الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la recherche scientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Electronique
Option : Micro-informatique et Instrumentation

Réf: …/2010

Mémoire de Fin d'Etudes
En vue d'obtention de diplôme: Master en électronique

Thème

ETUDE ET REALISATION D'UN PROGRAMMATEUR DE CARTE A PUCE ET SON LECTEUR

Présentée par :
SELMI SYFEDDINE
SEGHIR SALAH

Proposé et Dirigé par :
Mr. ABDESSELAM SALIM

PROMOTION 2009/2010

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la recherche scientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Electronique
Option : Micro-informatique et Instrumentation

Mémoire de Fin d'Etudes
En vue d'obtention de diplôme:
Master en électronique

Thème :

ETUDE ET REALISATION D'UN PROGRAMMATEUR DE CARTE A PUCE ET SON LECTEUR

Présenté par :

SELMI SYFEDDINE

SEGHIR SALAH

Avis favorable du président du Jury : **Avis favorable de l'encadreur :**

Nom et prénom **Mr. ABDESSELAM SALIM.**

.....

.....

Signature

Cachet et signature

Résumé du mémoire de fin d'étude
En vue d'obtention de diplôme:
Master en électronique
Option : Micro-informatique et Instrumentation

Thème :

ETUDE ET REALISATION D'UN PROGRAMMATEUR DE CARTE A PUCE ET SON LECTEUR

Proposé par : **Mr. ABDESSELAM SALIM**

Dirigé par : **Mr. ABDESSELAM SALIM**

RESUME

La carte à puce est un nouveau « sésame » dans le monde des électroniciens, elle intervient dans plusieurs applications sensibles tel que les cartes bancaires, carte santé, carte Télévision à péage et le contrôle d'accès sécurisé.

On conclut dans ce mémoire l'étude et la réalisation d'un programmateur de carte à puce et son lecteur.

Remerciement

Nous remercions Allah le tout puissant de nous avoir donné le courage, la volonté, la santé et surtout la patience, pour pouvoir, durant toutes ces longues années d'études, d'arriver la ou nous sommes aujourd'hui et d'exploiter tous nos efforts pour ce modeste travail.

Nous voudrions d'abord exprimer notre profonde reconnaissance à Mer ABDESSELAM.S notre encadreur de thèse pour nous avoir encadré durant cette année, il a guidée efficacement nos travaux. Nous le remercions pour sa disponibilité et pour les conservations régulières sur ce travail, ses conseils scientifiques et techniques.

Aussi nous remercions les employeurs de laboratoire d'électronique surtout :

Mer, HAMZA.

Nous également remercions les membres du jury.

Finalement nous remercions toutes les personnes, qui de prés ou de loin, par leur compréhension leur coopération nous ont facilités la tâche et ont contribués à la mise en forme du thème.

Dédicace

Je dédie ce travail qui est complété par l'aide de Dieu.

La prunelle de nos yeux, ceux qui nous éclairés le chemin de la réussite à nos parents que dieu les garde.

Je tiens aussi à dédier ce travail

A mes sœurs et mon frère Salem pour leur soutien.

A mes chers amis : Riad, Belkacem chikh, Okba, Houari, Fares, Fethi remli, Feth Zidane, Amor baa, Zohir benziadi, Lazhar, Abdelhakim, Amine chalhoub, chikh sofienne, Abdelhakim Amri, Isam, Mostapha, et hachouf ammar.

A tous mes collègues de sciences et techniques surtout d'option de

Micro-informatique et Instrumentation.

A toute personne de qui m'aide loin ou un autre.

Enfin, avec toute ma gratitude, je le dédie à la plus tendre et merveilleuse mère de tout les algériens et algériennes : Notre chère

Algérie.

Salah.

Dédicace

Je dédie ce travail qui est complété par l'aide de Dieu.

La prunelle de nos yeux, ceux qui nous éclairés le chemin de la réussite à nos parents que dieu les garde.

Je tiens aussi à dédier ce travail

A mes sœurs et mes frères Ali, Chaanbane, Elhedi et Abd elrahim pour leur soutien.

A mes chers Amis: salaheddine, lakhder, Mohamed, khaled, hichem, imadeddine, dinar, Adel.

Fethi, Yazide, fika, walid, Wail, Josef, ramzi, hatem, sohaib, fares, kameleddine

Choiki, bd torki, hassane, islame, issa, Nessereddine, Samire, Soufyane, toufa.

A tous mes collègues de électronique surtout d'option de micro-informatique et instrumentation et a tous les étudiants de la promotion. 2^{eme} *Master des autres spécialités.*

A toute personne de qui m'aide loin ou un autre.

Enfin, avec toute ma gratitude, je le dédie à la plus tendre et merveilleuse mère de tout les algériens et algériennes : Notre chère

Algérie.

Seyfeddine.

SOMMAIRE

CHAPITRE II : Les microcontrôleurs (PIC) et mémoires.

ETUDE EXPERIMENTALE

CHAPITRE III : Réalisation du programmateur carte à puce (Gold et Silver) et son lecteur.

PRTIE I : Réalisation de Programmateur.

PRTIE II : Réalisation de Lecteur.

Liste des figures

Résumé

La carte à puce est un nouveau « sésame » dans le monde des électroniciens, elle intervient dans plusieurs applications sensibles tel que les cartes bancaires, carte santé, carte Télévision à péage et le contrôle d'accès sécurisé.

On conclut dans ce mémoire l'étude et la réalisation d'un programmateur de carte à puce et son lecteur.

Mot clés :

Carte à puce, Programmateur, Lecteur, Microcontrôleur, Pic.

SUMMARY

The smart card its new « sesame » in the world of electronics, she entre in a lot of applications sensible such as banks cards ,television cards in toll reod and the secures access control .

It concludes in this memoir the study and realization of the smart card programmer and his reader.

Keywords:

Smart card, Programmer, Reader, Microcontroller, Pic.

الخلاصة

البطاقة الذكية عبارة عن مفتاح جديد في عالم الالكترونيات ، تدخل في عدة استعمالات حساسة على سبيل الذكر البطاقات البنكية ، البطاقات الصحية ، البطاقات التلفزيونية ذات الدفع المسبق و مراقبة الدخول المؤمن.

ونلخص في هذه المذكرة دراسة و انجاز كل من مبرمج آلبطاقات الذكية و قارئها.

الكلمات الرئيسية:

البطاقة الذكية، المبرمج، القارئ، ميكروكنترولر، بيك.

INTRODUCTION GENERALE

Pendant la dernière décennie, la carte à puce est en train d'occuper une place de plus en plus importante dans notre société. Même si les gens ne sont pas conscients de cela, chaque fois qu'ils paient leur courses dans des grandes surfaces, achètent leur repas ou retirent de l'argent, ils utilisent une carte à puce.

Cette invention à pointe de la technologie est bien sur présente dans plusieurs domaines, dont le plus connu est celui de la téléphonie mobile – toutes les cartes SIM sont en faite des cartes à puce.

Les cartes à puce fournissent des moyens d'effectuer des applications d'une manière flexible, bloquée, standard, avec l'intervention humaine minimale.

Pour mieux comprendre comment la carte à puce est entrée dans notre quotidien on se doit de rappeler un peu le contexte dans lequel elle est apparue. En 1968, deux Allemands Jürgen Dethloff et Helmut Grötrupp introduisent un circuit intégré dans une carte plastique. Ce n'est juste que quatre ans après que le français Roland Moreno, le père de la carte à puce dépose 47 brevets dans 11 pays. La large étendue de ce brevet nous montre que la carte à puce était considérée comme une invention qui pouvait avoir du succès.

Elle commence à être utilisée pour la première fois en 1983, lors de l'apparition des premières cartes téléphoniques à mémoire. Mais sans-doute le plus grand impact de la carte à puce est survenu lors de l'introduction de la carte bancaire (1987), qui a commencée à être largement utilisée en France d'une manière intensive lors de la dernière décennie.

Nous allons étudier et réaliser un programmateur de carte à puce (Wafer Gold et Silver) et son lecteur. Le but est d'arriver à une conclusion qui à partir d'une étude et réalisation, nous donne un aperçu de programmation (écriture) et lecture de carte à puce.

Pour permettre de bien comprendre l'environnement de programmateur et lecteur des cartes à puces on commencera dans le premier chapitre par l'étude des grandes lignes d'architecture technique de ces dernières.

Dans le deuxième chapitre, nous donnerons les caractéristiques des microcontrôleurs PIC 16F84 et 16F876 qui sont l'essentiel dans notre programmateur.

La deuxième partie : c'est la partie expérimentale.

Dans le premier chapitre on exposera l'étude et la méthode de réalisation de notre programmateur carte à puce (GOLD et SILVER).

Dans le deuxième chapitre, étude et méthode de réalisation de notre lecteur carte à puce.

Enfin la conclusion générale et perspectives pour notre étude et réalisation.

Chapitre I

Les cartes à puce

I.1. Introduction :

La carte à puce est un nouveau « sésame » dans le monde des électroniciens, elle intervient dans plusieurs applications sensibles tel que : les cartes bancaires, carte santé, carte télévision à péage et le contrôle d'accès sécurisé.

I.2. Généralités :

I.2.1. Définition de la carte à puce :

La carte à puce (Smart Card en anglais), est une carte plastifiée aux dimensions 85,6×54 millimètres avec une épaisseur de l'ordre de 70 à 90 centièmes de millimètres. Elle comporte au moins un circuit intégré (la puce). La carte est un support électronique mobile et fiable pour conserver des secrets. Ces secrets peuvent être des clefs cryptographiques servant à coder ou décoder des messages confidentiels. **[1]**

I.2.2. Classification des cartes à puces :

La classification des cartes à puces est basée sur trois paramètres. **[2]**

I.2.2.a. A base de ces composants :

On distingue deux types de carte :

- **a_1 : La carte à mémoire :** c'est une des cartes qui sont les plus communes et les moins chères. Elle contient une EEPROM (Electronic Erasable Programmable Read Only Memory) qui stocke toutes les données de l'application. Sa capacité varie de 2ko à 8ko, et une ROM (Read Only Memory) qui stocke des données qui ne changent jamais pendant la vie de la carte.
- **a_2 : La carte à microcontrôleur :** Techniquement, seulement cette carte est digne du nom « carte à puce » qu'on utilise souvent pour tout type de carte. Elle contient une ROM qui contient le système d'exploitation de la carte, une EEPROM qui contient des programmes et des données de l'application de carte, une mémoire versatile RAM (Random Access Memory) qui est utilisée par le processeur pour exécuter des fonctions désireux, un CPU (Central Processing Unit) est le cœur de la carte à puce par exemple : carte GOLD, carte SILVER, carte FUN. La principale différence entre tous ces noms est : le type du microcontrôleur et de l'EEPROM.

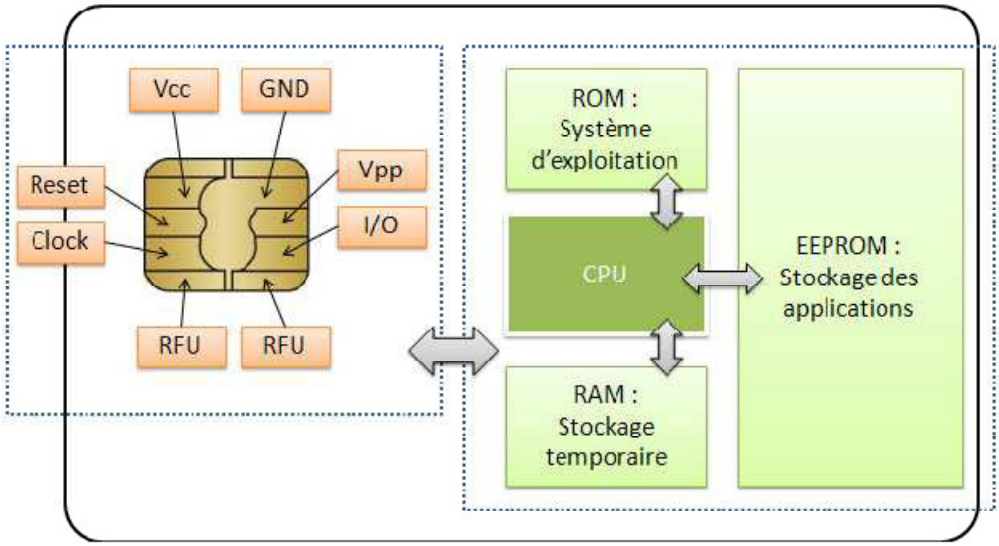


Figure [I.1] : Architecture de la carte à puce à microcontrôleur.

I.2.2.b. A base de son interface :

On distingue trois types de carte :

b1 : La carte à contact : elle exige de l'introduction dans le lecteur de la carte.

b2 : La carte sans contact : elle n'exige pas l'introduction dans le lecteur de la carte.

b3 : La carte combi ou carte hybride : est ce qui a un contact et une interface sans contact.

I.2.2.c. A base de système d'exploitation utilisée :

Le système d'exploitation de la carte à puce généralement appelés SCOS (Smart-card operating system) est embarqué sur la ROM. Il est généralement écrit en language C ou en langage assembleur. Il est responsable de :

- Gère les communications avec le monde extérieur.
- Exécute les commandes reçues via I/O.
- Supervise l'exécution des programmes exécutables stockés dans la carte.
- Gère et assure un accès sécurisé à l'ensemble des fichiers.
- Assure les fonctions de cryptographie

Il y a plusieurs systèmes d'exploitation :

- ✓ Multos : est un système d'exploitation multi-applications pour des cartes qui doivent la sécurité élevé.

- ✓ JavaCard : des cartes à puces qui exécutent des programmes Java.

I.2.3. Positionnement des contacts :

Il y a deux types de positionnement des contacts, la norme ISO (International Organization for Standardization) Internationale et la norme AFNO (Association Française de Normalisation).

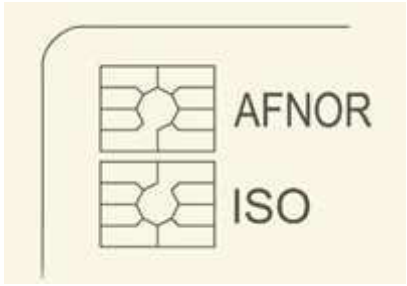


Figure [I.2] : positionnement des contacts ISO et AFNOR.

Dans notre travail en utilise les cartes sous norme ISO7816.

I.2.4. Norme ISO 7816 : [2]

Le standard ISO7816 contient un ensemble de standards qui couvrent différent aspects des cartes à puce avec contact. Ce standard se décline en plusieurs sous normes que voici :

- Partie1 : caractéristiques physiques.
- Partie2 : dimensions et position des contacts.
- Partie3 : signaux électroniques et protocoles de transmission.
- Partie4 : commandes inter-industrie pour l'échange.
- Partie5 : identificateur d'application.
- Partie6 : éléments de données inter-industrie.
- Partie7 : commandes inter-industrie pour SCQL (Structured Card Query Language).
- Partie8 : sécurité de l'architecture et des commandes inter-industrie.

On va détailler une par une ces sous-normes.

I.2.4.a. Norme ISO7816-1 :

Cette norme définit, comme nous l'avons déjà précisé, les caractéristiques physiques des cartes (Figure I.3) ayant une interface physique par des contacts électrique. Elle ne définit pas la nature, le nombre et la position des circuits intégrés dans la carte. Elle fait référence aux normes

relatives aux cartes d'identification ISO7810, ISO7811 partie 1 à 5, ISO7812 et ISO7813. Les caractéristiques introduites par la norme portent sur :

- La protection contre les ultraviolets et les rayons X.
- Le profil de surface des contacts.
- La résistance mécanique des cartes et des contacts (au pliage, à la torsion).
- La résistance électrique des contacts.
- L'interférence électromagnétique entre les pistes magnétiques éventuelles de la carte et de ses circuits intégrés.
- L'exposition de la carte à des champs magnétiques.
- L'exposition de la carte à une décharge d'électricité statique.
- La dissipation thermique du circuit intégré.

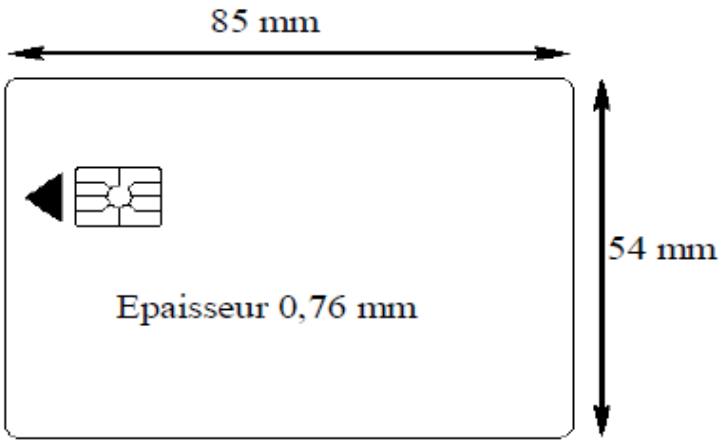


Figure [I.3] : Dimension de la carte à puce.

I.2.4.b. Norme ISO7816-2 :

La norme ISO7816-2 définit les aspects électriques et la situation des contacts sur la carte. La carte comporte huit contacts (Figure I.4) dont voici l'utilisation :

- Le contact 1 correspond à la tension d'alimentation en lecture(VCC).
- Le contact 2 correspond au signal reset (RST), si une tension appliquée sur ce contact, elle déclenche l'initialisation physique et logique du composant.
- Le contact 3 correspond au signal d'horloge (CLK) il fournit le signal d'horloge externe dont dérivera l'horloge interne.

- Le contact 4 est réservé à une utilisation futur (RFU) « Reserved for Futur Use »
- Le contact 5 correspond à la masse (GND).
- Le contact 6 correspond à la tension d'alimentation (VPP) à appliquer sur demande de la carte, pour programmer la mémoire de données (tension en écriture).il est seulement utilisé sur des cartes anciennes.
- Le contact 7 est le contact d'entrée/sortie (I/O) par lequel transitent en half-duplex toutes les données échangées entre la carte et le monde extérieur.
- Le contact 8 est réservé à une utilisation future(RFU).

Une norme utilisant les contacts RFU serait en cours de préparation pour l'utilisation de l'USB avec la carte.

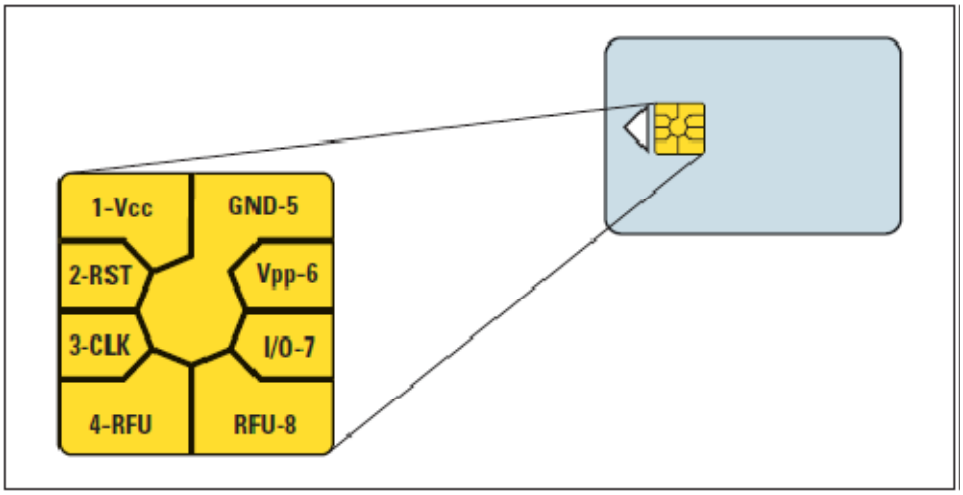


Figure [I.4] : les contacts de la carte à puce.

