République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 8Mai 1945 – Guelma
Faculté des Sciences et de la Technologie
Département d'Electronique et Télécommunications

Mémoire de fin d'études
pour l'obtention du diplôme de Master Académique

Domaine : **Sciences et Technologie**
Filière : **Electronique**
Spécialité : **Systèmes Electroniques**

Etude matérielle et logicielle d'une carte d'acquisition de signaux lents à base d'un microcontrôleur PIC 16F88

Présenté par :

Chahira Bourara

Sous la direction de :

Abdelaziz Younsi

Mai 2014

Remerciement:

Je ne peux conclure ce mémoire sans montrer ma sincère gratitude et mon grand remerciement à mon promoteur Mr A/A Younsi pour son dévouement et ses conseils bien constructifs.

J'adresse également mes remerciements à :

- Mon mari Youcef pour sa présence à mes côté.
- Mon frère A ghani pour son aide bien aimable.
- Mes parents et ma famille frères et sœurs pour leurs encouragements et soutiens.

Sommaire :

CHAPITRE II : Mise en forme et acquisition des données

CHAPITRE III : Etude et réalisation de la carte d'acquisition

Introduction

INTRODUCTION GENERALE :

Le progrès de la microélectronique et de l'informatique ont apporté une nouvelle dimension à l'acquisition des données.
Le domaine numérique et informatique ont connu depuis la dernière décennie une croissance rapide. Les prix et les dimensions des composants électroniques ont diminué, cela a permis la conception de circuits de plus en plus complexe, dont l'architecture est conçue de manière à optimiser les traitements numériques.

Dans le travail que nous proposons dans ce mémoire consiste à étudier et élaborer une carte d'acquisition destinée à un usage médical, entre autres le montage électronique est réalisé autour d'un microcontrôleur et aura pour fonction de recueillir et traiter des signaux lents (température, pression,...etc.) qui sont par la nature analogique, ce traitement est en fait une acquisition, dans le cas échéant, une amplification, une conversion A/N, et un traitement numérique des données.

Les cartes d`acquisitions sont conçues pour répondre à une ou plusieurs fonctions, elles peuvent se suffire à elle-même dans le cas où toutes les fonctions sont embarquées sur le même circuit ou être extensibles à l'aide des cartes additionnelles qui sont rajoutées à l'aide de bus.

Dans le domaine du traitement de l'information, les convertisseurs Analogique/ Numérique ont une grande utilité. Les tests de ces composants toujours plus performants deviennent de plus en plus sophistiqués. C'est pourquoi l'utilisation d'outils puissantes et conviviats, tel que l'informatique est devenu nécessaire.

L'objectif de notre projet de fin d'étude est la réalisation d'une carte d'acquisition qui puisse fonctionner sur PC. A ce titre, nous proposons l'étude et

la réalisation d'une carte d'acquisition 7 voies analogiques pour des signaux lents, les résultats de conversion sont affichés sur PC via la liaison RS 232

Ce rapport est composé de cinq chapitres. Après une brève introduction sur le développement des cartes numériques d'acquisition dans le premier chapitre, nous rappelons dans le deuxième chapitre quelques notions principales des microcontrôleurs en général et du pic 16F88, qui est utilisé dans notre montage en particulier.
Nous développons ensuite dans le troisième chapitre nous développons la mise en forme et l'acquisition des données, la conversion analogique ainsi que la gestion de RS232.

Le quatrième chapitre aborde la carte d'acquisition 7 voies analogiques pour les signaux lents, son schéma électrique, son principe de fonctionnement, le protocole de communication avec le PC et la simulation avec le logiciel de simulation ISIS.

Une conclusion vient finalement pour conclure notre travail.

Chapitre 1

I. INTRODUCTION : Du microprocesseur au microcontrôleur :

La découverte du microprocesseur date aujourd'hui de prés de quarante ans, en effet la fabrication du premier circuit commence en 1970, année où la société INTEL met au point le premier microprocesseur le 4004. On n'imagine pas à l'époque que cette révolution industrielle donnera naissance à l'ordinateur individuel. Depuis, leur puissance de calcul et l'intégration des transistors les constituants n'ont cessé d'évoluer. On retrouve désormais les microprocesseurs dans la plupart des applications.

Les microprocesseurs ne sont jamais employés seuls, des circuits périphériques leur sont toujours associés pour pouvoir être intégrés au sein d'une application (figure 01).

