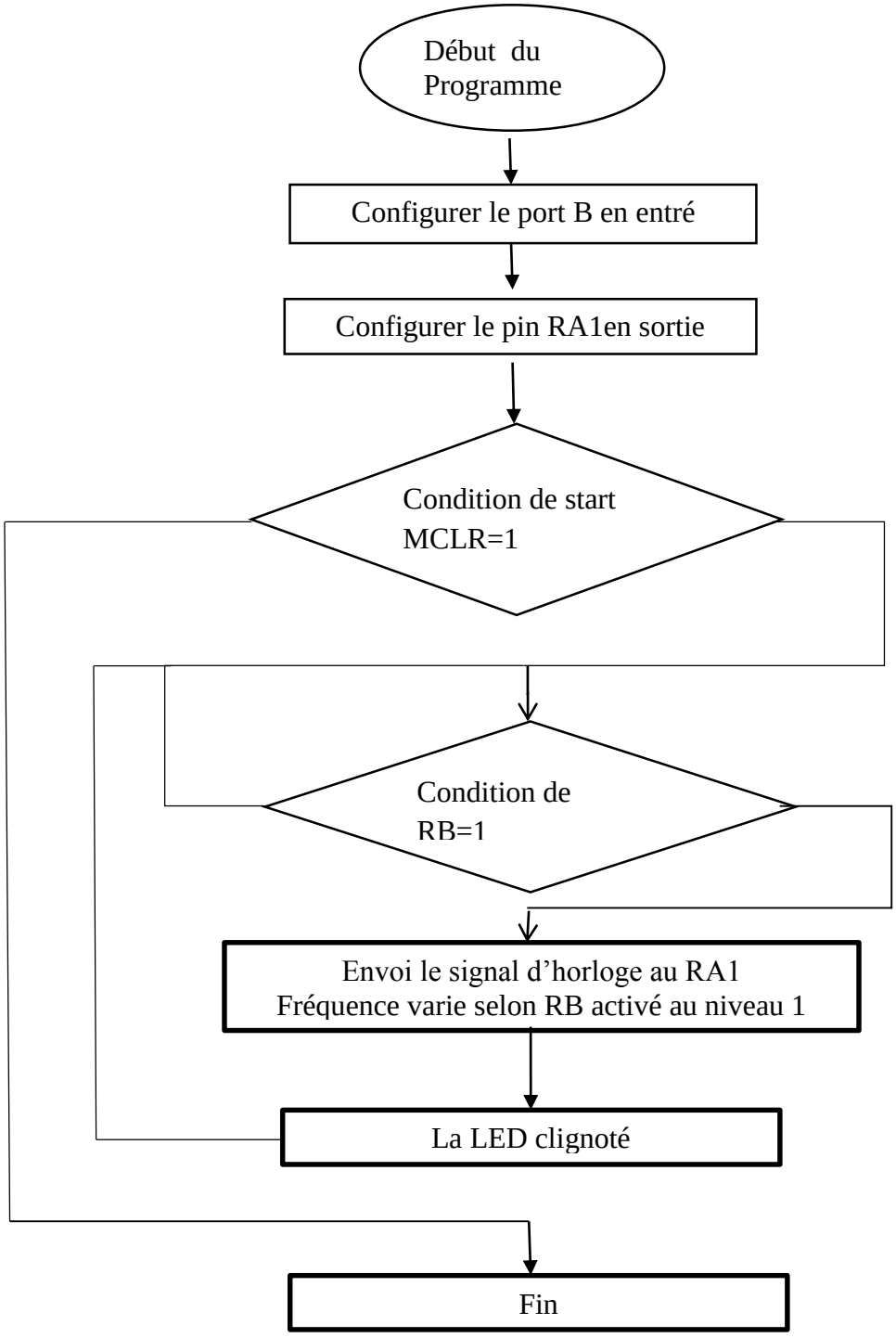

Figure 3.7 : Organigramme général de programmation.

3.4.3 L'explication de notre programme :

Le programme à charger dans le microcontrôleur PIC 16F84A est écrit en langage assembleur (fichier.asm).

Pour programmer le PIC en va convertir ce fichier à un fichier .hex avec le compilateur MPLAB IDE.

Les registre interne de PIC qui nous avons utilisées est :
Le registre (W), Le registre STATUS, Le registre OPTION, Le registre PORTA et PORTB Le registre INTCON, Le registre TMR0, Ces registres sont définis dans le chapitre précédent.

a. L'identification :

Si vous remarquerez sur notre programme sur les premières lignes sont précédées du symbole (;).c'est une zone de commentaires, On utilise cette zone comme une zone d'identification dans laquelle on inscrira : le titre du notre programme (clignotant à LED), la date d'édition (mars 2015), le nom de l'auteur (les étudiants slimani hichem et nekrouf med said), et le langage utilisé (assembleur).

b. Les directives :

►La directive "**list**" :

En utilise La directive "List" en bas de l'entête est destinée au compilateur pour indiquer le type de processeur est utilisé dans notre system (ici : Pic16F84A), donc on écrit :

List p=16F84A

►Les directives "**__config**" :

C'est la ligne "**__config**" destinée au compilateur pour établir les bits de configuration.