I.2.4.c. Norme ISO 7816-3 :

Cette partie normalise :

- Les caractéristiques électriques comme la fréquence d'horloge (entre 1Mhz et 5Mhz), la vitesse des communications… etc.
- Les protocoles de transmission TPDU (Transmission Protocol Data Unit).
- La sélection de type de protocole PTS (Protocol Type Sélection). Si plusieurs protocoles sont supportés.
- La réponse au reset ATR (Answer To Reset). Qui correspond aux données envoyées par la carte immédiatement après la mise sous tension.

I.2.4.d. Norme ISO 7816-4 :

ISO 7816-4 vise à assurer une inter-opérabilité. Elle spécifie :

- Le contenu des massages entre la carte et le lecteur CAD (Card Acceptance Device). Qui utilise le protocole APDU (Application Protocole Data Unit) pour les commandes et les réponses.
- Les structures des fichiers et des données : l'accès à ces données, l'architecture de sécurité, la sécurisation des communications.

I.2.4.e. Norme ISO 7816-5 :

Cette norme spécifie le système de numérotation et les procédures d'enregistrement des identifiants d'applications AID (Application Identifier). Les AIDs sont utilisés pour l'identification unique d'une application de la carte et de certains types de fichiers. Un AID est une séquence de 5 à 16 octets. Sa forme est dépendue dans le tableau.

Application identifier (AID)	
National registered application provider (RID)	**Proprietary application identifier extension (PIX)**
5 octets	0 à 11 octets

- la première partie RID (Ressource Identifier), de taille fixe, qui fait toujours cinq octets.
- la seconde partie PIX (Proprietary Identifier eXtension), de taille variant entre 0 et 11 octets.

I.2.4.f. Norme ISO 7816-6 :

Cette norme spécifie les éléments de données inter-industrie pour les échanges, tels que le numéro du porteur de carte, sa photo, sa langue, la date d'expiration,….etc.

I.2.4.g. Norme ISO 7816-7 :

Cette norme propose des commandes inter-industrie pour SCQL. Elle est destinée à la gestion de base de données à l'intérieur de la carte. Son utilisation actuellement serait très marginale.

I.2.4.h. Norme ISO 7816-8 :

La norme concerne la sécurité de l'architecture et des commandes inter-industrie. Les informations sur cette partie ne nous sont pas accessibles pour le moment.

I.2.5. Spécification de la norme ISO 7816 :

- La carte ne dispose pas de sa propre horloge, elle est fournie par l'interface, ou maître (Dans notre cas c'est le PIC16F84 (gold) et le PIC16F876 (silver)), et est généralement sous la barre des 4MHZ.
- Le temps séparant chaque bit est définit par la norme comme étant de 1bit émis toutes les 372 impulsions de l'horloge du maître .Ceci donne un débit de l'ordre de 9600 bits/s avec un Quartz de 3.579 MHZ.
- La communication est du type half-duplex, c'est-à-dire que les entrées et les sorties s'effectuent par alternance et en se servant de la même ligne physique.
- La transmission est du type asynchrone, avec 1 bit de START ,8 bits de données,1 bit de parité pair, et 2 bits de stop.

I.2.6. Les commandes ISO 7816 :

La carte ne prend jamais l'initiative de l'échange d'information .Elle ne fait que répondre à des commandes de la forme :

CLASSE	INSTRUCTION	PARAMETRE1	PARAMETRE2	LONGUEUR

Ou encore sous forme abrégée :

CLASS	INS	P1	P2	LEN

Les significations de chacun des ses octets de la commande reçu se présente comme suit :

- **CLASS** : détermine le groupe d'instructions concernées.
- **INS** : c'est l'instruction proprement dite, ou encore la commande. C'est elle qui détermine l'opération à effectuer par la carte.
- **P1 et P2** : sont deux paramètres que la carte reçoit avec la commande. Leur signification dépend de la commande envoyée. Ils peuvent être inutilisés mais doivent tout de même être présents.

- **LEN** : peut représenter deux choses : soit c'est le nombre d'octet qui suit la commande, dans ce cas la carte s'attendra à recevoir LEN octets supplémentaires, avant de traiter la commande dans sa totalité. Soit ce sera la longueur de la chaîne de réponse que le maître s'attend à recevoir en provenance de la carte. C'est INS qui permet d'établir le rôle de LEN.

I.2.7. Le protocole d'échange d'informations :

L'échange standard d'informations entre le maître et la carte selon la norme ISO 7816, se déroule comme suit :

- A la mise en service, le maître génère un RESET sur la pine MCLR, la carte répond avec un certain nombre d'octets. Cette réponse s'appelle : ATR, pour « Answer To RESET ».
- Le maître envoie la commande sous la forme : **Class INS P1 P2 LEN**
- La carte renvoie l'instruction comme accusée de réception **INS**.
- Le maître envoie éventuellement les données complémentaires.
- La carte envoie la réponse.
- La carte envoie le statut et repasse en mode d'attente de commande.

I.2.8. Les liaisons asynchrones : [4]

Asynchrone, il s'agit donc d'une liaison qui ne fournit pas une horloge destinée à indiquer le début et la fin de chaque bit envoyé. Nous aurons donc besoin d'un mécanisme destiné à repérer la position de chaque bit. Il s'agit d'un mode asynchrone particulier où un bit est donné toutes les 372 impulsions d'horloge.

Pour recevoir correctement les bits envoyés, il faut convenir d'un protocole de communication. Ce dernier doit fixer les informations suivantes :

- La vitesse de transmission en bauds *(*bits par seconde).
- Le format, c'est-à-dire le nombre de **bit de START**, de **bit de stop**, de **bits de données** et le **type de parité**.

Nous allons maintenant expliquer ces concepts et indiquer quelles sont les valeurs employées dans la norme ISO 7816.

- **Le bit de START :** Au repos, la ligne se trouve à l'état haut. L'émetteur fait alors passer la ligne à l'état bas : c'est-à-dire le bit de START. C'est ce changement de niveau qui va permettre de détecter le début de la réception des bits en série. Les valeurs admissibles sont 1 ou 2 bits START (x). La norme ISO 7816 utilise un seul bit de START à zéro « 0 ».

- **Les bits de donnée** : Après avoir reçu le bit START, on trouve les bits de données. Les normes usuelles utilisent 7 ou 8 bits de données. Pour la norme ISO 7816, il y a 8 bits de données, ce qui autorise des valeurs admissibles de 0 à 0xFF pour chaque octet reçu.

- **Le bit de parité** : Le bit de parité est une vérification du bon déroulement du transfert. Dans la norme ISO7816 on utilise une parité paire.

- **Le bit stop (stop-bit)** *:* Après la réception des bits précédents, il est impératif de remettre la ligne à l'état haut pour pouvoir détecter le bit START de l'octet suivant. C'est le rôle du STOP-Bit. Les valeurs admissibles sont de 1 ou 2 STOP-Bits.

Voici, pour illustrer tout ceci, l'émission d'une donnée codée sur 8 bits, avec parité paire et un STOP-Bit. Les lectures sont indiquées par les flèches inférieures :

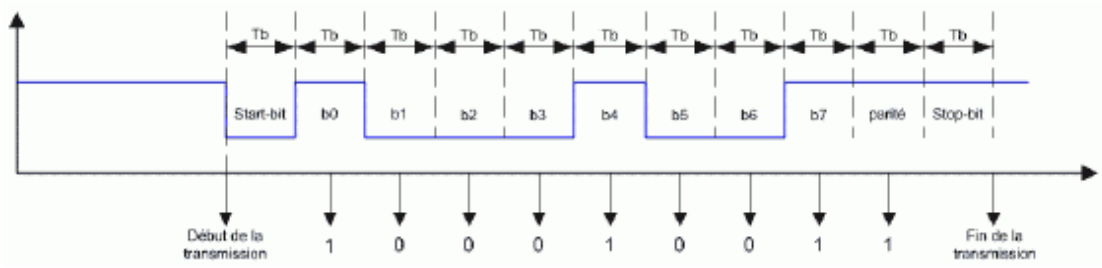


Figure [I.5] : Chronogramme présentant l'émission d'une donnée par une liaison asynchrone.

- **Vitesse et débit** : La durée de transmission d'un bit est constante et dépend de la vitesse 9600 bits/secondes, c'est-à-dire, chaque bit durera 1s/9600=104.17µS.

Dans le cas de la norme ISO 7816, nous utiliserons 1 bit START+8bits de données +1 bit de parité+2 bits de stop=12bits .Le temps total pour recevoir une trame est donc de 1250µs.

I.2.9. Avantages de la carte à puce:

Les avantages de la carte à puce sont la sécurité, la portabilité, la facilité d'utilisation et la personnalisation. La carte à puce est résistante aux attaques car elle n'a pas besoin d'être dépendante d'une ressource externe potentiellement vulnérable. La sécurité de la carte à puce se fait à plusieurs niveaux :

I.2.9.a. Au niveau physique:

- Par des techniques d'impression sur le corps de carte très sophistiquées.

I.2.9.b. Au niveau Hardware :

- Par un numéro de série unique.
- Par l'utilisation de mémoire de type PROM (Programmable Read Only Memory). L'information peut être stockée de manière durable hors-usine et aussi être déplacée par moment à l'intérieur de cette mémoire afin d'éviter des attaques localisées.
- Par blindage physique du composant par collage du micro chip au micromodule avec des techniques spécialisées et insertion des couches métallisé.
- Par contrôle des accès aux mémoires (EEPROM, ROM).

I.2.9.c. Au niveau software :

- Par des contrôles d'accès aux données grâce, par exemple, à un code secrets ou par authentification cryptographique.
- Par un maintien de l'intégrité des données en utilisant la vérification des témoins, les calcules de signatures sur les données internes.
- Grace à des entrées/sortie sécurisées par chiffrement/signature et temps d'exécution des commandes constants.

I.2.10. Applications des cartes à puces :

Les principaux secteurs qui utilisent la carte à puce sont :

- ✓ L'industrie des télécommunications par exemple les cartes téléphoniques prépayées, les carte SIMs insérées dans les téléphones GSM, … etc.
- ✓ L'industrie bancaire et monétaire avec les cartes de crédits qui sont des cartes à microprocesseur avec contact.
- ✓ Le secteur de la santé.
- ✓ Le contrôle d'accès physique des personnes à des locaux.
- ✓ Le porte-monnaie électronique.
- ✓ L'industrie audiovisuelle avec la télévision à péage, … etc.

Pour notre application, on a choisi les cartes de type GOLD et SILVER, possédant un microcontrôleur PIC de type 16F84(GOLD) et 16F876(SILVER) et une mémoire 24C16

(GOLD) et 24C64 (SILVER). Ce type de cartes permet de garantir des performances au niveau puissance de microcontrôleur et grande capacité de mémoire intégrée.

I.3. La carte GOLD: [10]

Même si elle est encore disponible sur le marché car elle peut recevoir de nombreux applications, cette carte est historiquement la première à avoir été réalisée si l'on excepte les Basic Card qui constituent une famille à part. Elle est de ce fait plus simple que certaines cartes actuelles, compte tenu de son âge et donc des composants qui étaient disponibles à l'époque de sa conception. Comme le montre son schéma ci-dessous, elle ne contient en effet qu'un classique microcontrôleur PIC 16F84 associé à une mémoire EEPROM série externe de type 24LC16 (figure I.7). Cette approche, qui a de quoi surprendre puisque le 16F84 contient déjà en interne une mémoire EEPROM de données, est justifiée par le fait que la taille de cette mémoire est très réduite puisqu'elle n'est que de 64 octets. L'ajout de la 24LC16 externe permet de porter cette capacité de mémorisation à 2 K octets ce qui est déjà plus confortable. Cette carte est appelée à la quasi-unanimité **carte Wafer 1** ou bien encore **carte Gold** en raison de la couleur dorée de certaines versions.

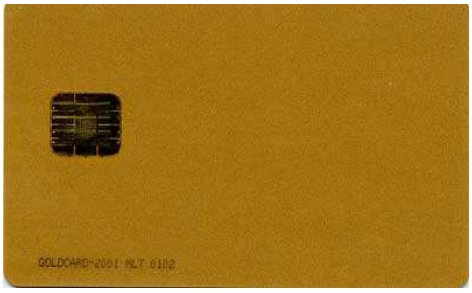


Figure [I.6] : La carte Gold a une couleur dorée. **[S1]**

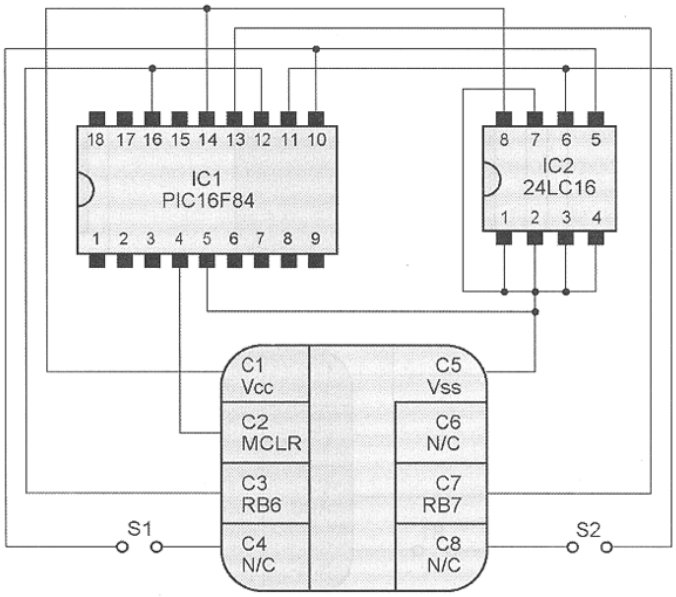


Figure [I.7] : Schéma électronique d'une carte GOLD (Wafer 1). **[S1]**

La carte Gold existe aussi sous la dénomination Wafer 2 ou bien encore Gold 64. Son schéma est alors identique à celui de la Wafer 1, comme le montre la figure I.8, mais la mémoire EEPROM devient de la 24LC64 c'est à dire qu'elle offre une capacité de 8 K octets.

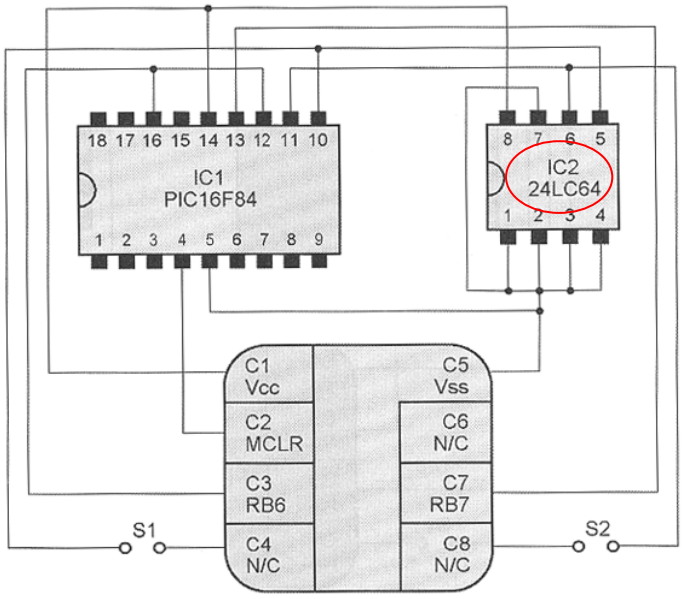


Figure [I.8] : Schéma électronique d'une carte GOLD 64(Wafer 2)**. [S1]**

I.4. La carte SILVER: [10]

Appelée **carte Wafer3** ou **carte SILVER**, car de nombreuses versions de cette carte sont disponibles sous une belle livrée argentée, cette carte reste fidèle à la famille PIC de Microchip, mais fait appel à un circuit plus richement doté en ressources internes avec le 16F876, comme le montre son schéma visible ci-dessous.

Figure [I.9] : la carte SILVER classique. **[S2]**

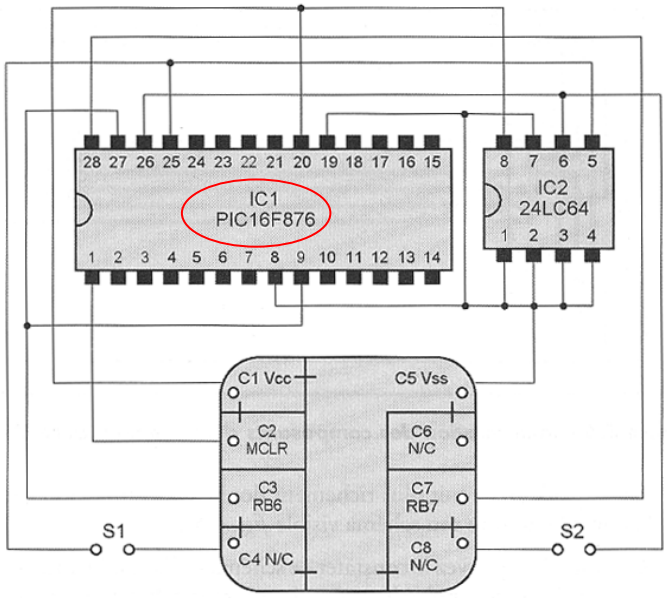


Figure [I.10] : Schéma électronique d'une carte SILVER. **[S2]**

Pour travailler avec une carte à puce, il faut avoir au moins deux appareils :

- Appareil d'écriture (qui est le programmateur de carte à puce).
- Appareil de lecture.

I.5. Programmateur de carte à puce : [9]

Programmateur de carte à puce est un dérivé de JDM Programmer (programmateur des pic) équipé des ports USB ou ports série. Il permet de la programmation extrêmement rapide de toutes les cartes à puces à base de microcontrôleurs PIC et ATMEL, (Gold, Silver, Fun, ATmega, …etc.) à l'aide de son logiciel. Le port série permet d'utiliser les logiciels compatibles avec le mode Phoenix ou Smart Mouse en 3,58MHz, 3,68MHz ou 6,00MHz pour la programmation des cartes sécurisées.

Figure [I.11] : Exemple d'un programmateur Carte à Puce PHOENIX en boitier.

I.6. Lecteur carte à puce : [6]

Le lecteur est le point de passage obligé pour lire dans n'importe quelle carte à puce et aussi est un terminal intelligent qui prend plus ou moins largement en charge les protocoles de communication avec les cartes à puce.

Tout lecteur de carte à puce, associé à son logiciel « ou driver », se charge de toutes les protocoles de communication normalisés.