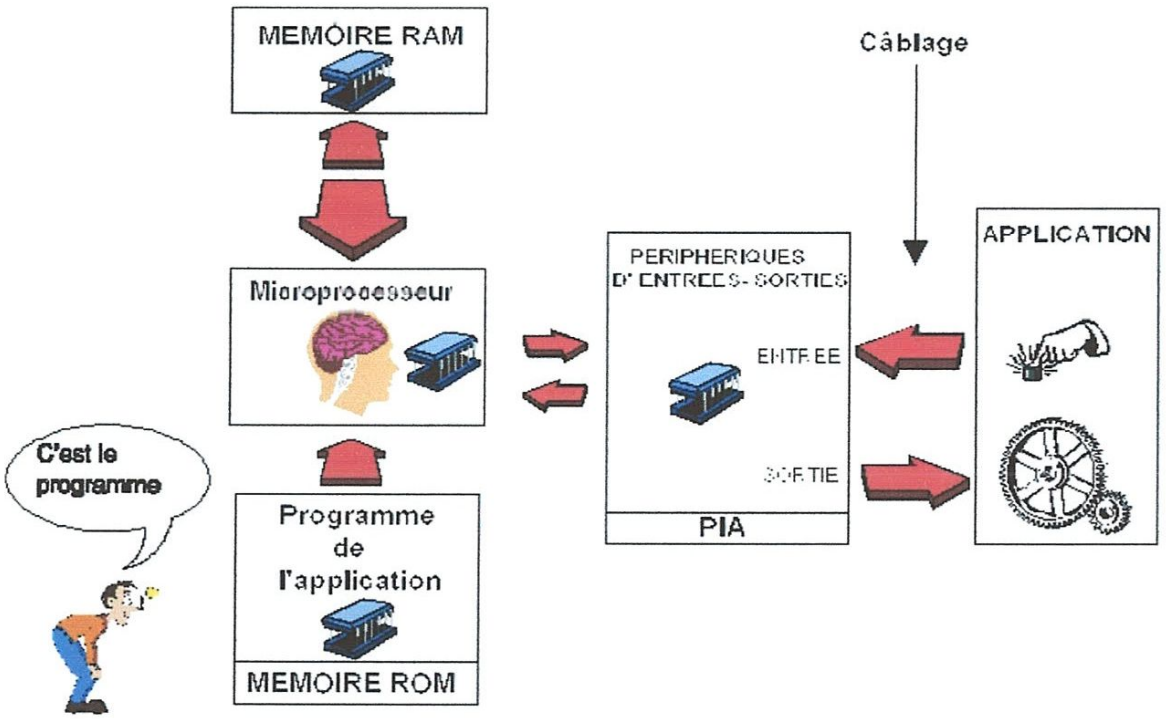


Figure 1.1 : les microprocesseurs sont associés avec des circuits périphériques pour pouvoir être intégrés au sein d'une application.

Un des avantages d'un montage à base de microprocesseur par rapport à un montage en logique câblée, est sa souplesse d'emploi, en effet nous entrons dans le domaine de la logique programmable où le fonctionnement du montage dépend d'un programme logé dans une mémoire, celui-ci peut être modifié pour changer les équations régissant l'application, sans toutefois entraîner de changement au niveau du câblage des entrées sorties.

II. LES MICROCONTROLEURS :

Les microprocesseurs comme nous venons de le voir possèdent un indéniable avantage sur la logique câblée, en effet pour modifier le fonctionnement d'une application il suffit de modifier le programme sans refaire le câblage.
Les microcontrôleurs possèdent quant à eux la puissance d'un microprocesseur mais ont un atout en plus, ils possèdent dans le même boitier, les périphériques intégrés (figure 2).
Cela veut dire que le programme de l'application est à l'intérieur et non dans un circuit mémoire externe et que les périphériques d'entrées-sorties sont également intégrés, ce qui fait l'économie de nombreux circuits périphériques.