__config _CP_OFF & _WDT_OFF & _PWRTE_ON & _HS_OSC

Pour notre programme en choisi l'oscillateur de type HS (haute vitesse : 8Mhz, 10Mhz) quartz ou résonateur céramique ou oscillateur externe a travers le bit 1 ,par l'instruction :

_HS_OSC

Et pour Le bit 2 qui autorise ou non le fonctionnement du watchdog timer ; dans notre programme ne besoin pas d'utiliser le watchdog donc on écrit : _WDT_OFF c'est-à-dire le chien de garde non valider.

A partir de bit 3 on autorise le fonctionnement du power up timer et on écrit :

_PWRTE_ON

Le bit 4 permettent d'interdire la lecture de la mémoire programme.
CP_OFF dans notre programme ne travaille pas par le code de protection.

►La directive « #**include** »

Afin d'éviter l'écriture d'une multitude de lignes d'assignation au début de chaque programme, on peut les regrouper dans un fichier et les appeler par une seule commande :

#include "P16F84.INC" ou #include <P16F84A.INC>

c. Les macros :

Ce sont des sous-programmes, Permet d'indiquer dans quelle banque travaillé on utilise les instructions
bsf : mise à 1 bcf : mise a 0
de fin de macro "endm" (end of macro).

Pour le démarrage sur reset on utilise la directive org pour placé l'initialisation à l'adresse 0x0000 on écrit :
org 0x0000
goto initialisation

d. L'interruption :

Dans notre programme on utilise une source d'interruption : TMR0 en mode timer c'est-à-dire le dépassement du registre timer (TMR0)

Sous-programme :

```
org 0x0004                      ; vecteur d'interruption
        bank0
        comf PORTA , f          ; on inverse le niveau logique de la sortie RA1
                                        ; => inversion de l'état de la LED
        bcf INTCON,T0IF         ; on efface le drapeau T0IF
        movf temp,w
        movwf TMR0              ; retour d'interruption
        retfie
```

Lorsqu'une interruption est provoquée Le programme termine l'instruction en cours Il arrête la procédure qu'il était en train d'exécuter Il passe à l'adresse 0x04 à travers la directive org et utilise aussi l'instruction comf Effectue le complément à 1 du contenu de l'emplacement mémoire spécifié. Dit de façon plus évidente : inverse tous les bits de l'emplacement (on inverse l'état de la LED).
Et utilise l'instruction bcf pour mise a 0 le bit 2(effacé le drapeau).
On utilise l'instruction' movf temp,w' pour charger la valeur ' temp' dans le registre w,puis on chargée dans le registre tmr0 par l'instruction 'movwf TMR0'
Les interruptions sont remises en service automatiquement lors du retour de l'interruption par l'instruction « retfie ».

e. L'initialisation :

On charge la valeur 199 dans le registre option qui correspond a la valeur 11000111 en binaire.
b7 : RBPU =1 : (actif niveau haut) place cette pin à la masse si on le presse.
résistances de rappel hors service (on n'en n'a pas besoin),

b6 : INTEDG=1 : Si b6 = 1, on a interruption si le niveau sur RB0 passe de 0 vers 1.

b5 : TOCS=0 : ce bit détermine simplement le fonctionnement du timer0, et ça dire que le timer0 est incrémenté en fonction de l'horloge interne (synchronisé au programme).

b4 : TOSE=1 : Comme nous avons placé b5=0, b4 est alors inutilisé. Nous laisserons donc b4 = 0.

b3 : PSA=0 : il effectue une prédivision au niveau du timer0 (tmr0) car nous travaillons avec timer0

b2, b1, b0 : PS2, PS1, PS0=1 : Ces trois bits déterminent la valeur de prédivision pour le registre. Montrées dans le petit tableau précédent (tableau 2).

Nous prenons la plus grande valeur disponible, soit **256**, nous aurons donc une interruption toutes les (256*256) = **65536 temps de cycle** = 65536µs=65,536ms (à 4Mhz) (prescaler 1:256).

Remarque : Notez que le temps de débordement du timer dépend à la fois du nombre de cycles comptés.