Figure [I.12] : Exemple d'un lecteur carte à puce**.**

I.7. La communication entre le lecteur et la carte à puce : [6]

Echange de commandes avec le lecteur de carte à puce tel que défini dans l'ISO 7816-4. La communication entre l'hôte et la carte est half-duplex. Elle se fait à l'aide de paquets appelés APDU (Application Protocol Data Units) en respectant le protocole de l'ISO 7816-4. Un APDU contient une commande ou une réponse. Le mode Maître/Esclave est utilisé. Ainsi la carte joue un rôle passif et attend une commande APDU à partir de l'hôte. Elle exécute l'instruction spécifiée dans la commande et retourne une réponse APDU.

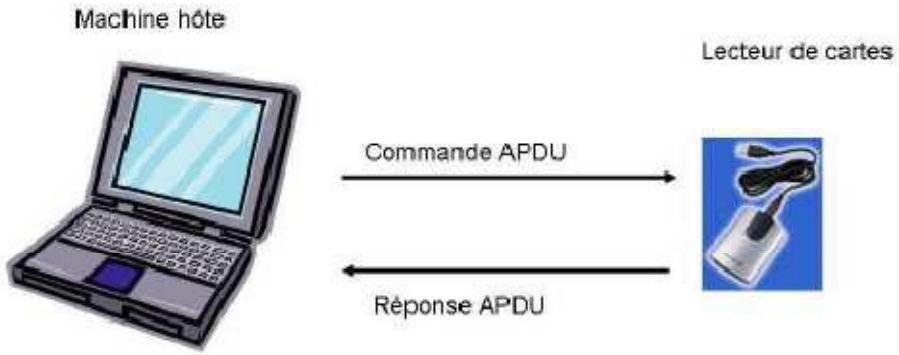


Figure [I.13] : le modèle de communication de la carte à puce.

Conclusion :

La carte à puce est un outil très puissant. On peut le considérer comme un petit ordinateur. L'avantage le plus important de la carte à puce est sa sécurité, d'autre part avec la taille petite, cet outil rend la commodité dans l'utilisation.

Pour construire une application de la carte à puce. On doit développer parallèlement trois parties qui sont le langage de programmation, le programmateur de carte à puce et le lecteur de carte à puce.

Le choix d'un lecteur ou d'un programmateur de carte dépend de ce que nous voulant faire, mais aussi et surtout des types de cartes que nous voulant lire ou programmer.

Chapitre II

Les microcontrôleurs (PIC) et mémoires

II.1 Introduction :

Un microcontrôleur est un composant électronique autonome. Il est généralement moins puissant qu'un microprocesseur en terme de rapidité ou de taille mémoire, il se contente le plus souvent d'un bus de 8 ou de 16 bits. Ce que fait que c'est un composant très bon marché parfaitement adapte pour piloter les applications embarquées dans de nombreux domaines d'application. Nous pensons qu'on ne se tromperait pas beaucoup si on affirme qu'aujourd'hui il y'a un microcontrôleur (± grand) dans chaque équipement électronique :

- Informatique (souris, modem, …etc.) ;
- Vidéo (Appareil photos numérique, caméra numérique, …etc.) ;
- Contrôle des processus industriels (régulation, pilotage) ;
- Appareil de mesure (affichage, calcul statistique, mémorisation) ;
- Automobile (ABS, injection, GPS, airbag) ;
- Multimédia (téléviseur, carte audio, carte vidéo, magnétoscope) ;
- Téléphones (fax, portable, modem) ;
- Carte à puce (GOLD, SILVER, FUN…etc.).

Plusieurs constructeurs se partagent le marché des microcontrôleurs, citons INTEL, MOTOROLA, ATMEL, ZILOG, PHILIPS et enfin MICROCHIP avec ses PICs très populaires qui nous intéresse ici dans ce projet.

On peut dire que seul le langage de programmation (Assembleurs) constitue la différence majeure en deux microcontrôleurs (similaires) venant de deux constructeurs différents.

Nous avons choisit dans ce chapitre d'apprendre les microcontrôleurs à travers une étude détaillée des microcontrôleurs 16F84 et 16F876, qui sont constitue les éléments fondamentaux de la famille mid-Line qui est la famille « moyenne puissance » de MICROCHIP.

II.2 Généralités:

Les microcontrôleurs, quelque soit leurs constructeurs, ont des architectures très similaires et sont constitués des modules fondamentaux assurant les mêmes fonctions:

- UAL (Unité Arithmétique et Logique).
- Ports d'E/S
- interfaces de communications série
- Interfaces d'E/S analogiques
- Timers et horloge temps réels.

En illustrons l'architecture des microcontrôleurs dans la figure suivante.

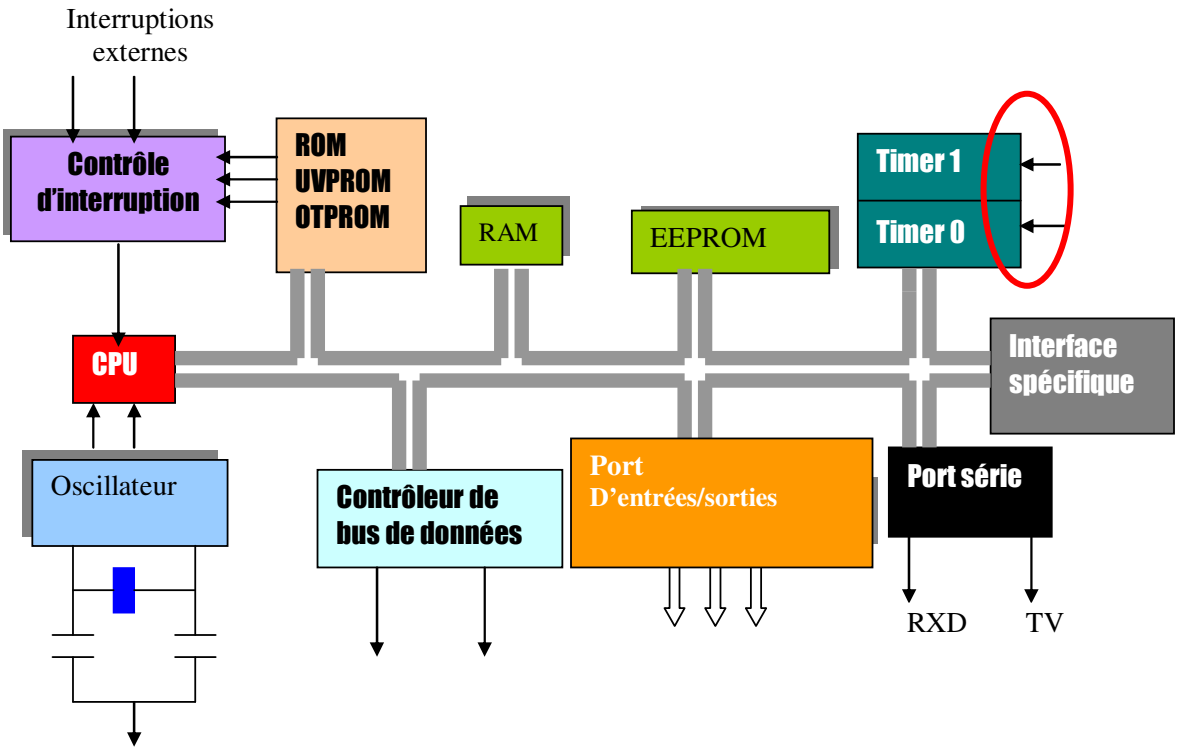


Figure [II.1] : Architecture générale des microcontrôleurs.

Un PIC est un microcontrôleur de chez Microchip. Ses caractéristiques principales sont:

- Séparation des mémoires de programme et de données (architecture Hardware) : on obtient ainsi une meilleure bande passante et des instructions et des données pas forcément codées sur le même nombre de bits.
- Communication avec l'extérieur seulement par des ports : il ne possède pas de bus d'adresses, de bus de données et de bus de contrôle comme la plupart des microprocesseurs.

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

Il existe trois familles de PIC :

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

Un PIC est identifié par un numéro de la forme suivante :

xx(L) XXyy–zz

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

Par exemple PIC 16F84 –10, soit :

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

II.3. Le PIC 16F84: [7]

Ce modèle de PIC (*Programmable Interface Contrôler*) est un circuit de petite taille, fabriqué par la Société américaine ***Arizona MICROCHIP Technologie***.

En le regardant pour la première fois, il fait davantage penser à un banal circuit intégré logique TTL ou MOS, plutôt qu'à un microcontrôleur.

Son boîtier est un DIL (*Dual In Line*) de 2x9 pattes.

En dépit de sa petite taille, il est caractérisé par une architecture interne qui lui confère souplesse et vitesse incomparables.

Ses principales caractéristiques sont :

- 13 lignes d'entrées/sorties, réparties en un port de 5 lignes (Port A) et un port de 8 lignes (Port B)
- alimentation sous 5 Volts
- architecture interne révolutionnaire lui conférant une extraordinaire rapidité
- une mémoire de programme pouvant contenir 1.019 instructions de 14 bits chacune (allant de l'adresse 005 à l'adresse 3FF)
- une mémoire RAM utilisateur de 68 emplacements à 8 bits (de l'adresse 0C à l'adresse 4F)
- une mémoire RAM de 2x12 emplacements réservée aux registres spéciaux
- une mémoire EEPROM de 64 emplacements
- une horloge interne, avec pré diviseur et chien de garde
- possibilité d'être programmé en mode *in-circuit*, c'est à dire sans qu'il soit nécessaire de le retirer du support de l'application
- vecteur de Reset situé à l'adresse 000
- un vecteur d'interruption, situé à l'adresse 004
- bus d'adresses de 13 lignes
- présence d'un code de protection permettant d'en empêcher la duplication
- facilité de programmation, simplicité et faible prix.
- 35 instructions codées sur 14 bits.
- Données sur 8 bits.
- 1 cycle machine par instruction, sauf pour les sauts (2 cycles machines).
- Vitesse maximum 10 MHz soit une instruction en 400 ns (1 cycle machine = cycles d'horloge).

- 4 sources d'interruption.
- 1000 cycles d'effacement/écriture pour la mémoire flash, 10.000.000 pour la mémoire de donnée EEPROM

II.3.1. Brochage du 16F84 : [12]

La figure [II.2] montre le brochage de circuit intégré des pics 16F8X. 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

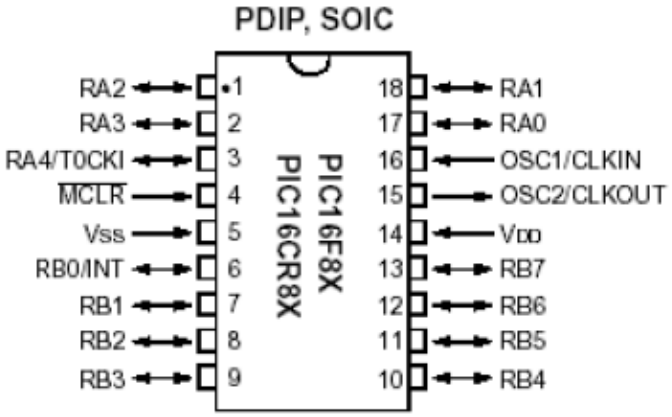


Figure. [II.2] : brochage des PICs 16F8X et 16CR8X.

II.3.2. Architecture générale : [12]

La Figure [II.3] présente l'architecture générale d'un PIC. Il est constitué des éléments suivants :

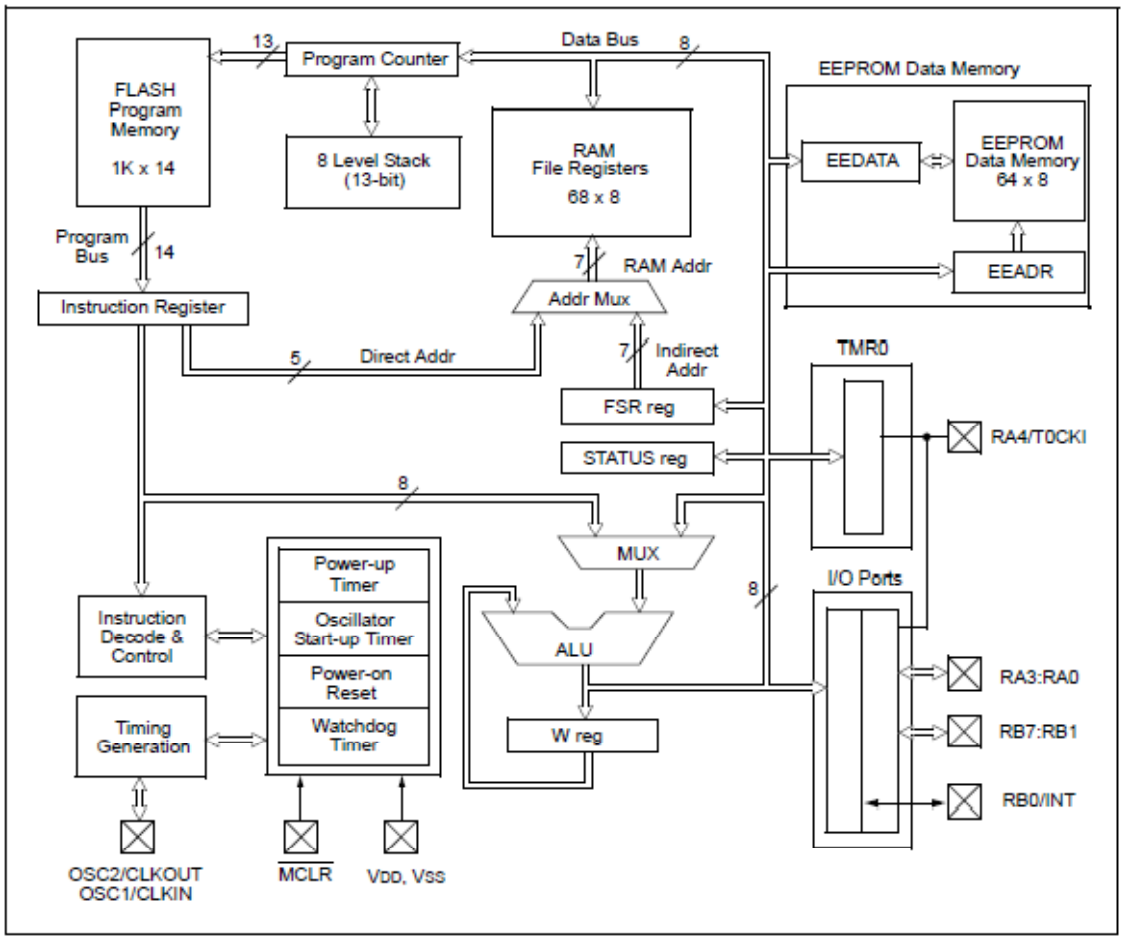


Figure [II.3] : Architecture générale du PIC 16F84.

- ✓ un système d'initialisation à la mise sous tension (Power-up Timer, Oscillator Strat-up Timer,…etc.).
- ✓ un système de génération d'horloge à partir du quartz externe (Timing Generation).
- ✓ une unité arithmétique et logique (UAL ou en anglais ALU).
- ✓ une mémoire flash pour programme de 1k "mots" de 14 bits.
- ✓ un compteur de programme (Program Counter) et une pile (Stack).
- ✓ un bus spécifique pour le programme (Program Bus).
- ✓ un registre contenant le code de l'instruction à exécuter (Instruction Register).
- ✓ un bus spécifique pour les données (Data Bus).

- ✓ une mémoire RAM de 68 emplacements à 8 bits.
- ✓ les SFR(Special Function Registers).
- ✓ une mémoire EEPROM de 64 octets de données.
- ✓ deux ports d'entrées/sorties.
- ✓ un compteur (Timer).
- ✓ un chien de garde (Watchdog).

II.3.3. Organisation de la mémoire : [7]

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

II.3.3.a. Mémoire de programme :

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

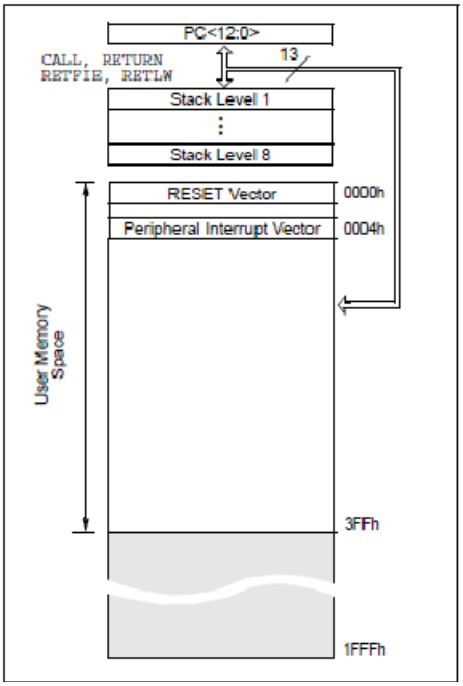


Figure [II.4] : Organisation de la mémoire de programme et de la pile.

II.3.3.b. Mémoire de données :

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

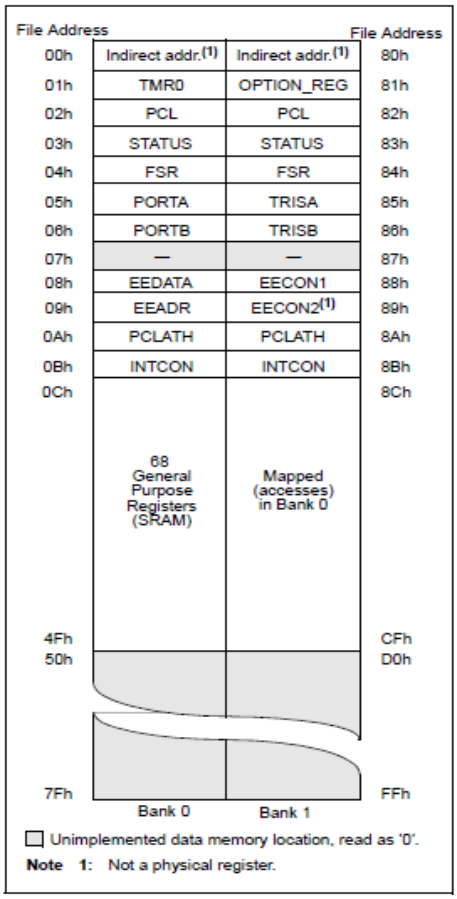


Figure [II.5] : Organisation de la mémoire de données.

Les instructions orientées octets ou bits contiennent une adresse sur 7 bits pour désigner l'octet avec lequel l'instruction doit travailler. D'après la Figure [II.5], l'accès au registre TRISA d'adresse 85h, par exemple, est impossible avec une adresse sur 7 bits. C'est pourquoi le constructeur a défini deux banques. Le bit Rb0 du registre d'état (STATUS.5) permet de choisir entre les deux. Ainsi, une adresse sur 8 bits est composée de RP0 en poids fort et des 7 bits provenant de l'instruction à exécuter.

b1 : Registres généraux :

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

b2 : Registres spéciaux – SFRs :

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

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ Reset.