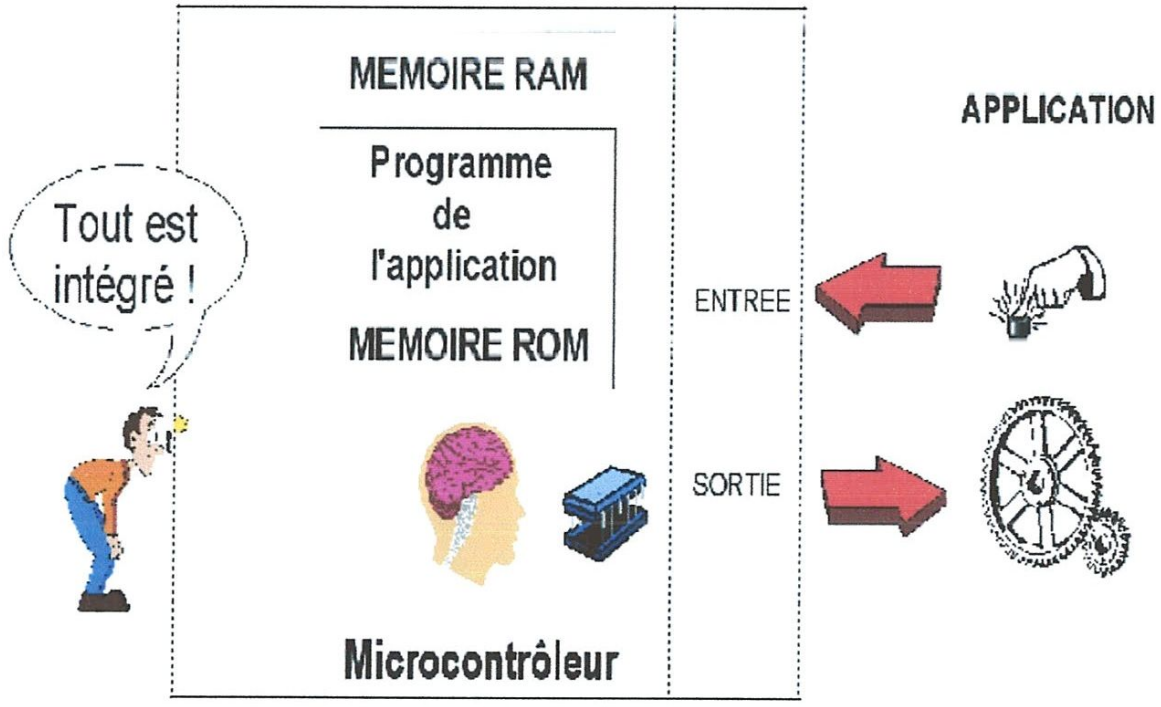


Figure 1.2 : les microcontrôleurs sont des circuits intégrés qui contiennent dans le même boitier, outre le microprocesseur, les circuits périphériques permettant de réaliser des montages simples sans nécessiter l'ajout de composants annexes.

Cette caractéristique fait que les montages deviennent encore plus simples et la programmation plus aisée (un système à base de microprocesseur oblige le concepteur à réaliser un décodage d'adresse pour permettre au microprocesseur de ne dialoguer qu'avec un seul périphérique à la fois).
Un microcontrôleur seul peut donc gérer une application sans faire appel forcement à d'autres circuits associés.

Donc c'est une unité de traitement et d'exécution de l'information à laquelle on a ajouté des périphériques internes permettant de réaliser des montages sans

nécessiter l'ajout de composants annexes. Un microcontrôleur peut donc fonctionner de façon autonome après programmation.

II.1. Les microcontrôleurs PIC :

Les **microcontrôleurs PIC** (ou **PICmicro** dans la terminologie du fabricant) forment une famille de microcontrôleur de la société MICROCHIP. Ce microcontrôleur encore très utilisé à l'heure actuelle est un compromis entre simplicité d'emploi et prix de revient. Il fait partie de la famille des circuits RISC (Reduced Instruction Set Computer) c'est-à-dire un processeur à jeux d'instructions réduit.

Plus on réduit le nombre d'instructions, plus facile et plus rapide en est le décodage, et plus vite le composant fonctionne. Cependant, il faut plus d'instructions pour réaliser une opération complexe.

Avec un quartz de 4 MHz relié au PIC, on obtient donc pour l'horloge interne, 1 000 000 de cycles par seconde. Or, comme le PIC exécute pratiquement 1 instruction par cycle, hormis les sauts, cela donne une puissance de l'ordre de 1 MIPS (1 million d'instructions par seconde).

Il existe de nombreuses versions de PIC possédant chacune des caractéristiques différentes. Connaitre ces versions permet de choisir le PIC le plus adéquat par rapport à l'application envisagée.

II.2. Pourquoi choisir un PIC ?