Le débordement du timer (de1jusque 256) nécessite un temps de 65,536ms.
On calcule le temps nécessaire pour faire une incrémentation dans le registre timer0 à partir la relation suivant :
65,536 /256=0,22ms et a partir de cette valeur on remplies le tableau suivant :

Période T(ms)	**Demi-période T/2(ms)**	**Fréquence(Hz)**	**Nombre d'incrémentation**	**La valeur temp** **temp=256- Nombre d'incrémentation**
40	20	5	90,90	156
60	30	3.33	136,36	120
70	35	2.85	159,09	97
80	40	2.5	181,81	74
90	45	2.22	204,54	51
100	50	2	227,27	29

105	52.5	1.90	238,63	17
110	55	1.81	250	6

Tableau 3.2 : les valeurs temp correspond aux périodes interruption

Sous-programme :

```
bank1
movlw .199
movwf OPTION_REG    : (OPTION_REG) = (11000111)
```

Puis nous utilise bcf TRISA, 1 pour la configuration de la pin RA1 en sortie c'est à dire la commande de la LED.
Et la valeur de 199 en décimale correspond a la valeur (11000111) en binaire le contenue de registre option.

f. Le programme principal :

Dans le programme principale nous utilise l'instruction btfsc pour tester les bits du PortB et saut si il vaut 0 car dans notre circuit place les masses avec les pins de portb.
Et les boutons poussoir relie avec l'alimentation.
Et si vous changer dans notre circuit et relie l'alimentation directe avec les pins de portb
Donc on va changer l'instruction et utilise l'instruction BTFSS qui dit que Teste le bit précisé du contenu de l'emplacement F et saute s'il vaut 1.

Puis utilise l'instruction goto pour effectue ce qu'on appelle un saut inconditionnel aux les sous-programmes temp_0, temp_1……etc.

Les sous-programmes temp_0, temp_1……etc. qui contient les valeurs correspond a les périodes qui utilisé dans notre programme pour clignoté la LED et qui définit dans le tableau précédent (Tableau3.2).

Exemple: temp_0

```
movlw .165
movwf temp
goto activation
```

Dans ce sous-programme on charger la valeur 165 dans le registre W (registre de travail).puis utilise l'instruction movwf qui Permet de sauvegarder le contenu (165) du registre de travail W dans un emplacement mémoire (temp).
Aussi utilise L'étiquette goto activation pour le passage direct a le sous-programme activation.

Sous-programme activation :

```
activation
        bank0
    movf temp,w
```

```
movwf TMR0
movlw B'10100000'
movwf INTCON
```

Dans ce sous-programme on charge la valeur temp dans le registre TMR0 pour commencer l'incrémentation à partir de cette valeur jusqu'à 256 pour faire une interruption. puis on charge la valeur B'10100000' dans le registre de travaille W et sauvegarder dans le registre INTCON

b7 : GIE=1 : pour valider toutes les interruptions d'une seule fois.
b6 : EEIE=0 : pour valider l'interruption de fin d'écriture en eeprom.
b5 : T0IE=1 : pour Valide l'interruption générée par le débordement du timer0
b4 : INTE=0 : ne travaille pas par RB0 comme une source d'interruption.
b3 : RBIE=0 : non Valide les interruptions si on a changement de niveau sur une des entrées RB4 à RB7.
b2 : T0IF=0 : initialiser le flag**.**
b1 : INTF=0 :n'utilise pas
b0 : RBIF=0 :n'utilise pas.

Remarque :
MCLR :
La broche MCLR permet de réaliser un Reset du circuit quand elle est placée à 0V

3.5 Partie Simulation

3.5.1 Présentation d'Isis :

Avant de passer à la réalisation pratique de notre système nous avons eu recours à la simulation de notre système.

Pour cela on utilisé le logiciel ISIS qui permet simuler le fonctionnement des microcontrôleurs PIC avec tous les périphériques de la carte.

L'ISIS est un logiciel professionnel, utilisé dans l'électronique pour simuler des circuits et créer des typons. Il est également capable de simuler le fonctionnement du PIC avec tous les périphériques de la carte de commande.

En effet, nous avons utilisé ce logiciel afin de mieux visualiser le bon déroulement du système ainsi que d'avoir une idée claire sur la partie matérielle et la conception des circuits imprimés. Il nous permet de limiter les essais réels.

La figure suivante (figure 3.7) présentera un imprime écran de la dernière étape de la simulation de notre carte électronique en utilisant le logiciel ISIS.