3: Other (non power-up) RESETS include: external RESET through $\overline{MCLR}$ and the Watchdog Timer Reset.

4: On any device RESET, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

Figure [II.6] : Description des SFR.

On donne la description des registres spéciaux :

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

- PCL (02h-82h) : Contient les poids faibles du compteur de programmes (PC). Le registre PCLATH (0Ah-8Ah) contient les poids forts.
- STATUS (03h-83h) : Il contient l'état de l'unité arithmétique et logique ainsi que les bits de sélection des banques.
- FSR (04h-84h) : Permet l'adressage indirect
- PORTA (05h) : Donne accès en lecture ou écriture au port A, 5 bits. Les sorties sont à drain ouvert. Le bit 4 peut être utilisé en entrée de comptage.
- PORTB (06h) : Donne accès en lecture ou écriture au port B. Les sorties sont à drain ouvert. Le bit 0 peut être utilisé en entrée d'interruption.
- EEDATA (08h) : Permet l'accès aux données dans la mémoire EEPROM.
- EEADR (09h) : Permet l'accès aux adresses de la mémoire EEPROM.
- PCLATCH (0Ah-8Ah) : Donne accès en écriture aux bits de poids forts du compteur de programme.
- INTCON (0Bh-8Bh) : Masque d'interruptions.
- OPTION_REG (81h) : Contient des bits de configuration pour divers périphériques.
- TRISA (85h) : Indique la direction (entrée ou sortie) du port A.
- TRISB (86h) : Indique la direction (entrée ou sortie) du port B.
- EECON1 (88h) : Permet le contrôle d'accès à la mémoire EEPROM.
- EECON2 (89h) : Permet le contrôle d'accès à la mémoire EEPROM.

b3. Mémoire EEPROM :

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

II.3.4. Jeu d'instructions :

Les PICs sont conçus selon une architecture RISC. Programmer avec un nombre d'instructions réduit permet de limiter la taille de leur codage et donc de la place mémoire et du temps d'exécution.

II.3.4.a. Format général :

Toutes les instructions sont codées sur 14 bits. Elles sont regroupées en trois grands types (Figure [II.7]) :

- Instructions orientées octets ;
- Instructions orientées bits ;
- Instructions de contrôle.

Le registre de travail W joue un rôle particulier dans un grand nombre d'instructions.

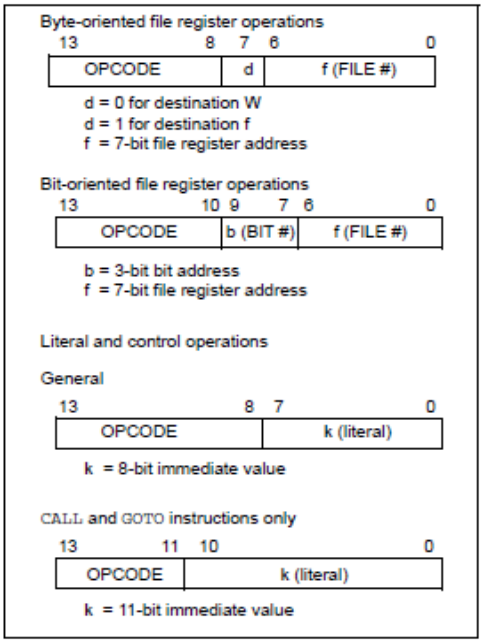


Figure [II.7] : Format générale des instructions.

II.3.4.b. Exemple d'instruction – le transfert :

Trois instructions de transfert sont disponibles sur le PIC 16F84. La Figure [II.8] permet de transférer le contenu du registre W dans un registre f. On peut noter la valeur du bit 7 à 1 et les bits 0 à 6 donnant le registre concerné.

MOVWF — **Move W to f**

Syntax:	[*label*] MOVWF f
Operands:	$0 \le f \le 127$
Operation:	$(W) \rightarrow (f)$
Status Affected:	None
Encoding:	00 | 0000 | 1fff | ffff
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example

```
MOVWF   OPTION_REG
```

Before Instruction
OPTION = 0xFF
W = 0x4F
After Instruction
OPTION = 0x4F
W = 0x4F

Figure [II.8] : transfer du registre W dans le registre f.

La Figure [II.9] permet de transférer une donnée contenue dans un registre W vers le registre f. Dans ce cas, l'intérêt est de positionner le bit Z. On peut noter ici le bit 7 qui prend la valeur « d » fournie dans le code de l'instruction pour choisir la destination : W ou f.

MOVF — **Move f**

Syntax:	[*label*] MOVF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) → (destination)
Status Affected:	Z
Encoding:	00 | 1000 | dfff | ffff
Description:	The contents of register f is moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
MOVF    FSR, 0
```

After Instruction
W = value in FSR register
Z = 1

Figure [II.9] : Transfer du contenu de registre W dans le registre f.

La figure [II.10] présente une instruction qui permet de charger une constante dans le registre W. Ici, la valeur à charger est donnée sur 8 bits, le bit 7 n'étant pas utile puisque le code de l'instruction dit que la valeur est à charger dans le registre W.

MOVLW	Move Literal to W			
Syntax:	[*label*] MOVLW k			
Operands:	$0 \le k \le 255$			
Operation:	$k \rightarrow (W)$			
Status Affected:	None			
Encoding:	11	00xx	kkkk	kkkk
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.			
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'	Process data	Write to W
Example	MOVLW 0x5A			
	After Instruction			
	W = 0x5A			

Figure [II.10] : Transfer d'une constante dans le registre W.

II.3.4.c. Liste des instructions : [12]

La figure [II.11] donne la liste de toutes les instructions du PIC 16F84.

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	TO,PD	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

Figure [II.11] : Liste des instruction du PIC 16F84.

II.3.5. Exécution d'un programme – notion de pipe-line :

La Figure [II.12] montre l'enchaînement des instructions tous les 4 cycles d'horloge. Pendant le premier cycle machine, l'instruction à exécuter est stockée en mémoire RAM. Le cycle suivant, elle est exécuté. Chaque instruction dure donc 2 cycles machine.

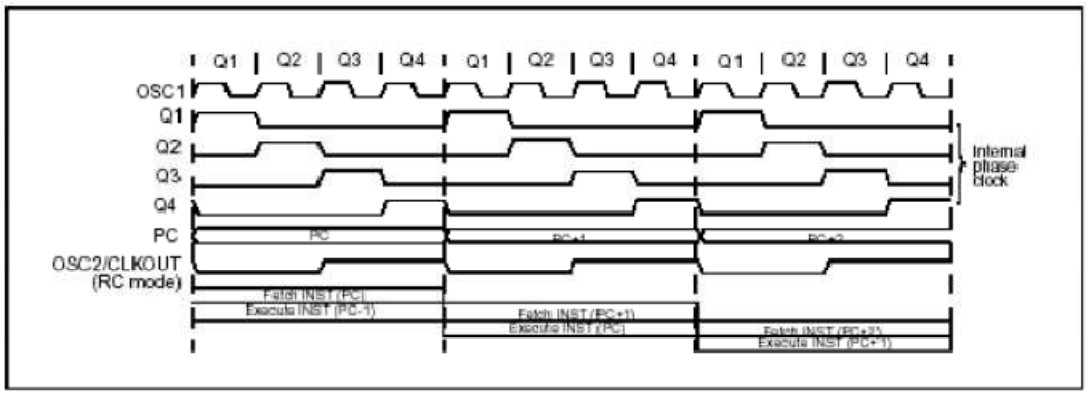


Figure [II.12] : Enchaînement des instructions.

La notion de pipeline permet de réduire ce temps à un seul cycle machine. L'idée est d'exécuter l'instruction n-1 pendant que l'instruction n est chargée en mémoire RAM. Ainsi, une fois le système enclenché, pendant chaque cycle machine une instruction est chargée et une autre exécutée. On a donc l'équivalent d'une instruction par cycle machine.

La Figure [II.13] montre un exemple d'exécution d'un programme. Notons que l'instruction CALL dure 2 cycles machine comme toutes les instructions de branchement.

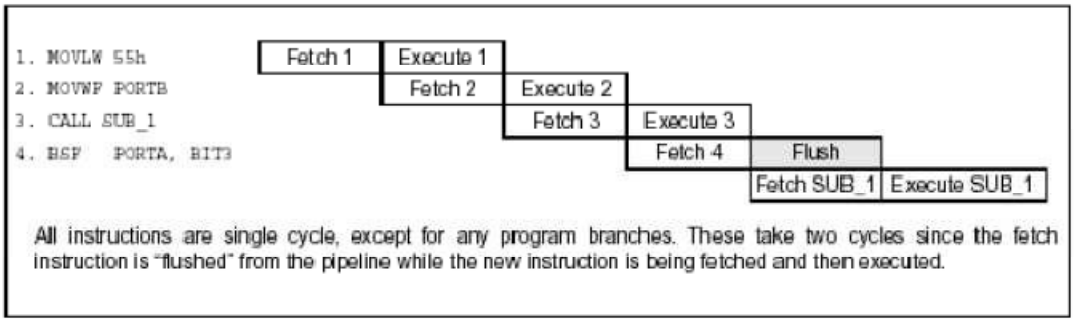


Figure [II.13] : La notion Pipe line du PIC 16F84.

II.3.6. Modes d'adressages : [7]

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

- ✓ **Adressage immédiat :** La donnée est contenue dans l'instruction.
- ✓ **Adressage direct :** La donnée est contenue dans un registre.
- ✓ **Adressage indirect :** L'adresse de la donnée est contenue dans un pointeur.

II.3.6.a. Adressage immédiat :

La donnée est contenue dans l'instruction.

Exemple : movlw 0xC4 ; **Transfert la valeur 0xC4** dans W

II.3.6.b. Adressage direct :

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

Exemple : movw 0x2B, 0 ; **Transfert dans W** la valeur **contenue à l'adresse 0x2B.**

II.3.6.c. Adressage indirect :

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

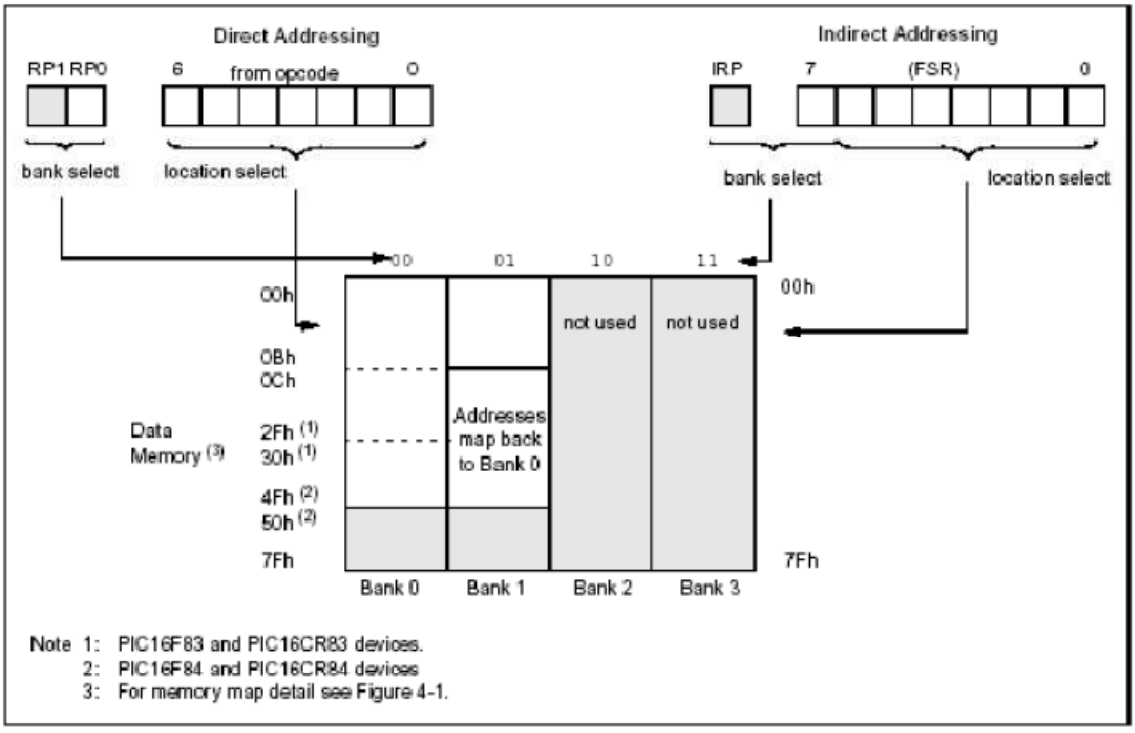


Figure [II.14] : Adressage direct et indirect à la mémoire de données.

L'accès au registre d'adresse contenue dans FSR se fait en utilisant le registre INDF. Il se trouve à l'adresse 0 dans les deux banques. Il ne s'agit pas d'un registre physique. On peut le

voir comme un autre nom de FSR, utilisé pour accéder à la donnée elle-même, FSR servant à choisir l'adresse.

Exemple :