- Les performances sont identiques à ses concurrents.
- Les prix sont les plus bas du marché.
- Très utilisé donc très disponible.
- Les outils de développement sont gratuits et téléchargeables sur le WEB.
- Le jeu d'instruction réduit est souple, puissant et facile à maîtriser.
- La communauté des utilisateurs des PICs est très présente sur le WEB. On trouve sur le net quasiment tout ce dont on a besoin, tutoriaux pour démarrer, documents plus approfondis, schémas de programmeurs avec les logiciels qui vont avec, librairies de routines, forums de discussion . . .

II.3. Structure d'un PIC :

Les PIC, au même titre que les microprocesseurs, sont composés essentiellement de registres ayant chacun une fonction bien définie. Les PIC possèdent également des périphériques intégrés, tels qu'une mémoire EEPROM, un

timer, des ports d'entrées/ sorties ou bien encore un convertisseur analogique/numérique.
Selon le type de PIC utilisé, on retrouvera en interne un certain nombre de registres et périphériques possédant des caractéristiques différentes. Les différences de caractéristiques selon le PIC utilisé sont :

- La taille de la RAM interne ;
- La mémoire EEPROM intégrée ;
- Le type de mémoire programme : FLASH, EPROM ou OTP (PROM) et la taille de celle-ci.
- Le timer intégré ;
- Les convertisseurs analogique/numérique intégrés.

II.4. Structure minimale d'un PIC :

La structure minimale d'un PIC est constituée des éléments ci-dessous (Figure 3) :

- **Une mémoire de programme** contient le code binaire correspondant aux instructions que doit exécuter le microcontrôleur. La capacité de cette mémoire est variable selon les PIC.

- **Une mémoire RAM** (Random Access Mémory « Mémoire de lecture seule ») sauvegarde temporairement des données. Sa capacité est aussi variable selon les PIC. Il est à noter que le contenu d'une RAM n'est sauvegardé que pendant la phase d'alimentation du circuit.

- **Une Unité Arithmétique et Logique** (UAL ou ALU en anglais) qui est le cœur du système, elle est chargée d'effectuer toutes les opérations arithmétiques de base (addition, soustraction, etc.) ainsi que les opérations logiques de base (ET, OU logique, etc.).

- **Des ports d'entrées/sorties** permettent de dialoguer avec l'extérieur du microcontrôleur, par exemple pour prendre en compte l'état d'un interrupteur (entrée logique), ou encore pour commander un relais (sortie logique).

- **Des bus internes** permettent la communication entre les différents éléments intégrés au microcontrôleur.

- **Un registre compteur de programme** (CP ou PC en anglais), est chargé de pointer l'adresse mémoire courante contenant l'instruction à réaliser par le microcontrôleur.
  Le contenu du registre PC évolue selon le pas de programme.

- **Un registre pointeur de pile** (PP ou SP en anglais) est essentiellement utilisé lorsque l'on réalise un sous-programme. Le pointeur de pile est chargé de mémoriser l'adresse courante que contient le compteur de programme avant le saut à l'adresse du sous programme.
  Lorsque ce dernier est terminé, le pointeur restitue l'adresse sauvegardée vers le compteur de programme.

- **Un registre d'instruction** contient tous les codes binaires correspondant aux instructions à réaliser par le microcontrôleur.

- **Un registre d'état** qui est en relation avec l'UAL et permet de tester le résultat de la dernière opération effectuée par le microcontrôleur. Selon la dernière opération effectuée, des bits sont positionnés dans le registre d'état et ceux-ci peuvent être testés à l'aide d'une instruction de branchement pour effectuer des sauts conditionnels.

- **Une horloge système** permet de cadencer tous les échanges internes ou externes au microcontrôleur.