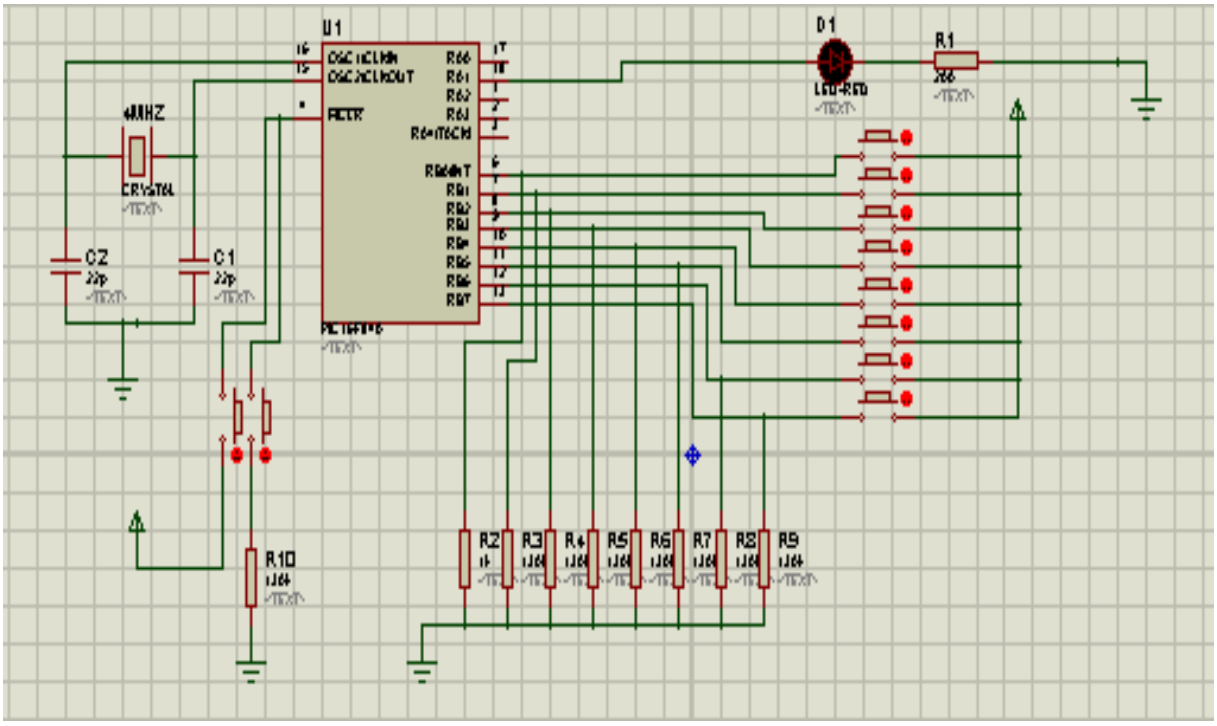

Figure 3.8 : schéma de simulation de la carte électronique en ISIS

Circuit imprimé :

A partir de logiciel ARES on fait le dessin de notre circuit pour imprimer sur une carte électronique

- **ARES :**

C'est un logiciel permettant le routage des cartes électroniques en mode automatique ou manuel. Il est possible d'utiliser ARES sans avoir créé au préalable un schéma dans ISIS. Cette fonctionnalité permet de réaliser des circuits de faible complexité en plaçant les composants et en traçant les pistes directement sur ARES. Une fois les connections établies, il est possible d'effectuer un routage automatique des pistes.

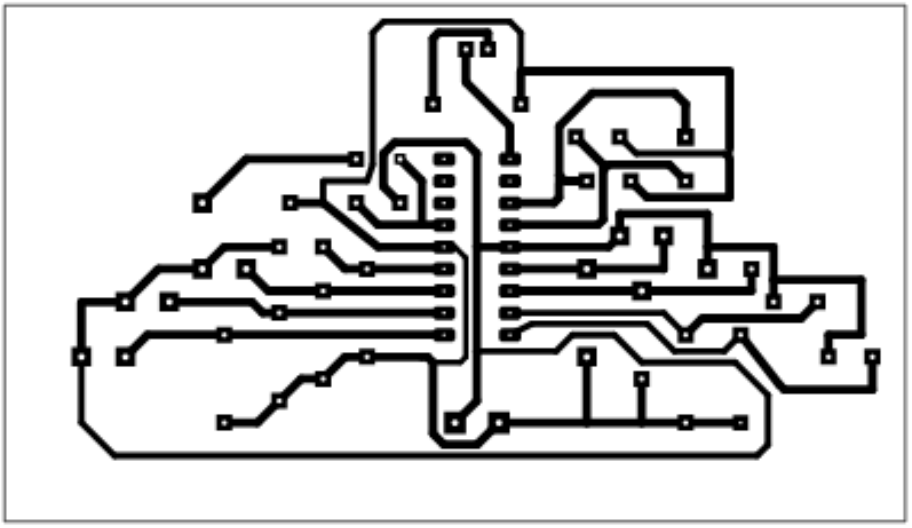

Figure 3.8 : Circuit imprimé coté cuivre

3.6 Réalisation sur plaque d'essai :

On va réaliser notre système sur une plaque d'essai et tester par le bouton poussoir Le clignotement de la LED a différents fréquences, et on remarque qu'on a une variation de la vitesse de clignotement et après cette étape on va passer à la réalisation de circuit imprimé.