```
Movlw    0x1A      ; Charge 1Ah dans W.
Movwf    FSR       ; Charge W, contenant 1Ah, dans FSR.
Movw     INDF, 0   ; Charge la valeur contenue à l'adresse 1Ah dans W.
```

II.3.7. Ports d'entrées/Sorties : [12]

Le PIC 16F84 est doté de deux ports d'entrées/Sorties appelés Port A et Port B.

II.3.7.a. Port A :

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

Le registre PORTA, d'adresse 05h dans la banque 0, permet d'y accéder en lecture ou en écriture.

Le registre TRISA, d'adresse 85h dans la banque 1, permet de choisir le sens de chaque patte (entrée ou sortie) : un bit à 1 positionne le port en entrée, un bit à 0 positionne le port en sortie.

La Figure [II.15] donne le câblage interne d'une patte du port A :

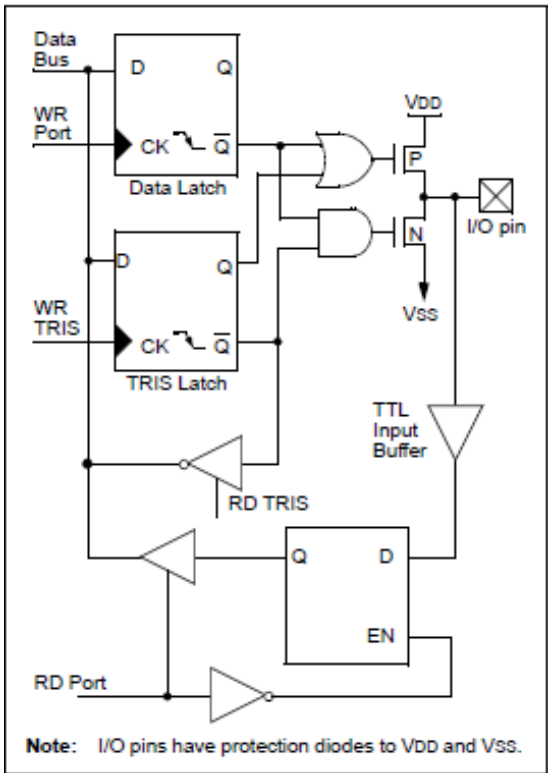


Figure [II.15] : Câblage interne d'une patte du part A.

- ✓ "Data Latch" : Mémorisation de la valeur écrite.
- ✓ "TRIS Latch" : Mémorisation du sens (entrée ou sortie) de la patte.
- ✓ "TTL input buffer" : Buffer de lecture de la valeur du port. La lecture est toujours réalisée sur la patte, pas à la sortie de la bascule d'écriture.
- ✓ Transistor N :

  En écriture : Saturé ou bloqué suivant la valeur écrite.

  En lecture : Bloqué.

- ✓ Transistor P : Permet d'alimenter la sortie.

II.3.7.b. Port B :

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

Le registre PORTB, d'adresse 06h dans la banque 0, permet d'accéder en lecture ou en écriture.

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

Le câblage interne d'une porte du port B ressemble beaucoup à celui du port A (Figure [II.16]).

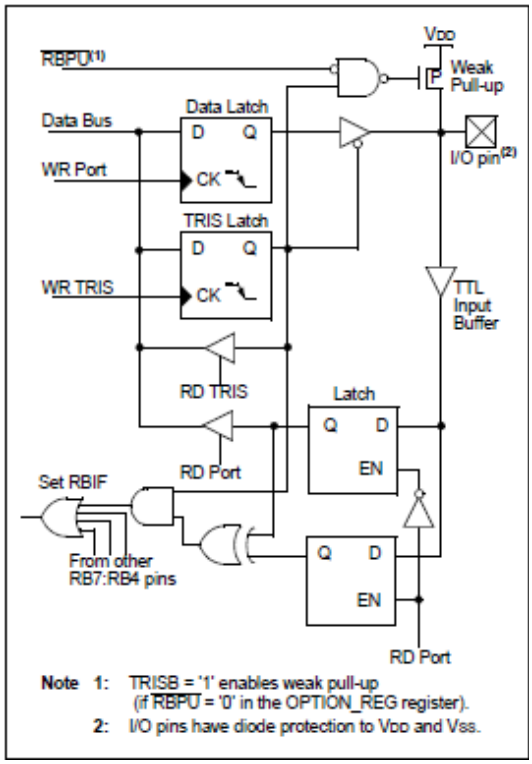


Figure [II.16] : Câblage interne d'une patte du port B.

On peut noter la fonction particulière pilotée par le bit RBPU (OPTION_REG.7) qui permet d'alimenter (RBPU=0) ou non (RBPU=1) les sorties.

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

II.3.8. Le Compteur :

Le PIC 16F84 est doté d'un compteur 8 bits. La Figure [II.17] en donne l'organigramme.

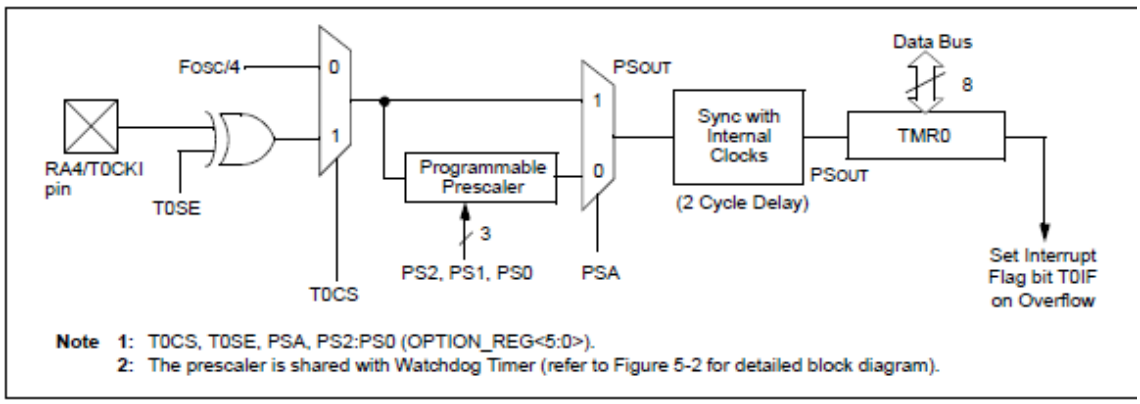


Figure [II.17] : Organigramme du Timer.

II.3.8.a. Le Registre TMR0 :

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

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

II.3.8.b. Choix de l'horloge :

Le timer0 peut fonctionner suivant deux modes en fonction du bit T0CS (OPTION_REG.5) :

- *En mode Timer (T0CS=0)* : le registre TMR0 est incrémenté à chaque cycle machine (si le pré-diviseur n'est pas sélectionné).
- *En mode Compteur (T0CS=1)* : le registre TMR0 est incrémenté sur chaque front montant ou chaque front descendant du signal reçu sur la broche RA4/T0CKl en fonction du bit T0SE (OPTION_REG.4). Si T0SE=0, les fronts montants sont comptés, T0SE=1, les fronts descendants sont comptés.

II.3.8.c. Pré-diviseur : [7]

En plus des deux horloges, un pré-diviseur, partagé avec le chien de garde, est disponible. La période de l'horloge d'entrée est divisée par une valeur comprise entre 2 et 256 suivant les bits PS2, PS1 et PS0 (respectivement OPTION_REG.2, 1 et 0) (Figure [II.18]). Le bit PSA (OPTION_REG.3) permet de choisir entre la pré-division de timer0 (PSA=0) ou du chien de garde (PSA=1).

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

Figure [II.18] : Valeurs du pré-diviseur en fonction de PSA, PS2, PS1 et PS0.

II.3.8.d. Fin de comptage et interruption :

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

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

II.3.9. Accès à la mémoire EEPORM : [7]

Le PIC possède une zone EEPROM de 64 octets accessibles en lecture et en écriture par le programme.

Quatre registres sont utilisés pour l'accès à la mémoire EEPROM du PIC :

- ✓ EEDATA : contient la donnée ;
- ✓ EEADR : contient l'adresse ;
- ✓ EECON1 : est le registre de contrôle de l'accès à l'EEPROM ;
- ✓ EECON2 : joue un rôle spécifique lors de l'écriture.

II.3.9.a. Lecture :

Pour lire une donnée dans la mémoire EEPROM, il faut mettre l'adresse dans EEADR et positionner RD à 1. La valeur lue est alors disponible dans EEDATA au cycle machine.

II.3.9.b. Ecriture :

Pour écrire une donnée dans la mémoire EEPROM, il faut d'abord mettre l'adresse dans EEADR et la donnée dans EEDATA. Un cycle bien spécifique doit ensuite être respecté pour que l'écriture ait lieu.

II.3.10. Interruptions :

II.3.10.a. Différentes sources d'interruption :

Dans le cas du PIC 16F84, il existe 4 sources d'interruption :

- ✓ INT : Interruption externe, broche RB0/INT
- ✓ TMR0 : Fin de comptage
- ✓ PORTB : Changement d'état du port B (RB7-RB4)
- ✓ EEPROM : Fin d'écriture en EEPROM

II.3.10.b. Validation des interruptions :

Chacune de ses sources peut être validée indépendamment grâce aux bits 3 à 6 du registre INTCON. Le bit GIE de ce même registre permet une validation générale des interruptions. Ainsi, pour que le déroulement du programme en cours soit déclenché, il faut qu'un des événements extérieurs soit détecté, que l'interruption correspondante soit validée et que la validation générale soit activée.

II.3.10.c. Séquence de détournement vers le sous-programme d'interruption :

Par construction, l'interruption survient n'importe quand pendant l'exécution du programme. Avant l'exécution du sous-programme d'interruption, il faut donc sauvegarder l'adresse de l'instruction, suivante celle en cours, pour l'exécuter après le sous-programme d'interruption. L'adresse de retour est stockée dans la pile. Cette opération est gérée automatiquement par le processeur.

Une fois l'adresse de retour sauvegardée, le compteur de programme peut être chargé avec l'adresse du sous-programme à exécuter.

Dans le cas du PIC, à cause de la faible taille de la pile, une interruption n'est pas interruptible. Le bit GIE de validation générale est donc mis à 0 au début du sous-programme d'interruption. Cette opération est gérée automatiquement par le processeur.

II.3.10.d. Sauvegarde et restitution du contexte :

C'est un point important pour tous les sous-programmes, qui devient capital pour les sous- programmes d'interruption. En effet, beaucoup d'instructions modifient le registre STATUS et/ou utilisent le registre W. Afin de les rendre dans le même état à la fin du sous-programme d'interruption qu'au début, il faut les sauvegarder au début et les recopier à la fin. Si d'autres registres sont utilisés dans le sous-programme d'interruption, il faut généralement les sauvegarder aussi.

II.3.10.e. Retour au programme initial :

Une fois le sous-programme d'interruption terminé, après la restitution du contexte, il faut revenir au programme initial. C'est l'instruction« retfie » qui le permet. Elle commence par revalider les interruptions (GIE=1) puis elle revient au programme initial grâce à la valeur du compteur de programme empilée.

II.3.11. Chien de garde :

C'est un système de protection contre un blocage du programme. 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 bloquer. 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, déclenche le chien de garde qui relance le programme.

II.3.12. Mode sommeil :

Lorsque le PIC n'a rien à faire (par exemple lors de l'attente d'une mesure extérieure), ce mode est utilisé pour limiter sa consommation : le PIC est mis en sommeil (le programme s'arrête) jusqu'à son réveil (le programme repart). Ce mode est principalement utilisé pour les systèmes embarqués fonctionnant sur pile.

II.4. Le PIC 16F876: [8]

Le PIC 16F876 possède en plus des instructions très puissantes donc : un programme à développer réduit, une programmation simple grâce au mode série

En fait, la cause principale du choix du 16F876, est qu'il dispose de l'option du convertisseur A/D pour satisfaire coté « Acquisition ».

Nous allons maintenant s'intéresser à la structure interne du PIC 16F876, avec lequel nous avons travaillé.

Le 16F876 est un microcontrôleur de MICROCHIP, fait partie intégrante de la famille des Mid-Range (16) dont la mémoire programme est de type flash (F) de type 876 et capable d'accepter une fréquence d'horloge maximale de 4Mhz.

II.4.1.Caractéristiques générales :

PIC	FLASH	RAM	EEPROM	I/O	A/D	PORT //	Port série
16F876	8K	368	256	22	5	NON	USART/MSSP

II.4.2. Brochage du 16F876 : [13]

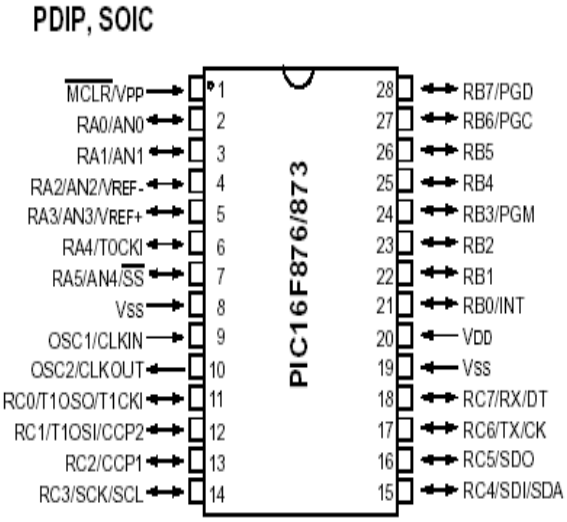


Figure [II.19] : Brochage du PIC16F876.

II.4.3. Les particularités électriques :

On constate que sur le schéma concernant le 16F876, on a 2 connexions «VSS» qui sont reliées à la masse. En fait, en interne, ces pins sont interconnectés. La présence de ces 2 pins s'explique pour une raison de dissipation thermique. Les courants véhiculés dans le PIC sont loin d'être négligeables du fait des nombreuses entrées/sorties disponibles.

Le constructeur a donc décidé de répartir les courants en plaçant 2 pins pour l'alimentation VSS, bien évidemment, pour les mêmes raisons (dissipation thermique), ces pins sont situées de part et d'autre du PIC, et en positions relativement centrales.

II.4.4. Organisation du 16F876 : [8]

La mémoire du 16F876 est divisée en 3 parties.

- **La mémoire programme :** Elle est constituée de 8K mots de 14 bits. C'est dans cette zone que nous allons écrire notre programme.
- **La mémoire EEPROM :** Cette EEPROM (Electrical Erasable Programmable Read Only Memory), est constituée de 256 octets que nous pouvons lire et écrire depuis notre programme. Ces octets sont conservés après une coupure de courant et sont très utiles pour conserver des paramètres semi-permanents.

- **La mémoire RAM et organisation :** Cette mémoire RAM est celle que nous allons sans cesse utiliser. Toutes les données qui y sont stockées sont perdues lors d'une coupure de courant.

La mémoire RAM disponible du 16F876 est de 368 octets. Elle est répartie de la manière suivante:

- 80 octets en banque 0, adresses 0x20 à 0x6F
- 80 octets en banque 1, adresses 0xA0 à 0XEF
- 96 octets en banque 2, adresses 0x110 à 0x16F
- 96 octets en banque 3, adresses 0x190 à 0x1EF
- 16 octets communs aux 4 banques, soit 0x70 à 0x7F = 0xF0 à 0xFF ; 0x17F=0x1F0 à 0x1FF.

- **Watchdog :** Sous ce nom étrange nous allons découvrir une fonction capable de surveiller le bon fonctionnement du programme que le micro contrôleur exécute. Le rôle du Watchdog (ou chien de garde) est de "reseter" le micro contrôleur. Si « 1 » on ne remet pas à zéro périodiquement (à intervalle définissable) un registre interne grâce à 1 instruction CLRWDT (Clear Watchdog), si le programme tourne par exemple dans une boucle sans fin (c'est un bug !) il ne peut remettre à 0 le chien de garde et ainsi le micro contrôleur se reset afin de relancer le programme. Cette fonction est bien sûr désactivée au moment de la programmation du micro contrôleur, c'est la directive d'assemblage.

- **Le TIMER :** Un TIMER est un registre interne au micro contrôleur, celui-ci s'incrémente au grès d'une horloge, ce registre peut servir par exemple pour réaliser des temporisations, ou bien encore pour faire du comptage (par l'intermédiaire d'une broche spécifique : RA4/TOKI). Le PIC 16F876 possède trois TIMERS sur 8 bits (il compte jusqu' à 256) configurable par logiciel.

II.4.5. Les ports entrée/sortie :

On dispose de 22 broches d'entrées/sorties, chacune configurables soit en entrée soit en sortie (PORTA, PORTB, PORTC).

Un registre interne au PIC, nommé TRIS, permet de définir le sens de chaque broche d'un port d'entrées/sorties. En règle générale, un bit positionné à « 0 » dans le registre TRIS donnera

une configuration en ***sortie*** pour la broche concernée ; si ce bit est positionné à «1», ce sera une broche d'***entrée***.

a) Particularité du port A :

Le 16F876 dispose de 5 canaux d'entrée analogique. Nous pouvons donc échantillonner successivement jusque 5 signaux différents avec ce composant. Les pins utilisés sont les pins AN0 à AN4 (qui sont en fait les dénominations analogiques des pins RA0 à RA3 + RA5).

On peut noter également que les pins ANx sont des pins d'entrée. Il n'est donc pas question d'espérer leur faire sortir une tension analogique. Ceci nécessiterait un convertisseur numérique/analogique dont n'est pas pourvu notre PIC.