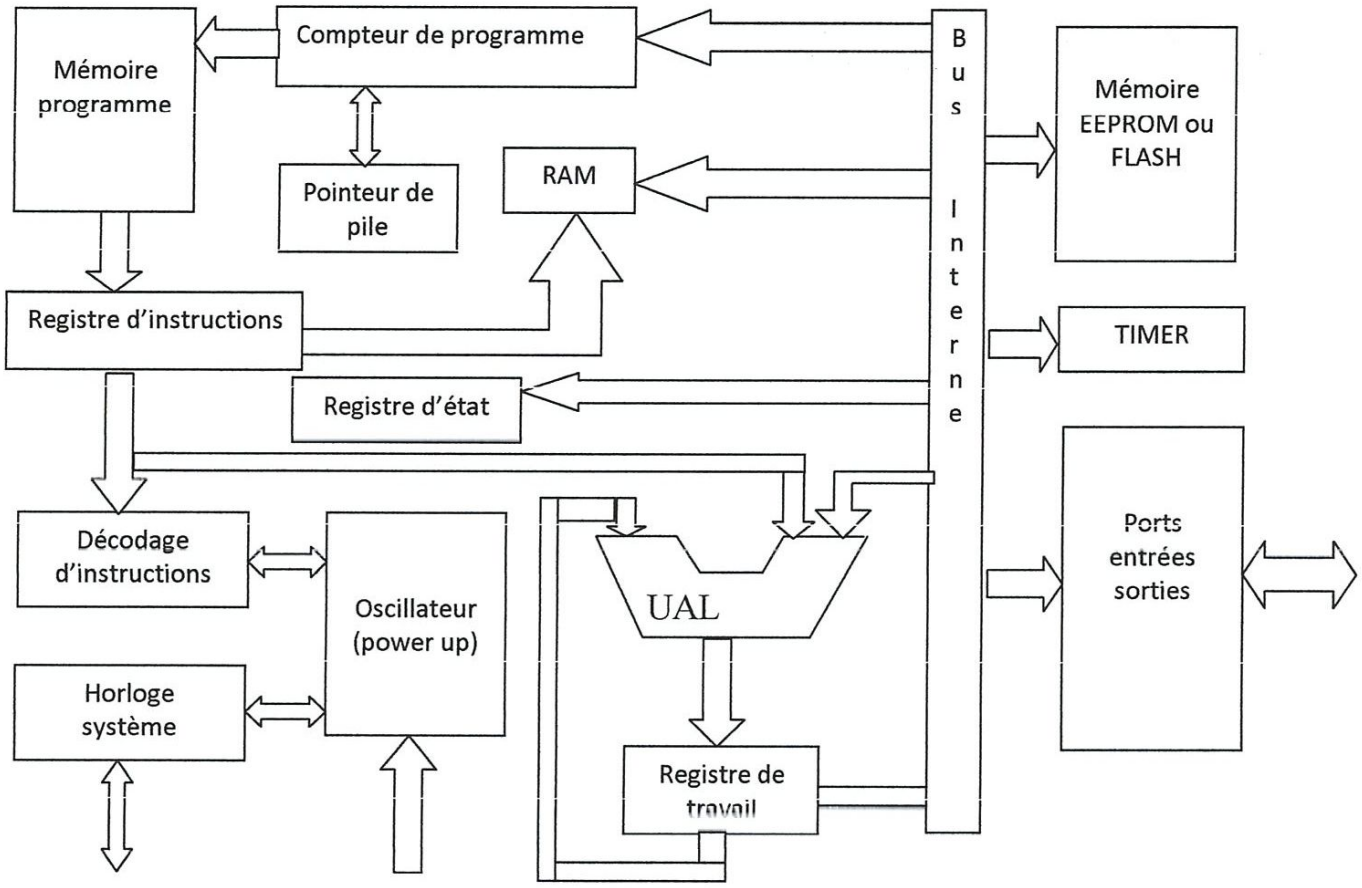


Figure 1.3 : Structure générale d'un PIC

II.5. Programmation des PIC :

Les PICs sont des composants qui ne sont capables de rien tant qu'on ne leur a pas fait intégrer un programme. C'est comme un ordinateur sans système d'exploitation : il ne démarre pas. Le programme que l'on doit mettre dans le PIC se présente sous la forme d'un fichier binaire (par oppositions à un fichier texte) comportant toutes les instructions à exécuter.

L'ensemble de ces instructions est nommé **jeu d'instructions**. Ces instructions machines permettent d'effectuer des opérations élémentaires (addition, ET logique…) ou plus complexes (division, passage en mode basse consommation…). Le jeu d'instruction définit quelles sont les instructions supportées par le

processeur et précise aussi quels sont les registres du processeur manipulable par le programmeur.

Chaque instruction machine contient souvent un code qui permet de préciser quelle est l'instruction : s'il s'agit d'une addition, d'une multiplication, d'une mise en veille du processeur, d'un branchement, etc.

Ce code est parfois associé à des opérandes. Ces opérandes peuvent être des nombres entiers, des adresses, des identifiants de registres, etc.