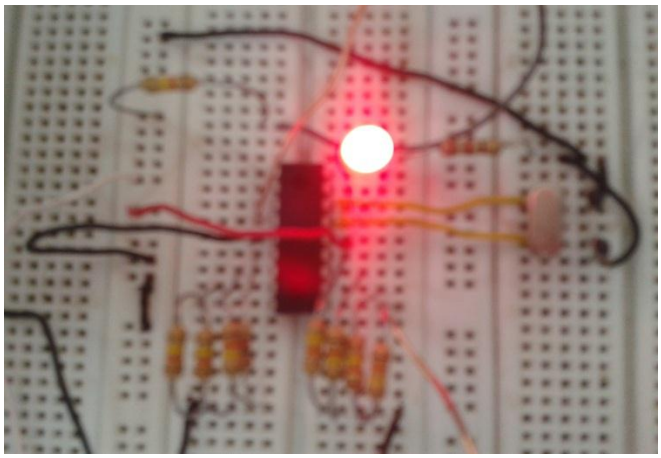

Figure 3.9 : présentent l'implantation des composants sur la plaque d'essai

3.7 Problèmes rencontrés et leurs solutions :

Lors de la réalisation de notre projet, nous avons rencontré plusieurs problèmes, dont les principaux sont :

- le packaging de bouton poussoir n'existe pas sur logiciel ARES pour faire le dessin de circuit c'est pour ça en remplaçant par un composant de même dimension (capacité).
- La programmation du notre PIC 16F84A par l'assembleur, poser quelques difficultés, que nous avons résoudre avec le temps et quelques recherches.
- Certains composants n'étaient pas disponibles dans le magasin de la faculté, par Exemple les buttons poussoir, de ce fait nous les avons acheté dans le Commerce.
- Pour cette raison dans une suite de ce projet, il serait intéressant de réaliser l'isolation par une boite de matière isolant.
- Lors de la réalisation des circuits imprimés, nous avons obtenues des cartes avec des ruptures ou des fissures de piste, et par conséquent nous avons été dans l'obligation de les refaire plusieurs fois.
- Le choix des composants électroniques nécessaire à la réalisation de notre système nous a aussi causé quelques tracas, surtout en ce qui concerne les résistances, la LED.

Nos choix se sont basés essentiellement sur la disponibilité du matériel dans le magasin.

3.8 Conclusion :

Au cours de ce chapitre, nous avons décrit toutes les étapes nécessaires de l'implémentation de notre système y compris la présentation de l'environnement logiciel, la réalisation de la carte électronique et Comme nous l'avons mentionné, notre système offre à l'utilisateur d étudie et mesuré la persistance rétinienne.

Nous avons traité dans ce chapitre l'aspect matériel de notre projet. Aussi le circuit utilisé ont été présentés et étudié.

Pour conclure, les difficultés ont été surmonté, mais nous demandé du temps et quelques Inquiétudes. Mais cela nous a permis de nous confronter à la réalité de la pratique.

Chapitre IV:
Application pratique

4.1 Introduction :

Le chapitre précédent concerne la phase finale dans la réalisation du prototype de ce projet. Nous y abordons les difficultés que nous avons rencontrées.

Nous y présentons également quelques résultats expérimentaux afin de valider les performances du notre system réalisé.

Notre système permet à l'ophtalmologue de faire des tests subjective pour diagnostiquer l'œil et découvrir est que le patient a une maladie de palinopsie ou non.

L'idée de base de cette examen ophtalmologique est de connaitre à quelle fréquence le patient va distinguer entre les images successive ce qu'on appelle test de persistance rétinienne.

Lorsque le personne subit une lumière intense (soleil, les phares des voitures ……etc.), une tâche sera sauvegarder sur la rétine pendant une durée.

L'état normale pour la disparition de cette image sauvegarder sur la rétine prendre un temps de 50 ms.

4.2 Le choix des fréquences :

Nous avons basé sur le cas normal de la persistance rétinienne de duré de 50 ms comme nous avons définies dans le premier chapitre.

On choisit des fréquences inférieur et supérieur à ce seuil pour mesurer la persistance rétinienne et pour étudier les cas normales et les cas pathologies (la palinopsie), de façons que si on appuyé sur un bouton poussoir la vitesse de clignotement va changer selon la fréquence correspondante.

On à 8 boutons dans notre système c'est-à-dire on a 8 fréquences telles :

5(Hz) , 3.33(Hz) , 2.85(Hz) , 2.5 (Hz) , 2.22(Hz) , 2(Hz) , 1.90(Hz) , 1.81(Hz) .