b) Particularités du port B :

Hors de sa fonction principale autant que ports d'entrées /sorties, on note la pine RB0 qui, en configuration d'entrée, est de type « Trigger de Schmitt » quand elle est utilisée en mode interruption «INT» ; La lecture simple de RB0 se fait, de façon tout à fait classique, en entrée de type TTL. Encore il y a (RB3-RB6-RB7) qui peut servir dans la programmation en cas d'absence de programmateur commercial.

c) Particularités du port C :

C'est un port tout ce qu'il y a de plus classique, or qu'il a deux pines qu'on utilisera plus tard dans la communication série avec le PC à travers (TX et RX) (pines 17 et 18).

II.4.6. L'oscillateur :

L'horloge système peut être réalisée soit avec un quartz, soit avec une horloge extérieure, soit avec un circuit RC. Dans ce dernier cas, la stabilit 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.

II.4.7. MCLR :

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

II.5. Les différents types de mémoires :

La mémoire sert à enregistrer des données sous forme binaire (une suite de 0 et 1). On distingue deux types de mémoire, la RAM et la ROM.

II.5.1. La RAM :

Les différentes dénominations portent sur les temps d'accès et la largeur de bus (8bits, 16bits, …etc.). Elle nécessite une mise sous tension pour conserver des données.

La DRAM (RAM Dynamique) est la plus rapide (temps d'accès de 60ns), mais nécessite un rafraichissement régulier de l'information.

II.5.2. La ROM :

Ce type sert à stocker des informations de façon indélébile, c'est à dire qu'elle n'est pas une mémoire volatile (son contenue n'est pas perdu en cas de coupure de la tension d'alimentation du calculateur).

Il y a plusieurs types de ROM : **[S11]**

II.5.2.a. La PROM: (Programmable Read Only Memory)

Ce sont des ROM programmable par l'utilisateur (à l'aide d'un programmateur), Ces mémoires sont constituées de fusibles pouvant être grillés grâce à un appareil qui envoie une forte tension (12V) dans certains fusibles. Un fusible grillé correspond à un 0, et un fusible non grillé à un 1. Ces mémoires ne peuvent être programmées qu'une fois.

II.5.2.b. L'EPROM :

C'est une PROM qui se laisse effacer à laide de lumière ultraviolette (utilisation d'un bruleur, encore appelé effaceur). S'il faut modifier le programme, on efface L'EPROM puis on la reprogramme avec les données modifiées (à l'aide d'un programmateur d'EPROM).

II.5.2.c. L'EEPROM :

Utilisant la technologie des portes de silicium flottantes, ce sont des ROM dont la programmation, l'effacement partiel ou global s'effectue électroniquement. Les EEPROM sont donc aussi des PROM effaçables.

De part de son prix et de sa faible intégration, l'EEPROM est très peu utilisée actuellement en grand série.

II.5.2.d. La FLASH-EPROM :

Cette mémoire morte a une caractéristique de pouvoir s'effacer électriquement, tous comme l'EEPROM. Son avantage est comme suit :

- La FLASH-EPROM peut être effacée et programmée sans la sortir du circuit dont elle fait partie.
- Les temps de programmation et d'effacement sont très rapides.
- Côut relativement bas en grande série.

Désavantage :

- La nécessite d'effacer toute la mémoire (pas d'effacement partiel).

II.5.3. Dénomination des mémoires EEPROM :

24CXX indique des mémoires série. Le XX indique le nombre de Kbits. Donc par exemple 24C64 indique mémoire série de 64Kbits soit 8Ko.

II.5.4. Description générale de l'EEPROM 24C16 :

II.5.4.a. Brochage de l'EEPROM 24C16 :

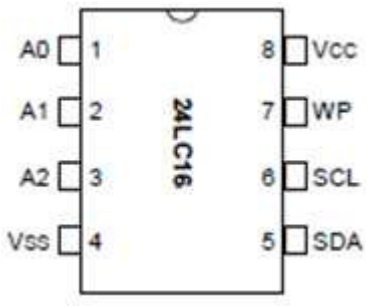


Figure [II.20] : Brochage de l'EEPROM 24C16.

La configuration des pins est comme suite :

- A0 à A2 : Adresse input.
- SDA: Serial Adresse Data I/0.
- SPL : Serial Clock Input.
- WP : Write Protect Input.
- Vcc : Power Supply.
- Vss : Ground.

Les dispositifs 24C16 son 16.384 bits de mémoire CMOS non volatile effaçable

électriquement. Ces dispositifs sont conforment à toutes les caractéristiques de protocole 12C standard à deux fils, ils sont conçus pour réduire au minimum le nombre de broche du dispositif, et simplifient les conditions de disposition au panneau du PC.

II.5.4.b. Protocole I2C :

Le protocole I2C (Inter Integrated Circuit) à été développer au début des années quatre-vingt par Philips Semi Conducteur, pour permettre de relier facilement à un microprocesseur les différents circuits.

Le protocole I2C permet de communique entre les composants électroniques très divers grâce à seulement trois fils : un signal de donnée (SDA), un signal d'horloge (SCL), et un signal de référence électronique (Masse).

Conclusion

Cette partie a été consacrée à la présentation de la carte à puce, des microcontrôleurs et les mémoires (en particulier l'EEPROM), tout en illustrant ces différentes caractéristiques afin des les mieux exploités.

Maintenant, nous pouvons passer à la conception de notre programmateur et lecteur puisque les composants, les plus importants dans notre système, nous est déjà familier.

Chapitre III

Partie I : Réalisation de Programmateur

III.1. Introduction :

Le programmateur de carte à puce qu'on va réaliser vient d'appliquer des notions théoriques requises au cours de l'étude théorique, donc on va montrer comment réaliser l'application la plus simple possible, elle consiste à mettre en œuvre avec la minimum de moyens un programmateur de carte à puce.

III.2. Description du Programmateur carte à puce :

Les cartes GOLD et SILVER contiennent toutes les deux un microcontrôleur PIC. La réalisation d'un programmateur pour ces deux cartes se borne donc à celle d'un programmateur de PIC, équipé bien sûr d'un connecteur pour cartes à puce.

Tous les microcontrôleurs récents supportent la programmation en circuit (ou **ICSP** pour : In Circuit Serial Programming). Cette programmation, qui a lieu sous forme série, n'impose de devoir accéder qu'aux pattes : MCLR (reset), CLK (horloge externe), RB6 et RB7 du microcontrôleur, pattes qui sont justement celles qui sont accessibles via le connecteur des cartes à puce GOLD et SILVER.

Le seul problème qui peut se poser est celui de la mémoire EEPROM associée au microcontrôleur pour laquelle trois situations différentes sont à considérer :

Il convient tout d'abord de faire la distinction entre les applications qui nécessitent une programmation préalable de cette mémoire et celles qui n'en ont pas besoin. Dans ce dernier cas, aucun problème ne se pose puisque l'application programmée dans le PIC gère la mémoire contenue dans la carte sans que l'on ait à s'occuper de quoi que ce soit en phase de programmation.

Par contre, dans le premier cas, il faut pouvoir programmer la mémoire EEPROM depuis l'extérieur de la carte. Deux cas sont donc à nouveau à considérer :

Si nous utilisons une carte Gold ou Silver« maison », en réalise le programmateur comme expliqué par ailleurs, rien ne nous empêche d'enlever momentanément la mémoire de son support afin de la transporter sur un programmateur adéquat.

Si vous utilisez une vraie carte Gold ou Silver, c'est à dire une carte dans laquelle les composants sont intégrés sous forme de puce, cette manipulation est évidemment impossible et il faut faire appel à un programme « loader ».

III.2.1. Schéma du programmateur :

Ce programmateur présenté ci-dessous est dérivé du schéma de base très connu sous le nom de "JDM Programmer". Il se connecte sur le port série de n'importe quel compatible PC et exploite les niveaux RS 232 pour générer les tensions de programmation nécessaires.

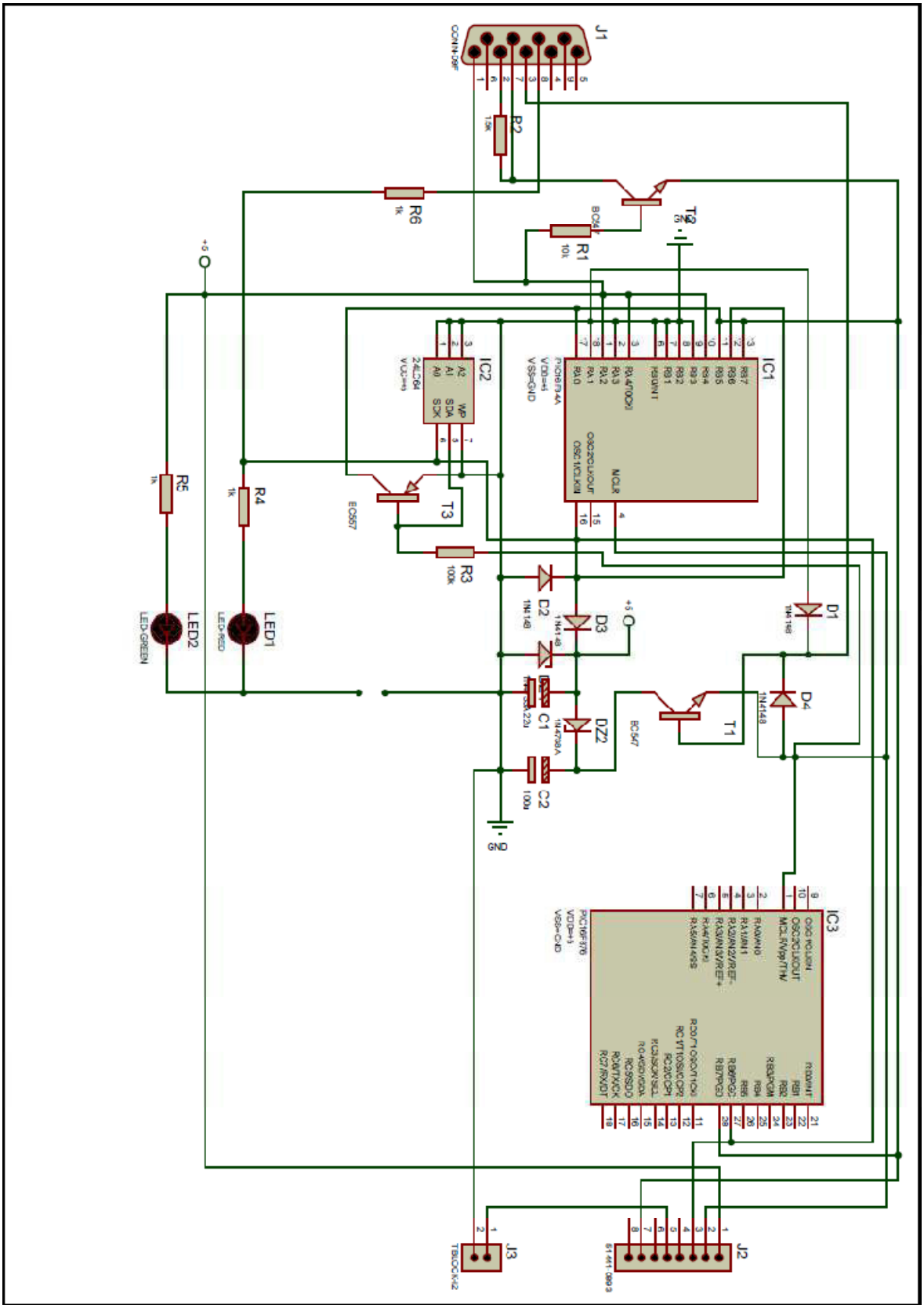


Figure [III.1] : Schéma électrique du programmateur**.**

III.2.2. Principe de fonctionnement :

Son principe est relativement simple, mais le schéma utilisé s'avère particulièrement astucieux de façon à générer les deux tensions nécessaires pour programmer les PIC, à savoir la tension d'alimentation VDD de 5 volts et la tension de mise en mode programmation, appliquée à l'entrée MCLR, de 13 volts. Ces tensions sont obtenues par redressement, filtrage et régulation au moyen des diodes Zener DZ1 et DZ2, à partir des niveaux +/- 12 volts disponibles sur les différentes lignes de l'interface série RS 232 du PC associé.

Ces mêmes lignes servent évidemment à délivrer au circuit l'horloge et les données de programmation après écrêtage de leurs niveaux à 5 volts. Le programmateur pilote trois supports vides :

1. un support 8 pattes destinées aux mémoires EEPROM de la série 24xx qui équipent les carte gold et silver "maison".
2. un support 18 pattes destiné aux PIC 16F84 ou 16C84 qui équipent les cartes Gold.
3. un support 28 pattes étroites destinées au PIC 16F876 qui équipe les cartes Silver.

Comme cela ne coûtait que le tracé de quelques pistes en plus sur le circuit imprimé, nous avons ajouté un support 8 pattes et un support 18 pattes dont le brochage qui permettent de programmer en outre, sans aucun adaptateur, les 12C5xx et 12C67x en boîtier 8 pattes et tous les PIC en boîtier 18 pattes : 16C55x, 16C61, 16C62x, 16C71, 16C71x, 16C8x, 16F8x et16F62x.

Le connecteur visible correspond quant à lui au connecteur de cartes à puce destiné à recevoir les cartes Gold ou Silver, véritables ou de fabrication "maison", dont il permet la programmation directe du microcontrôleur qu'elles contiennent.

III.2.3. Approvisionnement des composants :

Même s'il ne présente pas de difficulté majeure, l'approvisionnement des composants appelle un commentaire concernant le connecteur de cartes à puce. Le modèle utilisé est un type standard disponible chez de très nombreux revendeurs et se trouve être le plus souvent un modèle ITT Canon ou Molex. Tout modèle strictement compatible, prévu pour une disposition des contacts ISO, convient aussi mais il faut vérifier bien qu'il dispose d'un interrupteur de détection de carte.

III.2.4. La réalisation du notre programmateur :

Nous avons réalisé notre programmateur (de PICs, EEPROM et cartes Wafer) sur plaque d'essais. Et après la mise en marche de notre circuit, après autant de nombreux essais et tests,

nous avons utilisé le logiciel ISIS7 version Professionnel (appelé aussi PROTEUS) pour le traçage du circuit imprimé et nous avons obtenu le typon suivant :

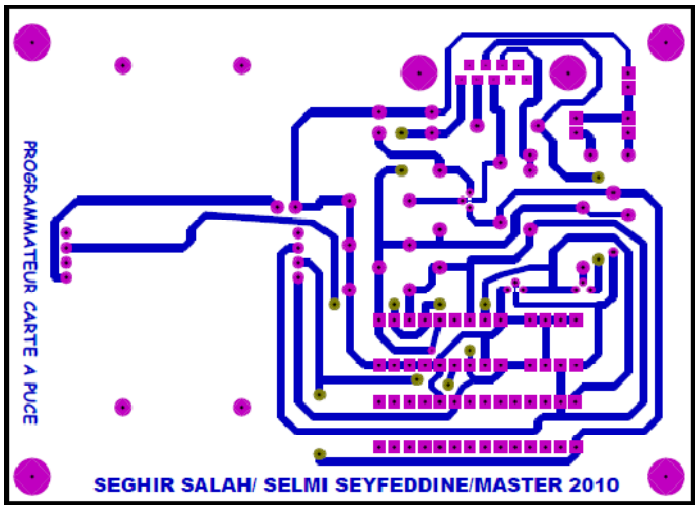


Figure [III.2] : typon (coté cuivre).

III.2.5. L'implantation des composants :

L'implantation des composants est à faire en suivant les indications de la figure ci-dessous. Nous commençons par les STRAPS, puis les connecteurs, les supports, les résistances et condensateurs. Nous terminerons par les transistors et diodes en veillant à bien respecter leur sens.

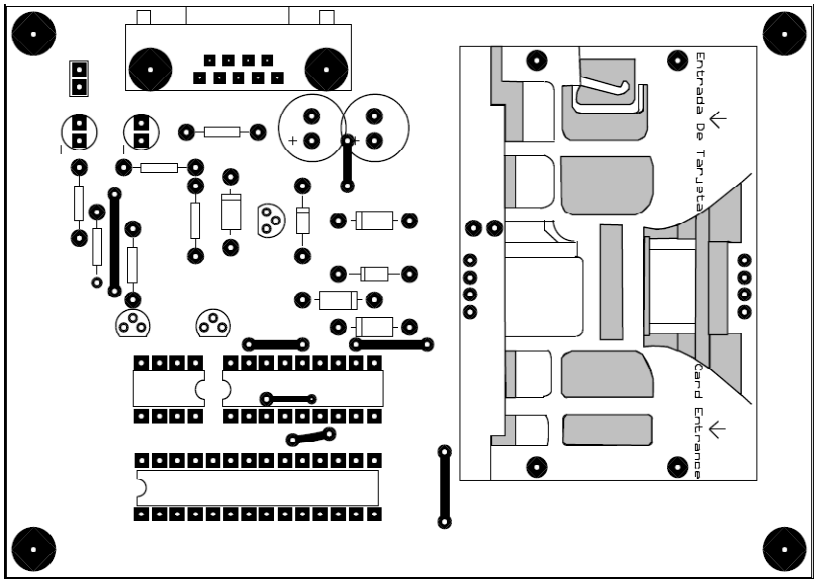


Figure [III.3] : implantations des composants (coté composant).

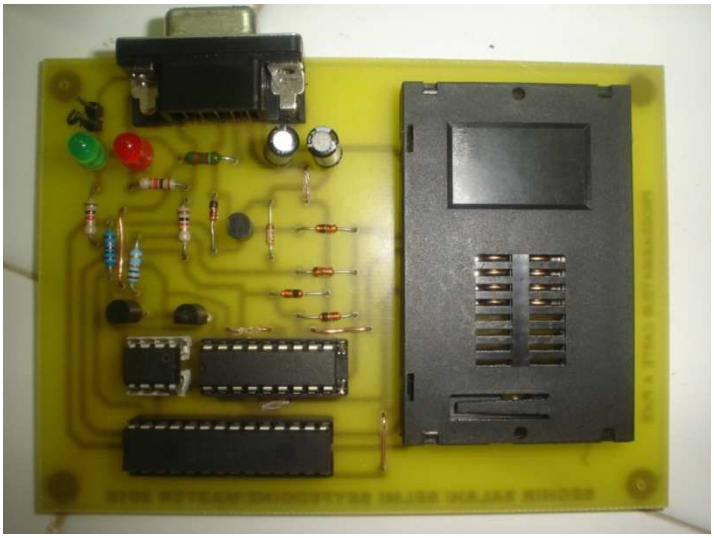


Figure [III.4] : Le programmateur pour cartes Gold et Silver réaliser.

Le montage doit être raccordé au port série de n'importe quel ordinateur personnel (PC de Personal Computer). Le brochage standard adopté par le connecteur 9 points, dont nous avons équipé le circuit imprimé, nous permet d'utiliser tout câble normalisé **droit** (c'est à dire sans croisement de fils) du commerce. Pour utiliser le programmateur il faut évidemment un logiciel. Tout logiciel supportant le "JDM Programmer" convient pour ce montage mais nous nous recommande l'excellent IC-Prog, Ce logiciel qui est aujourd'hui francisé, très souple d'emploi, et supporte d'innombrables programmateurs et circuits intégrés.

Avant de l'utiliser, nous faites appel à son menu "Configuration Hardware" et paramétrerons-le comme indiqué sur la recopie d'écran ci-dessous. Seul le port série utilisé (Com 2 sur cette figure) pourra éventuellement être modifié en fonction de celui que nous aurons utilisé sur le PC.

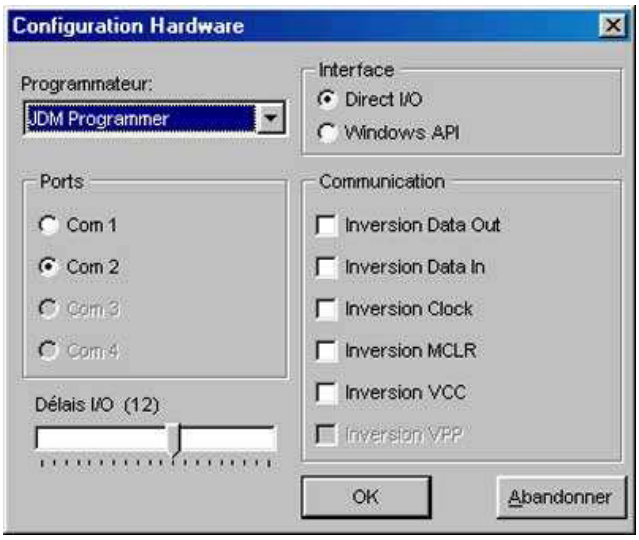


Figure [III.5] : Configuration du logiciel Ic-Prog.

Avant de nous lancer dans la programmation d'un PIC ou d'une carte, en test notre programmateur qui dispose pour cela des deux LED rouge et verte. En Mette en place le STRAP S1 puis en lance le Ic-Prog. La LED verte (LED2) doit s'allumer indiquant la présence de l'alimentation du montage. Selon l'initialisation du port série réalisée par notre système d'exploitation, il se peut même que cette LED s'allume dès la connexion du programmateur à ce port.

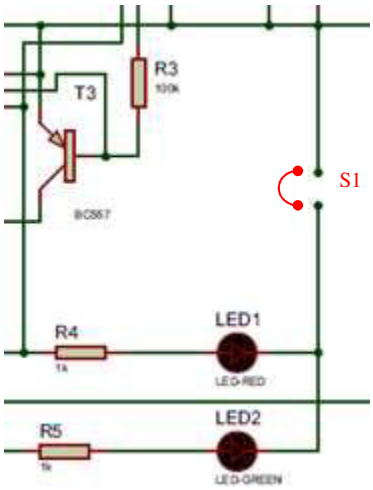


Figure [III.6] : Position du Strap S1.

Nous Sélectionnons ensuite un 16F84 et en lance sa programmation, sans aucun circuit ni carte dans le programmateur, et nous vérifions que la LED rouge clignote rapidement (son intensité lumineuse est assez faible mais c'est normal). Si tel est le cas, notre programmateur a de grandes chances d'être bon pour le service. Enlevons le strap S1 et passons à la phase de programmation.

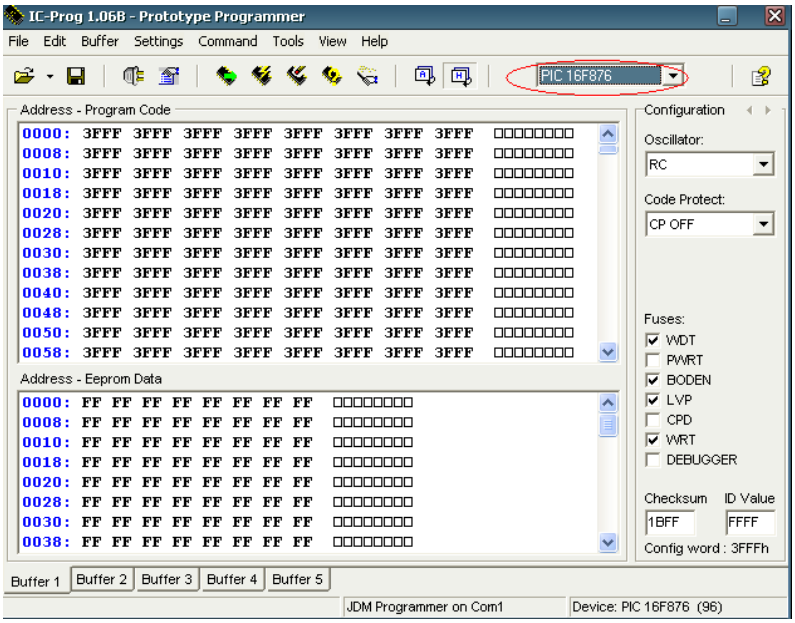


Figure [III.7] : Sélection du PIC16F84.

Nous pourrons alors lire, effacer et programmer tous les microcontrôleurs et mémoires directement supportés par notre montage, ainsi que les microcontrôleurs et les mémoires EEPROM contenues dans les cartes Gold et Silver ou de fabrication "maison".

Notre programmateur fonctionne de façon irréprochable pour tous les circuits et cartes qu'il supporte, cependant, compte tenu du mode de génération des tensions d'alimentation et de programmation à partir du port série du PC, on dois faire les remarques suivantes.

- ✓ En évite de faire fonctionner le programmateur avec S1 en place, car le courant consommé par les LED peut faire chuter la tension disponible en dessous du seuil nécessaire pour une bonne programmation
- ✓ Sur certains portables, les niveaux délivrés par les ports séries et surtout le courant pouvant être fourni par ces ports est anormalement faible et ne permet pas un fonctionnement normal du montage. Si nous somme dans cette situation, il n'y a évidemment aucune solution.
- ✓ Les adaptateurs transformant un port USB en port série, disponibles notamment pour les portables dépourvus de port série, ne permettent généralement pas un fonctionnement correct de notre montage.
- ✓ Si nous rencontrons des problèmes ou des erreurs de programmation, commençons par vérifier la tension entre VSS (5) et VDD (14) du support 18 pattes pendant la programmation d'un circuit : elle doit être supérieure à 4,7 volts. Vérifiions ensuite, toujours en phase de programmation, la tension entre VSS (5) et MCLR (4) du support 18 pattes. Elle doit être au minimum de 12,75 volts. Si ce n'est pas le cas, et si bien sûr aucune erreur de câblage n'a été commise, c'est que le port série de notre PC ne délivre pas des niveaux suffisants. Sur un PC de bureau, c'est cependant une situation qui reste rarissime.

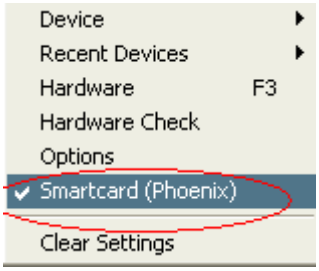
III.3. Programmation de notre carte à puce GOLD (PIC16F84+24lC16) à l'aide d'IC-prog V1.05:

Connecter le câble informatique entre le connecteur DB-9 femelle du programmateur et un port série de votre PC, COM1 ou COM2.

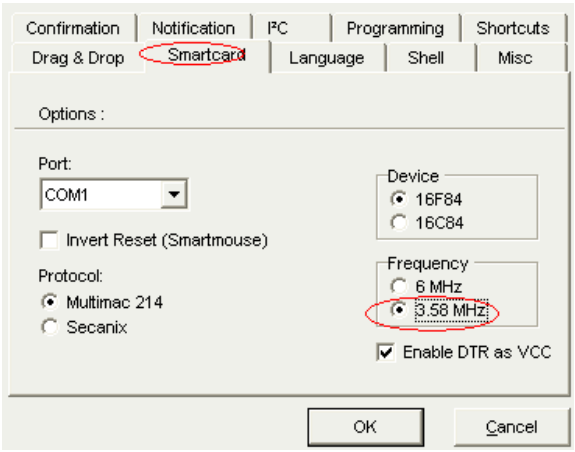
- Lancer le logiciel ICPROG.EXE.
- Insérer la carte à puce à fond dans le lecteur, contact vers le haut.

icprog

- Sélectionner le menu « Commande\Assistant Smartcard ».



- Sélectionner le mode «JDM(PIC) ».
- Cliquer sur « Continuer ». Le Loader permettant l'accès à l'EEprom se charge automatiquement dans le PIC16F84.
- Sélectionner le mode « Phoenix 3,5 MHz ».



- Cliquer sur « continuer ». Sélectionner le fichier pour l'EEprom 24LC16. Cliquer sur Ouvrir.
- Cliquer sur « Continuer ». Le programme se charge dans l'EEprom 24LC16.
- Sélectionner le mode «JDM(PIC) ».
- Cliquer sur « continuer ». Sélectionner le fichier pour le PIC16F84. Cliquer sur Ouvrir.
- Cliquer sur « Continuer ». Le programme se charge dans le PIC16F84.

Voilà, votre microcontrôleur PIC16F84 et votre EEprom 24LC16 sont programmés dans la GoldCard.

Le même cas pour la Silver Card (Pic16f876+24LC64).

Conclusion

On conclut dans cette première partie pratique que notre programmateur est capable de programmer les PICs 16F8x ,16F87x et les mémoires EEPROM de huit pattes avec l'essentiel est de programmer les cartes à puces Gold et Silver.

Chapitre III

Partie II : Réalisation de Lecteur

III.1.Introduction :

On va montrer dans cette partie comment réaliser l'application la plus simple possible, elle consiste à mettre en œuvre avec le minimum de moyens un lecteur de carte à puce GOLD, SILVER, FUN, …etc.

III.2.Description du lecteur carte à puce (Phoenix ou Smart Mouse) :

Comme nous l'ai indiqué dans notre chapitre consacré aux lecteurs de cartes, si nous utilisons les nombreux logiciels de manipulation de cartes à puce parfois un peu "spéciaux", nous nous apercevrons vite que ceux-ci utilisent presque tous un lecteur de cartes **Phoenix** ou **Smart Mouse**, selon le cas. Nous proposons donc de réaliser un tel lecteur et, pour qu'il soit vraiment polyvalent.

Ce lecteur a une deuxième fonction, propre aux cartes Gold et Silver, qui est de permettre la programmation des mémoires EEPROM contenues sur ces cartes avec beaucoup de facilité grâce à un petit logiciel ou "loader" par exemple : Phoenix.

III.2.1.Schéma du lecteur :

L'horloge utilisée par la carte est générée par un oscillateur à quartz réalisé autour d'IC1-a pour fonctionner à **3,579 MHz**, ou de IC1-b pour fonctionner à **6 MHz** (voir figure III.8). Les STRAPS **S4** et **S5** permettent de bloquer l'oscillateur non utilisé, l'autre se trouvant alors automatiquement relié à l'entrée horloge de la carte via IC1-c. La commande de **RESET** de la carte utilise la ligne de contrôle **CTS** du port série, disponible en 7 de J1 (voir figure III.9). Elle est convertie de RS 232 en TTL par l'intermédiaire d'IC2 qui n'est autre qu'un classique **MAX232**. Selon que le lecteur doit être compatible Phœnix ou Smart Mouse, cette commande de RESET peut être directe ou inversée. Un choix est donc possible au moyen des STRAPS **S1** et **S2** (voir figure III.9) qui permettent d'appliquer le signal issu de la patte 7 de J1 à la carte à puce, via S2, ou de lui faire subir une inversion par IC1-d avant d'arriver à la carte via S1. Dans tous les cas, la LED1 permet de visualiser l'état de la ligne RESET de la carte ce qui s'avère bien utile en présence d'un logiciel en cours de développement ou d'une carte récalcitrante.

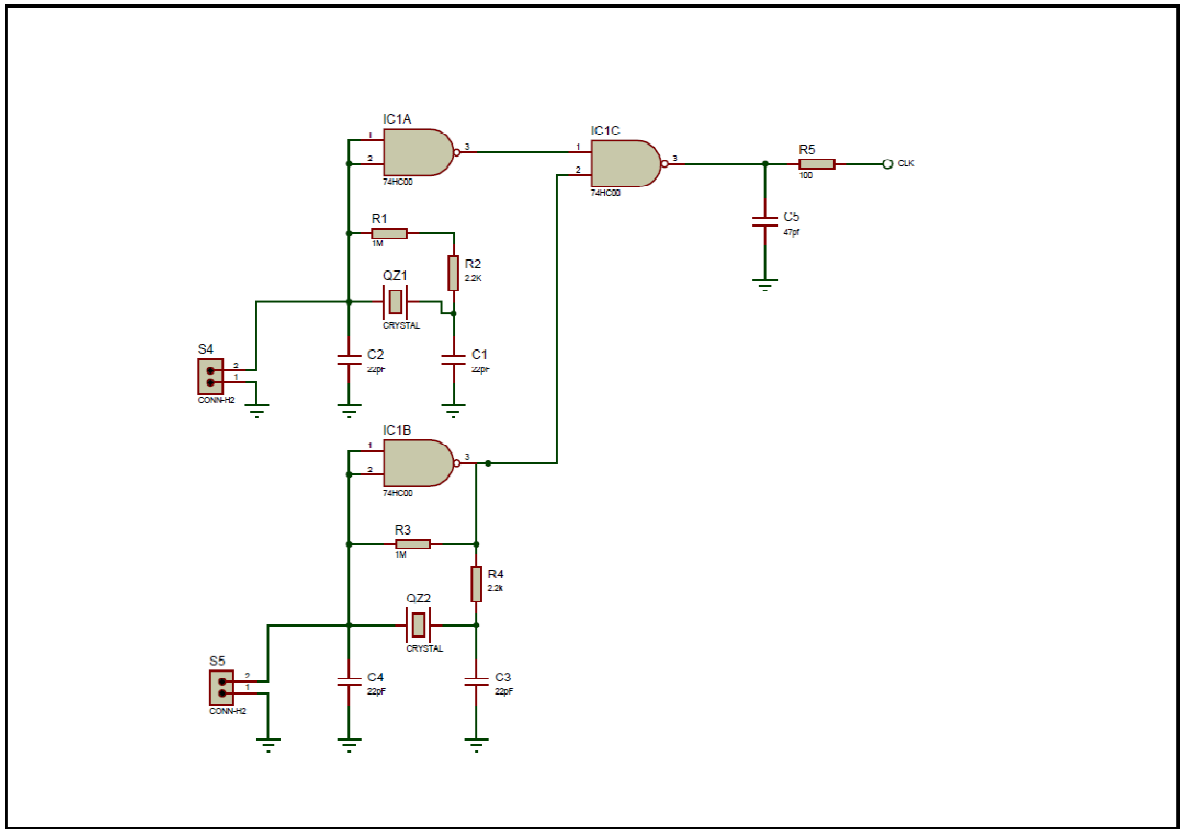


Figure [III.8] : Schéma de la partie horloge.

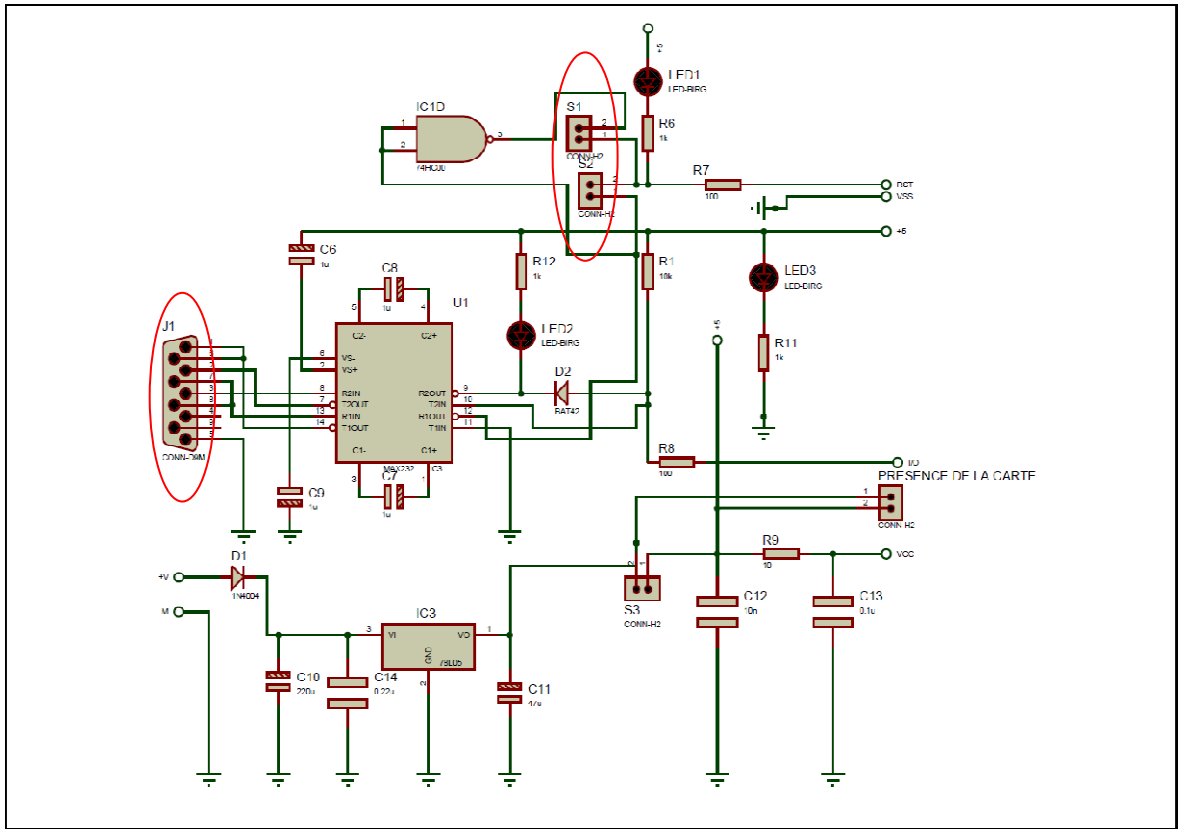


Figure [III.9] : Schéma électrique du lecteur.

III.2.2. Principe de fonctionnement :

Côté entrée/sortie de la carte, les signaux qui sortent de la carte sont appliqués directement à une des entrées d'IC2 (MAX232) qui se charge de les convertir de TTL en RS 232 pour les délivrer sur la patte 2 de J1.

Les signaux provenant de l'interface série du micro-ordinateur quant à eux sont disponibles sur la patte 3 de J1. Leur niveau est converti de RS 232 en TTL par le MAX et ils sont ensuite appliqués à la patte d'entrée/sortie de la carte, mais, pour ne pas court-circuiter les signaux sortants en cas d'erreur de protocole et de tentative d'écriture dans la carte alors que celle-ci fournit des données en sortie, la diode D2 a été prévue. Ici aussi, la LED2 permet d'indiquer l'application de signaux logiques à la carte et donc de vérifier, même si c'est assez sommaire, qu'un dialogue a bien lieu.

L'alimentation du montage est confiée à un bloc secteur externe style "prise de courant" connecté au jack J2. La diode D1 protège le montage de toute inversion de polarité éventuelle tandis que l'alimentation est régulée à 5 volts par IC3 (78L05).

En toute logique, et pour être parfaitement compatible Phoenix, notre montage devrait aussi fournir une information "présence carte" via les bornes 1 et 6 de J1. Cette information est en général obtenue à partir de l'interrupteur présent à cet effet dans tous les connecteurs de cartes ; interrupteur qui se ferme lorsque la carte est engagée à fond dans son logement. Pour notre part, nous préférons générer cette information de présence de la carte en permanence en mettant à la masse la patte 11 de IC2, ce qui permet à tous les logiciels compatibles Phoenix de fonctionner, quitte à ce qu'ils tentent de dialoguer avec une carte absente si vous avez oublié d'insérer cette dernière.

Par contre, nous utilisons l'interrupteur de détection de carte pour commuter l'alimentation 5 volts du montage, tant sur la carte que sur les différents composants actifs de ce lecteur. En procédant de la sorte, on est ainsi plus conforme à la norme qui veut que la carte ne soit alimentée et ne reçoive des signaux qu'une fois qu'elle est insérée à fond dans son lecteur. Remarquez cependant que nous avons prévu, au moyen du STRAP S3, de pouvoir court-circuiter cet interrupteur pour le cas où vous souhaiteriez alimenter en permanence le connecteur de carte afin de réaliser divers essais de résistance des cartes qui sont insérées.

III.2.3. Approvisionnement des composants :

Même s'il ne présente pas de difficulté majeure, l'approvisionnement des composants appelle un commentaire concernant le connecteur de cartes à puce. Le modèle utilisé est un type standard disponible chez de très nombreux revendeurs et se trouve être le plus souvent un

modèle ITT Canon ou Molex. Tout modèle strictement compatible, prévu pour une disposition des contacts ISO, convient aussi mais il faut vérifier bien qu'il dispose d'un interrupteur de détection de carte.

I.2.4. La réalisation du lecteur :

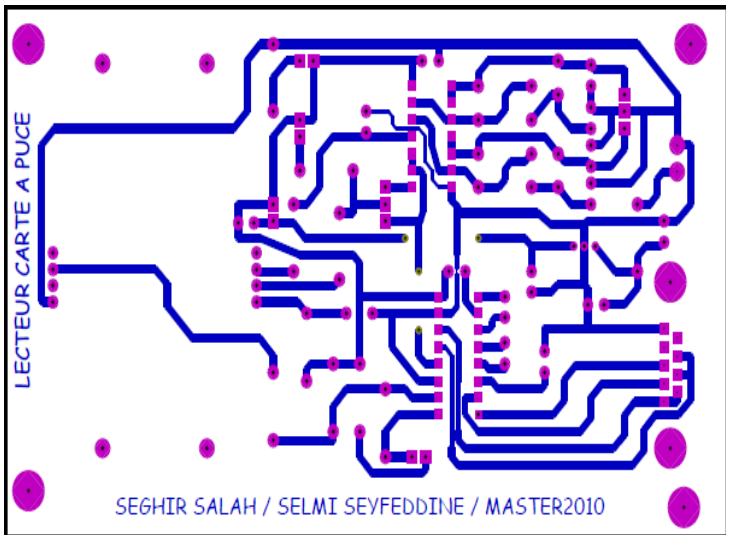


Figure [III.10] : typon (coté cuivre).

I.2.5. L'implantation des composants :

Le plan d'implantation est indiqué sur la figure (III.11). Le brochage que nous avons prévu au niveau du connecteur cartes à puce est compatible des modèles Molex et ITT Cannon. Pour le modèle Molex, il faut juste percer quatre trous pour les bossages en plastique dont est muni le connecteur. Pour le connecteur ITT Cannon, les grosses pastilles sont à l'emplacement des pions en plastique de maintient.

Dans les deux cas, il vous faudra couper les huit pattes du connecteur qui correspondent aux contacts destinés aux cartes à puce dont la puce est en position AFNOR et qui n'existent pratiquement plus aujourd'hui. Le montage des composants est à faire en suivant les indications de la figure. Les emplacements destinés aux STRAPS sont équipés de picots à souder mâles - mâles. La zone de cinq pastilles, repérées 1, 2, 3, 5, et 7 placées à côté du connecteur de carte à puce peut aussi être équipée de picots de ce type si nous l'estimons nécessaire. Cela permet en effet très facilement de connecter ensuite les sondes d'un voltmètre ou bien encore d'un oscilloscope pour examiner les signaux reçus et/ou émis par la carte, ce qui peut s'avérer très utile en phase de test.

Arrivé à ce stade des opérations, le montage peut être mis sous tension en le raccordant à un bloc secteur "prise de courant" délivrant environ 9 volts sous une centaine de mA. La LED3 doit alors s'allumer puisqu'elle signale juste la présence de l'alimentation 5 volts. Les deux autres LED peuvent être allumées ou éteintes selon l'état de la liaison série.

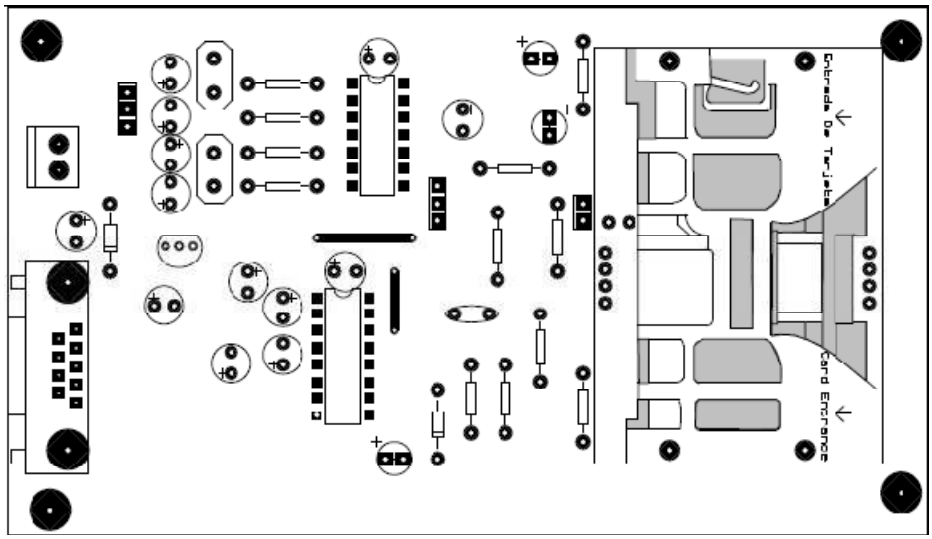


Figure [III.11] : implantations des composants (coté composant).

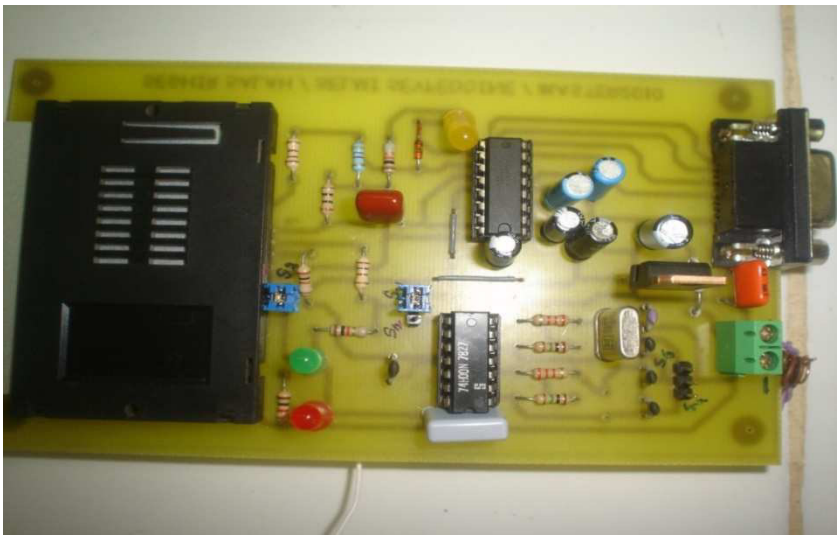


Figure [II.12] : Le lecteur Phoenix réaliser.

Pour tester le montage, il faut le raccorder au port série COM1 ou COM2 d'un PC au moyen d'un cordon DB 9 droit (c'est à dire câblé fil à fil). Il faut aussi bien sûr disposer d'un logiciel adéquat (WINPHEONIX).

Tout au plus faut-il parfois déplacer le STRAP en S1 ou S2 pour adapter la polarité du RESET appliqué à la carte. La LED1 nous aide alors beaucoup pour cela car, si elle reste allumée, cela indique que la carte est en RESET permanent et que le STRAP correspondant n'est

donc pas à la bonne place. Pour ce qui est des STRAPS d'horloge S4 et S5, il faut généralement laisse S5 en place pour les cartes à puce normales afin de leur appliquer une horloge à 3,579 MHz qui leur permet de transmettre à 9600 bits par seconde. Le fonctionnement avec S4 en place est réservé à certaines cartes particulières.