4.3 Explorations fonctionnelles subjectives :

Ces explorations nécessitent la participation volontaire de sujet, elles supposent donc la capacité de compréhension de la tache demandé et la collaboration de bonne foi du sujet. [22]

Le teste est fondé sur une question principal est ce qu'il y'a un clignotement de la LED ? (avec un changement de fréquence volontaire, destiner pour trouver à quelle fréquence le patient n'arrive pas à faire la différence entre une diode allumé ou clignotante)

4.4 Le déroulement de l'examen

Pour faire l'examen de la persistance rétinienne il faut que le patient soit en position face au prototype avec des yeux dirigé vers la LED qui clignote a des fréquences différentes.

Pendant le test il faut que le patient repose ses yeux une durée entre une fréquence et une autre pour avoir des résultats fiable et ne pas influencé la suite de l'examen.

Le changent de la vitesse de clignotement est fait à partir des boutons poussoirs, chaque bouton poussoir correspond à une fréquence bien déterminer. Et c'est le patient qui indique la fréquence dont il ne distingue pas le clignotement de la LED. Cette mesure nécessite une bonne coopération du patient.

4.5 Résultat des examens :

Les personnes	**sexe**	**L'âge**	**Maladie actuelle**	**acuité visuelle: l'œil droit**	**acuité visuelle : l'œil gauche**	**La fréquence de distinction (HZ)**
Sujet 1	homme	Adulte	Cas normale	10 /10	10 /10	2
Sujet 2	homme	Adulte	Cas normale	10 /10	10 /10	2.22
Sujet 3	homme	Adulte	Cas normale	10 /10	10 /10	2.22
Sujet 4	homme	Adulte	Cas pathologie (Myopie)	6/10	7/10	2.22
Sujet 5	homme	Adulte	Cas pathologie (Myopie)	7/10	9/10	2.22
Sujet 6	homme	Adulte	Cas normale	10 /10	10 /10	2 .5
Sujet 7	homme	Adulte	Cas pathologie (Myopie)	7/10	8/10	2

Sujet 8	homme	adulte	Cas pathologie (hypermétropie)	9/10	4/10	1.9
Sujet 9	femme	enfant	Cas normale	10/10	10/10	2,5
Sujet 10	femme	adulte	Cas normale	10/10	10/10	2
Sujet 11	femme	enfant	Cas normale	10/10	10/10	1,81
Sujet 12	homme	enfant	Cas normale	10/10	10/10	1,81
Sujet 13	homme	enfant	Cas normale	10/10	10/10	2,8
Sujet 14	homme	adulte	Cas normale	10/10	10/10	1,81
Sujet 15	homme	adulte	Cas normale	10/10	10/10	2,22
Sujet 16	homme	adulte	Cas normale	10/10	10/10	5

Tableau 4.1 : les résultats des tests sur les différentes personnes

4.6 Interprétation des résultats:

Comme nous dit précédemment la persistance rétinienne positive a un seuil de 50ms c'est-à-dire le temps nécessaire pour effacé l'image qui sauvegardé sur la rétine, correspond à une fréquence de 2Hz.

D'après le tableau (Tableau 4.1) :

Les sujets (2…5) a une fréquence de distinction 2 ,22Hz entre les images correspond au 45ms c'est le temps nécessaire pour l'effacement de premier image et l'enregistrement de deuxième image, 45ms<50ms (le seuil) alors la possibilité de la maladie de persistance rétinienne (palinopsie) est réduite.

Les sujets (1 ,7et 10) a une fréquence de 2 Hz correspond au temps nécessaire pour la distinction entre les images successives de 50ms= seuil, c'est le cas normal.

Les sujets (11,12et 14) a une fréquence de 1,81Hz corespond au temps de 55ms > (le seuil) c'est à dire le facteur de maladie est augment.

Comme interprétation les résultats inférieurs à ce seuil signifié que la possibilité de la pathologie de persistance rétinienne (palinopsie) augmentent et le contraire signifié que la possibilité de la maladie est réduite.

D'après les résultats obtenus :

On constate que la persistance rétinienne ne varie pas en fréquence selon le sexe et l'âge.

On conclue que la mesure de persistance rétinienne ne déponde pas aux pathologies de l'œil mais dépend de longtemps d'enregistrement de l'image par le cerveau et la faible vitesse d'effacement.

4.1 Conclusion :

La persistance rétinienne est un phénomène normal, seulement son intensité varie suivant les individus et les objets regardés.

On sait que quand on regarde un objet lumineux puis que l'on ferme les yeux, son image ne s'efface pas immédiatement. Elle persiste encore pendant un intervalle qui varie selon la puissance d'éclairement du sujet observé. C'est le phénomène de persistance rétinienne.

En effet les cellules sensibles de la rétine ne sont pas réceptives à une nouvelle image avant 1/10 ème de seconde, il leur faut ce temps pour se régénérer.
Le cinéma par exemple, consiste à se faire succéder des images défilant à une vitesse telle que l'on a une impression de mouvement continu.

Lorsque l'image se forme sur ta rétine, elle ne disparaît pas immédiatement mais reste « imprimée » une dixième de seconde. Tu te souviens que la rétine est constituée de milliards de cellules. Eh bien, certaines de ces cellules sont sensibles à la luminosité, tandis que d'autres sont sensibles aux couleurs primaires : le rouge, le vert et le bleu.

Conclusion générale

Il est évident que l'interdépendance entre la médecine et l'électronique est vitale pour l'être humaine, ce qui a engendré l'émergence d'un nombre incontournable d'appareillage médicale.

Le développent de la mesure incite à enrichir encore plus les techniques liées à son développement au fin de rehausser son exploitation. Les données physiologiques et informatiques ont été mieux décryptées grâce à la réalisation pratique de system d'acquisition au profit de médecin pour mieux diagnostiquer les maladies.

Le but principal de notre travail concernait l'étude et la réalisation pratique d'un Système de clignotement de LED à différent fréquences (vitesse) à base d'un microcontrôleur PIC 16f84A.

Ce système devra permettre d'étudié et mesuré la persistance rétinienne et découvrir la maladie de la palinopsie par exemple si elle existe. C'est un système qui peut être très utile pour les ophtalmologues qui veulent investir plus dans leur diagnostic.

Finalement, ce projet de fin d'étude nous a permis de nous enrichir dans les domaines de l'électronique, de l'informatique et de la programmation des microcontrôleurs en découvrant des logiciels tels que "PROTEUS" (ISIS et ARES), MPLAB. Il nous a permis aussi de comprendre et découvrir un phénomène optique très intéressant par sont comportement qui change d'une personne à une autre. Un changement qui peut être révélateur de maladies.

Bibliographie

[1] : Sébastien THON, Le système visuel humain ,2ème année 2014-2015.

[2] : Cours Anatomie & Physiologie CNFSOC/ISOs, Page 13.

[3] : Groupe scolaire Les Clarines – BERNEX -, Images in ‘école, 2011-2012.

[4] : Réalisé en collaboration avec des professionnels de la santé et de la médecine, sous la direction du docteur Pierrick HORDE, sante-medecine. commentcamarche.net.

[5] : Astrid Maury, Le folioscope ou l'expérience magique du dessin animé.

[6] : Didier LUTZ, CPD Arts Visuels de l’Allier, Octobre 2014.

[7] : Dossier pédagogique réalisé par la section Régendat Sciences Naturelles, Et si tu te trompais…, Mars 2006.

[8] : Christian. Vinent, le cinéma d’animation, école primaire, 2008, page 5.

[9] : InformationHospitaliere.com - wikipedia.org.

[10] : Wissem HENI et Imen Hmaied, MÉMOIRE DU PROJET DE FIN D’ÉTUDES Pour l’obtention de la Licence Appliquée en Sciences et Techniques de l’Information et de Communication Mise en place d’une plateforme de télécommande des équipements électrique à distance « Smart House », UNIVERSITÉ VIRTUELLE DE TUNIS, 2010-2011, page 21, page 33, page 76.

[11] : La programmation des PICs. BIGONOFF première partie Révision36.

[12] : V.Tourtchine, Microcontrôleur de la famille PIC. Support de cours & Prise en main du logiciel MPLAB, 2009.

[13] : Unisciel/K.Zampieri, Types et langages de programmation Algorithmique et Programmation.

[14] : BERNARD BÉGHYN HERMES-SCIENCE, Les microcontrôleurs PIC; Edition: 2003.

[15] : PASCAL MAYEUX DUNOD, Apprendre la programmation des PICs. Edition : 2002.

[16] : BIGONOFF Révision 6, La programmation des PICs. Edition : 2001.

[17] : M. Le Gonidec, INTRODUCTION à LA PROGRAMMATION – PRATIQUE DU LANGAGE C –, page 9.

[18] : Elham Firouzi, Les microcontrôleurs dans les systèmes embarqués.

[19] : Clément Tessier et Raphaël Roger, Parcours des écoles d'ingénieurs Polytech Mini-projet S3 Programmation pour des cubes de LED, 2010-2011.

[20] : Présenté par Thomas Grenier, Dominique Tournier, Olivier Bernard, David Lévèque. , TP MICROCONTROLEUR, FAMILLE PIC Les interruptions.