III.3. Lecture de notre carte à puce GOLD (PIC16F84+24LC16) à l'aide d'IC-prog V1.05:

Après avoir connecté le câble informatique entre le connecteur DB-9 femelle du lecteur et un port série de notre PC, fait connecter le bloc d'alimentation secteur continue 9 volt à jack J2.

- ✓ Lancer le logiciel ICPROG.EXE.
- ✓ Insérer la carte à puce à fond dans le lecteur (contact vers le haut).

La carte à puce contient en fait deux composants et donc il y a deux fichiers à sauvegarder, un pour le microcontrôleur PIC16F84 et un pour L'EEPROM 24LC16. Donc il faut lire le contenu de pic 16F84 et le contenu du l'EEPROM 24LC16.

Pour la lecture du contenu du microcontrôleur PIC16F84, il faut :

- ✓ Sélectionner le mode «JDM(PIC)» ;
- ✓ Sélectionner le mode «Phoenix 3,579 MHz».
- ✓ Sélectionner le menu «Configuration\Composant\Microchip PIC\Plus\PIC 16F84».
- ✓ Sélectionner le menu « Commande\Tout lire » pour charger à l'écran le contenu du microcontrôleur PIC16F84.

Pou la lecture du contenu de l'EEPROM 24LC16, il faut :

- ✓ Sélectionner le mode «Phoenix 3,579 MHz»
- ✓ Sélectionner le menu «Configuration\Composant\I2C EEPROM\24LC16».
- ✓ Sélectionner le menu «Commande\Tout lire» pour charger à l'écran le contenu de l'EEPROM 24LC16.

Voilà, le contenu de votre carte à puce Gold Card est lit.

Le même cas pour la carte Silver, mais il faut remplacer le PIC 16F84 par 16F876 et remplacer l'EEPROM 24LC16 par 24LC64.

Conclusion

On conclut dans cette deuxième partie pratique que notre lecteur Phoenix est capable de lire les cartes à puce Gold, Silver, et Fun.

CONCLUSION GENERALE

A travers les âges, la surveillance d'un groupe d'individus a nécessité des moyens de protection. **Garde**, on a compris qu'il fallait déterminer un périmètre de sécurité à l'intérieur du quel toute pénétration frauduleuse serait qualifiée d'intrusion.

A ce problème on a étudié au premier chapitre cette nouvelle technologie (la carte puce) et sont comportement interne ainsi que ces différents types, nous avons connu que la classification des cartes à puce est basée sur ces composants, interface et système d'exploitation utilisé, ainsi que les deux différents normes de position des contacts de la puce (AFNOR et ISO7816).

Aussi nous avons choisis deux cartes à microcontrôleur qui sont les cartes GOLD et SILVER. Pour développer ces cartes, il faut avoir un programmateur pour programmer le microcontrôleur interne dans les cartes et son lecteur.

Dans le deuxième chapitre, nous avons étudié les microcontrôleurs qui sont intégré dans la carte GOLD (16F84) et la carte SILVER (16F876) et son EEPROM.

En fin, nous avons expliqué dans la partie expérimentale la réalisation d'un programmateur de carte à puce et son lecteur Ce programmateur nous permet de programmer les PIC 16F84 et 16F876 et leurs EEPROMs et bien sûr les cartes à puce Gold et Silver. Le lecteur peut lire le contenue des cartes Gold, Silver, Fun…etc.

On propose comme perspectives pour la réalisation du programmateur et lecteur de carte à puce les points suivant :

- Connexion avec un PC sur port USB, ou port série avec un port USB au même temps.
- Réalisation d'un montage programmateur et lecteur polyvalent pour la majorité des types de carte à puce.
- On fait une étude de système de sécurité pour les cartes à puce.

Abréviation

- **AFNOR**: Association Française de Normalisation.
- **AID**: Application IDentifier.
- **APDU** : Application Protocole Data Unit.
- **ATR** : Answer To Reset.
- **CAD** : Card Acceptance Device.
- **CLK:** Clock.
- **Clrwdt :** clear watchdog.
- **CMOS** :Complementary Metal-Oxide Semi-conducteur.
- **CPU**: Central Processing Unit.
- **DIL :** Dual In Line.
- **E/S :** entrée/sortie.
- **EEPROM**: Electronic Erasable Programmable Read Only Memory.
- **GND**: Ground (Electric).
- **GSM**: Global System for Mobile Communication.
- **I/O :** input /output.
- **ISC :** Complex Instructions Set Construction.
- **ISO**: International Organization for Standardization.
- **MOS :** Metal-Oxide Semi-conducteur.
- **PIC : P**eripheral **I**nterface **C**ontrôler.
- **PROM**: Programmable Read Only Memory.
- **PTS:** Protocol Type Selection.
- **RAM:** Random Access Memory.
- **RFU:** Reserved for Futur Use.
- **RISC:** Reduced Instructions Set Construction.
- **ROM**: Read Only Memory.
- **RST**: Reset.
- **SCQL**: Structured Card Query Language.
- **SFR :** Special Function Registers.
- **SIM**: Subscriber Identification Humber.
- **TPDU** : Transmission Protocol Data Unit.
- **UAL :** unité arithmétique et logique.
- **USB:** Universel Sériel Bus.
- **VCC:** supply voltage.

Annexe

Liste des cartes à puce :

Nom commun carte----------Nom courant--------Type de puce

_____

PIC

_____

WAFER----------16C84-16F84-16F84A

GOLD----------PICCARD 1----------16F84/16F84A + 24LC16

BLUE----------16F84A + 24LC64

CANARY----------16F628 + 24LC16

EMERAUDE----------16F628 + 24LC64

SILVER----------16F876/16F877 + 24LC16

SILVER 2----------PICCARD 2----------16F876/16F877 + 24LC64

SILVER 3----------GREEN----------16F876/16F877 + 24LC128

SILVER 4----------GREEN 2----------16F876/16F877 + 24LC256

GREEN----------SILVER 3----------16F876/16F877 + 24LC128

GREEN 2----------SILVER 4----------16F876/16F877 + 24LC256

SINGLE PIC----------16F876-16F627-16F628

PICCARD 1----------GOLD----------16F84 + 24LC16

PICCARD 2----------SILVER 2----------16F876/16F877 + 24LC64

_____

ATMEL

_____

JUPITER----------PINK----------AT90S2343 + 24C16

JUPITER 2----------AT90S8535 + 24C64

X-CARD----------AT90S8535 + 24LC512

PURPLE----------FUN CARD 2----------AT90S8515A + 24C64

PURPLE 4A----------FUN CARD 3----------AT90S8515A + 24C128

PURPLE 4B----------FUN CARD 4----------AT90S8515A + 24C256

FUN CARD 2----------PURPLE----------AT90S8515A + 24C64

FUN CARD 3----------PRUSSIAN----------AT90S8515A + 24C128

FUN CARD 4----------PRUSSIAN 2----------AT90S8515A + 24C256

FUN CARD 5----------PRUSSIAN 3----------AT90S8515A + 24C512

FUN CARD 6----------PRUSSIAN 4----------AT90S8515A + 24C1024

FUN CARD 7----------PRUSSIAN 5----------AT90S8515A + 2 X 24C1024

ATMEGA 161----------ATM161 + 24C64

ATMEGA 163----------ATM163 + 24C256

ATMEGA 8515----------ATM8515 + 24C256

ATMEGA 128----------BLACKCARD----------ATM128 + 24C256

TITANIUM CARTE BLEUE

SMARTCARD------: ISO 7816-2

ARCHITECTURE---: AVR RISC

FLASH----------: 32KB (LIBRE 28 KB)

EPROM----------: 32 KB

RAM------------: 1024 BYTE

CRYPTO---------: OUI (RSA)

PROTOCOL-------: T0, T1, TE

LANGUAGE-------: ASM

PROGRAMMATION--: 3, 57 MHZ

TITANIUM 2

SMARTCARD

ARCHITECTURE

FLASH----------: 64 KB (LIBRE 28 KB)

EEPROM---------: 64 KBYTE

RAM

CRYPTO---------: OUI (RSA)

PROTOCOL-------: TO, T1, TE

LANGUAGE-------: ASM (C)

PROGRAMMATION--: 3,57 MHZ

Low-power, high-performance, 8-/16-bit secure microcontroller with 64 Kbyte FLASH and 64 Kbyte EEPROM. Security Features: OTP (one time programmable) EEPROM area, RNG (Random Number Generator), "out of bounds" detectors, side channel attack countermeasures. SPI port in addition to ISP ports Hardware DES/TDES, 16-bit RISC Co-processor, CRC, Common Criteria EAL4 target.

PLATINIUM CARD

SMART CARD-----------: ISO 7816

MICROCONTROLLEUR-----: AT90SC6464C

FLASH----------------: 64 KB

EEPROM---------------: 64KB

CRYPTO---------------: DES-PKI

COMPATIBLE-----------: GSM, 3GPP + EMV

TENSION UTILISATION--: 2,7 A 5,5 V

SECURITE-------------: MMU, MED, OPT, RNG, ACM

HARDDWARE------------: DE, VL3A

Nomenclature des composants (Programmateur):

- **Semi-conducteursA**

  T1, T2 : BC 547.

  T3 : BC 557.

  D1, D2, D3, D4 :1N 914 ou 1N 4148.

  DZ1: Zener 5.1 volts, 0.4 watt.

  DZ2: Zener 8.2 volts, 0.4 watt.

  LED1: LED rouge.

  LED2: LED verte.

- **Résistances ¼ de watt 5 % :**

  R1 : 10 kOhms (marron, noir, orange).

  R2: 1.5 kOhms (marron, vert, rouge).

  R3: 100 kOhms (marron, noir, jaune).

  R4, R5, R6 : 1 kOhm (marron, noir, rouge).

- **Condensateurs :**

  C1 : 22 µF 25 volts, radial.

  C2 : 100 µF 25 volts radial.

- **Divers :**

  Supports de CI : 1 x 28 pattes étroit, 1 x 18 pattes, (tous à contacts tulipes).

  Support ZIF 28 pattes étroit + 1 x 28 pattes étroit à contacts tulipes (optionnels).

  J1 : Connecteur DB 9 femelle, coudé à souder sur circuit imprimé.

  J2 : Connecteur pour carte à puce ISO format ID 1, ITT Canon, Molex, etc.

Nomenclature des composants (Lecteur):

- **Semi-conducteurs :**

  IC1 :74HC00.

  IC2 : MAX 232 ou ICL 232.

  IC3 : 78L05.

  D1 : 1N 4004.

D2 : BAT 41, BAR 28, etc. diode Schottky (impératif).

LED1, LED2, LED3: couleur au choix.

- **Résistances ¼ de watt 5 % :**

R1, R3 : 1 MOhm (marron, noir, vert).

R5, R7, R8 : 100 Ohms (marron, noir, marron).

R6, R11, R12 : 1 kOhms (marron, noir, rouge).

R9 : 10 Ohms (marron, noir, noir).

R10 : 4.7 kOhms (jaune, violet, rouge).

- **Condensateurs :**

C1, C2, C3, C4 : 22 pF céramique.

C5 : 47 pF céramique.

C6, C7, C8, C9 : 1 µF 25 volts chimique radial.

C10 : 220 µF 25 volts chimique radial.

C11 : 47 µF 25 volts chimique radial.

C12 : 10 nF céramique.

C13: 0.1 µF polyester, Mylar, MKT.

C14: 0.22 µF polyester, Mylar, MKT.

- **Divers :**

QZ1 : Quartz 3.579 MHz en boîtier HC18U.

QZ2 : Quartz 6.00 MHz en boîtier HC18U.

J1 : Prise DB9 femelle coudée à 90 ° pour circuit imprimé.

J2 : Jack femelle 2.1 mm pour circuit imprimé.

J3 : Connecteur pour carte à puce ISO format ID 1 (normal) avec interrupteur de détection de présence de carte, ITT Cannon, Molex ou compatible.

Supports de circuits intégrés : 1 x 14 pattes, 1 x 16 pattes.

Picots mâles - mâles au pas de 2.54 mm: 1 x 2, 2 x 3, 1 x 5.

Présentation du Logiciel ISIS7

GÉNÉRALITÉS

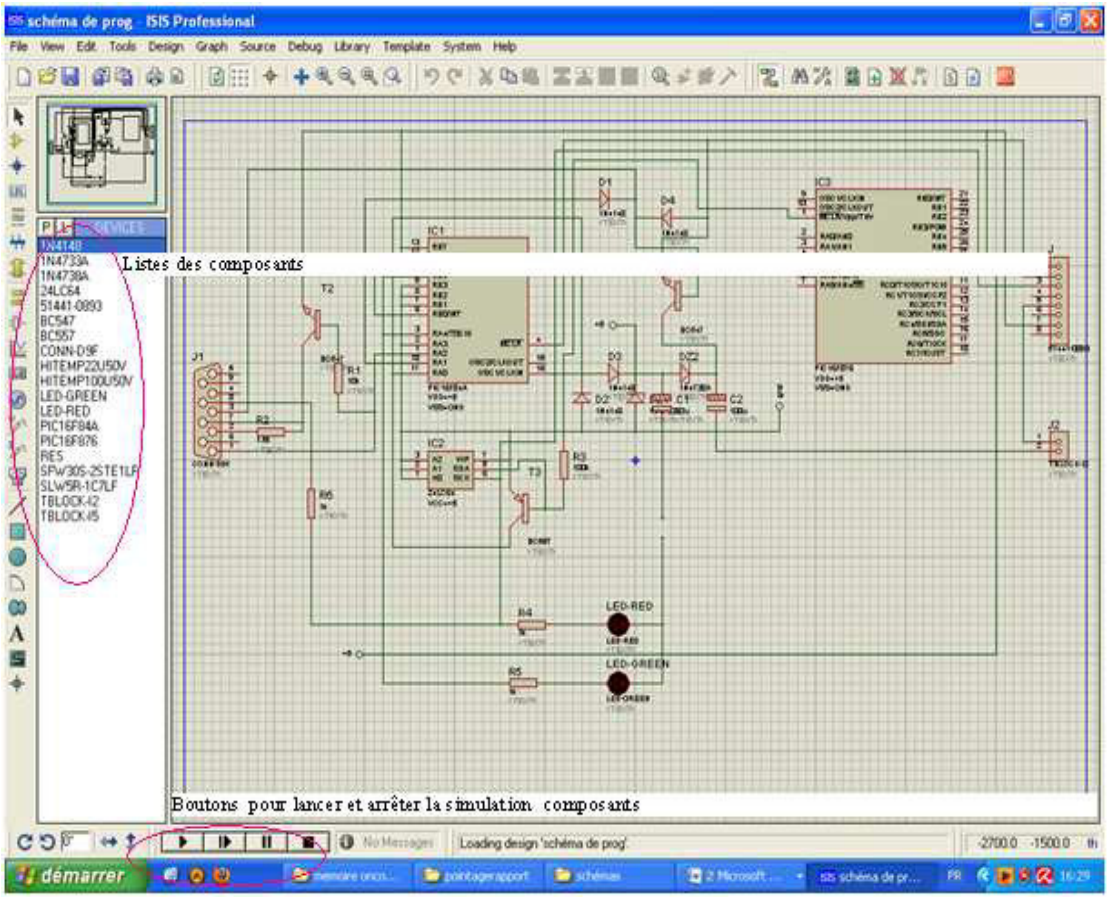
ISIS est un éditeur de schémas qui intègre un simulateur analogique, logique ou mixte. Toutes les opérations se passent dans cet environnement, aussi bien la configuration des différentes sources que le placement des sondes et le tracé des courbes.

Dans le cadre de ce projet, c'est la simulation numérique qui a sollicité notre intérêt.

La simulation analogique

Saisir un schéma

Pour saisir le schéma, il faut créer un nouveau projet puis placer les composants qui doivent être sélectionné à partir de la bibliothèque des composants :



Bibliographie

[1]. Christian Tavernier, *Les Cartes à puce: théorie et mise en œuvre, 2ème édition, Ed. Dunod, 2007.*

[2]. *JEAN Pierre TUAL.* *Introduction à la carte à puce. Avril2004.*

[3]. *SUMIT DAHR.* *Introduction to smart cards. Octobre 2003.*

[4]. *NICOLAS Droze, Jean-Noël ISNARD.* *Carte à puce et Programmation.*

[5]. Patrick GUEULLE. *PC et carte à puce. 1er édition .Ed Dunod 1998.*

[6]. Patrick GUELLE. *Plus loin avec les cartes à puce. Ed Dunod. Paris 2004.*

[7]. Programmation des PIC, Première partie-PIC16F84-Révision 5, par BIGONOFF.

[8]. Programmation des PIC, Seconde partie-PIC16F876-/77 Révision 7, par BIGONOFF.

[9]. Electronique pratique. *Juin 2002. N:267. Page 33-42.*

[10]. Electronique pratique. *Réalisez vos cartes. Juin 2002.N:266.*

[11]. PIC16F8X, Data Sheet. *Document DS30430C.*

[12]. PIC16F84, Data Sheet. *Document DS35007A.*

[13]. PIC 16F87X Data Sheet, *document DS30292C.*

Sites d'internet

[S1]. *www.carteapuce.fr/carte -cold.htm.*

[S2]. *www.carteapuce.fr/carte-silver.htm.*

[S3]. *http://fr.wikipedia.org/wiki/Carte_à _puce/.*

[S4]. *http://www.cardwerk.com/smartcards/.*

[S5]. *http://eurekaweb.free.fr/ih1-carte_a_puce.htm.*

[S6]. *http://www.microchip.com/1010/suppdoc/.*

[S7]. *http://fribotte.free.fr/bdtech/cours/pic16f84/.*

[S8]. ***http://www.abcelectronique.com/bigonoff/organisation.php?2654c.***

[S9]. ***www.carteapuce.fr//realisation-prog-gold-silver.htm.***

[S10]. ***www.carteapuce.fr//realisation-lecteurs_phoenix.htm.***

[S11]. ***http://www.vulgarisation-informatique.com/memoire.php***

Chapitre I

Chapitre II

Chapitre III

Partie I

Partie II

Annexes

Introduction Générale

Conclusion Générale