[21] : BRAHIMI Abdelkrim et GUEZOULI Housseyne , Mémoire de Master Etude et réalisation d'une carte de commande à base d'un microcontrôleur PIC 16f877 pour ponts redresseurs triphasés à thyristors, Département de Génie Electrique et Electronique, Filière : Electrotechnique, 20 octobre 2014, page32.

[22] : BREKCI .F, cours exploration fonctionnelle, 2008, page 1.

Annexes

Annexe 01

Programme assembleur :

```
; "Circuit de test : clignotant à LED"
; (C) slimani hichem et nekrouf med said, mars 2015
; email hichemsli72@gmail.com
; version 1.01
; microcontrôleur PIC 16F84A
;langage :assembleur
; développé avec Microchip MPLAB IDE
        List p=16F84A ; processeur utilisé
        #include <p16F84A.inc>
        __config _CP_OFF & _WDT_OFF & _PWRTE_ON & _RC_OSC
                ;bits de configuration :
                ;code protect OFF
                ;watchdog timer OFF
                ;power up timer ON
                ;oscillateur RC
;xxxxxx
; macro
;xxxxxx
bank1  macro                    ; passage en banque 1
                bsf STATUS,RP0
                endm
bank0  macro                    ; passage en banque 0
                bcf STATUS,RP0
```

```
            endm
        CBLOCK 0x00C                    ; début de la zone variables
        temp : 1                        ; compteur de temps
        ENDC                            ; Fin de la zone
;xxxxxxxxxxxxxxxxxxx
; démarrage sur reset
;xxxxxxxxxxxxxxxxxxx
        org 0x0000
        goto initialisation
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; Routine d'interruption
; 1 source d'interruption : TMR0 en mode timer
; Toutes les 256*256 = 65536 cycles (prescaler 1:256)
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        org 0x0004                  ; vecteur d'interruption
        bank0
        comf PORTA , f              ; on inverse le niveau logique de la sortie RA1
                                        ; => inversion de l'état de la LED
        bcf INTCON,T0IF             ; on efface le drapeau T0IF
        movf temp,w
        movwf TMR0                      ; retour d'interruption
        retfie
;xxxxxxxxxxxxxxx
; initialisation
;xxxxxxxxxxxxxxx
initialisation
```

```
        bank1
    movlw .199
         movwf OPTION_REG
        ; bit 5 (T0CS) = 0 : TMR0 en mode timer
        ; (OPTION_REG) = (11000111)
        ; prescaler 1:256
        bcf TRISA , 1
        ; bit 1 du port A (RA1) = 0 : configuration en sortie (commande de la LED)
        ; (TRISA) = (---11101)
        movlw B'00000000'
        movwf INTCON
;xxxxxxxxxxxxxxxxxxxx
; programme principal
;xxxxxxxxxxxxxxxxxxxx
debut_programme
        ; on attend le débordement de TMR0 (0xFF -> 0x00)
        ; ce qui génére une interruption
        bank0
                btfsc PORTB , 0                 ; tester RB0, sauter si vaut 0
                goto temp_0
                btfsc PORTB , 1                 ; tester RB1, sauter si vaut 0
      goto temp_1
                btfsc PORTB , 2                 ; tester RB2, sauter si vaut 0
      goto temp_2
                btfsc PORTB , 3                 ; tester RB3, sauter si vaut 0
      goto temp_3
```

```
        btfsc PORTB , 4                 ; tester RB4, sauter si vaut 0
        goto temp_4
        btfsc PORTB , 5                 ; tester RB5, sauter si vaut 0
        goto temp_5
        btfsc PORTB , 6                 ; tester RB6, sauter si vaut 0
        goto temp_6
        btfsc PORTB , 7                 ; tester RB7, sauter si vaut 0
        goto temp_7
           goto debut_programme
temp_0
                movlw .165 ;
                movwf temp
             goto activation
temp_1
                movlw .120 ;
                movwf temp
                goto activation
temp_2
                movlw .97 ;
                movwf temp
                goto activation
temp_3
                movlw .74 ;
                movwf temp
                goto activation
temp_4
```

```
                movlw .51 ;
                movwf temp
                goto activation
temp_5
                movlw .29 ;
                movwf temp
                goto activation
temp_6
                movlw .17 ;
                movwf temp
                goto activation
temp_7
                movlw .6 ;
                movwf temp
                goto activation
activation
                bank0
        movf temp,w
        movwf TMR0
        movlw B'10100000'
        movwf INTCON
        ; bit 7 (GIE) = 1 : autorisation globale des interruptions
        ; bit 5 (T0IE) = 1 : autorisation de l'interruption TMR0
        ; bit 2 (T0IF)= 0 : on efface le drapeau de l'interruption TMR0
        goto debut_programme
        END
```