Les PIC 16F84A, 16F628A, 16F88, 16F876A, 16F886 (famille Mid-range) ont le même jeu d'instructions, constitué de seulement **35** instructions (architecture RISC)

II.5.1. Organisation des instructions - Les types d'instructions

On constate qu'il existe 4 types d'instructions (voir Annexe) :

A. Les instructions « orientées octet »

Ce sont des instructions qui manipulent les données sous forme d'octets. Elles sont codées de la manière suivante :

- 6 bits pour l'instruction, car comme il y a 35 instructions, il faut 6 bits pour pouvoir les coder toutes
- 1 bit pour indiquer si le résultat obtenu doit être conservé dans le registre de travail de l'unité de calcul (W pour Work) ou sauvé dans l'opérante (F pour File).
- Reste 7 bits pour encoder l'opérande (File)

B. Les instructions « orientées bits »

Ce sont des instructions destinées à manipuler directement des bits d'un registre particulier. Elles sont codées de la manière suivante :

- 4 bits pour l'instruction (dans l'espace resté libre par les instructions précédentes)
- 3 bits pour indiquer le numéro du bit à manipuler (bit 0 à 7 possible).
- et de nouveau 7 bits pour indiquer l'opérande.

C. Les instructions « orientées manipulation d'une constante de 8 bits »

Ce sont les instructions qui manipulent des données qui sont codées dans l'instruction directement. Elles sont codées de la manière suivante :

- L'instruction est codée sur 6 bits
- Elle est suivie d'une valeur IMMEDIATE codée sur 8 bits (donc de 0 à 255).

D. Les instructions pour les sauts et appels de sous-routines

Ce sont les instructions qui provoquent une rupture dans la séquence de déroulement du programme. Elles sont codées de la manière suivante :

- Les instructions sont codées sur 3 bits
- La destination codée sur 11 bits

II.5.2 Le logiciel MPLAB :

Le logiciel **MPLAB** est un outil de développement pour programmer des microcontrôleurs de type PIC de la famille Microchip.
Il est mis au point par la société Microchip, et est entièrement gratuit. Ce logiciel vous permettra de créer un programme, de l'assembler, et de le simuler. Enfin, vous pourrez transférer votre programme réalisé sous MPLAB pour le mettre sur votre PIC.

Caractéristiques

Avec l'environnement de MPLAB il est possible de réaliser un fichier source en langage assembleur (fichier .asm). Mais l'avantage de MPLAB c'est de réaliser des programmes en langage C. Ainsi MPLAB peut utiliser un langage de programmation évolué pour le développement en électronique.

Après avoir réalisé le programme d'un fichier source en assembleur ou en c, il est possible de transformer ce dernier en fichier .hex. Ça le rend prêt à être chargé dans le microcontrôleur.

III. LE CHOIX DU MICROCONTROLEUR :

Le choix du microcontrôleur est primordial, car c'est de lui que dépendent en grande partie les performances, la taille, la facilité d'utilisation et le prix du montage.

Le PIC 16F88 possède en plus des instructions très puissantes, donc un programme à développer réduits, une programmation simple grâce au mode série. En fait la cause principale du choix du 16F88 est qu'il dispose de l'option du convertisseur A/D pour satisfaire côté acquisition.

III.1. Le microcontrôleur PIC 16F88 :

Nous allons maintenant s'intéresser à la structure interne du PIC 16F88, avec lequel nous avons travaillé.

Le PIC 16F88 est un microcontrôleur du MICROCHP, il fait partie intégrante de la famille des MID-RANGE(16) dont la mémoire programme et de type Flash(F) de type 88 et capable d'accepter une fréquence d'horloge maximale de 20 MHz. (figure 4).

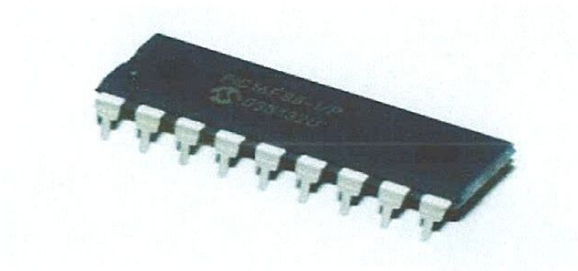


Figure 1.4 : PIC 16F88

III.1.1. Schéma fonctionnel du pic 16F88 :

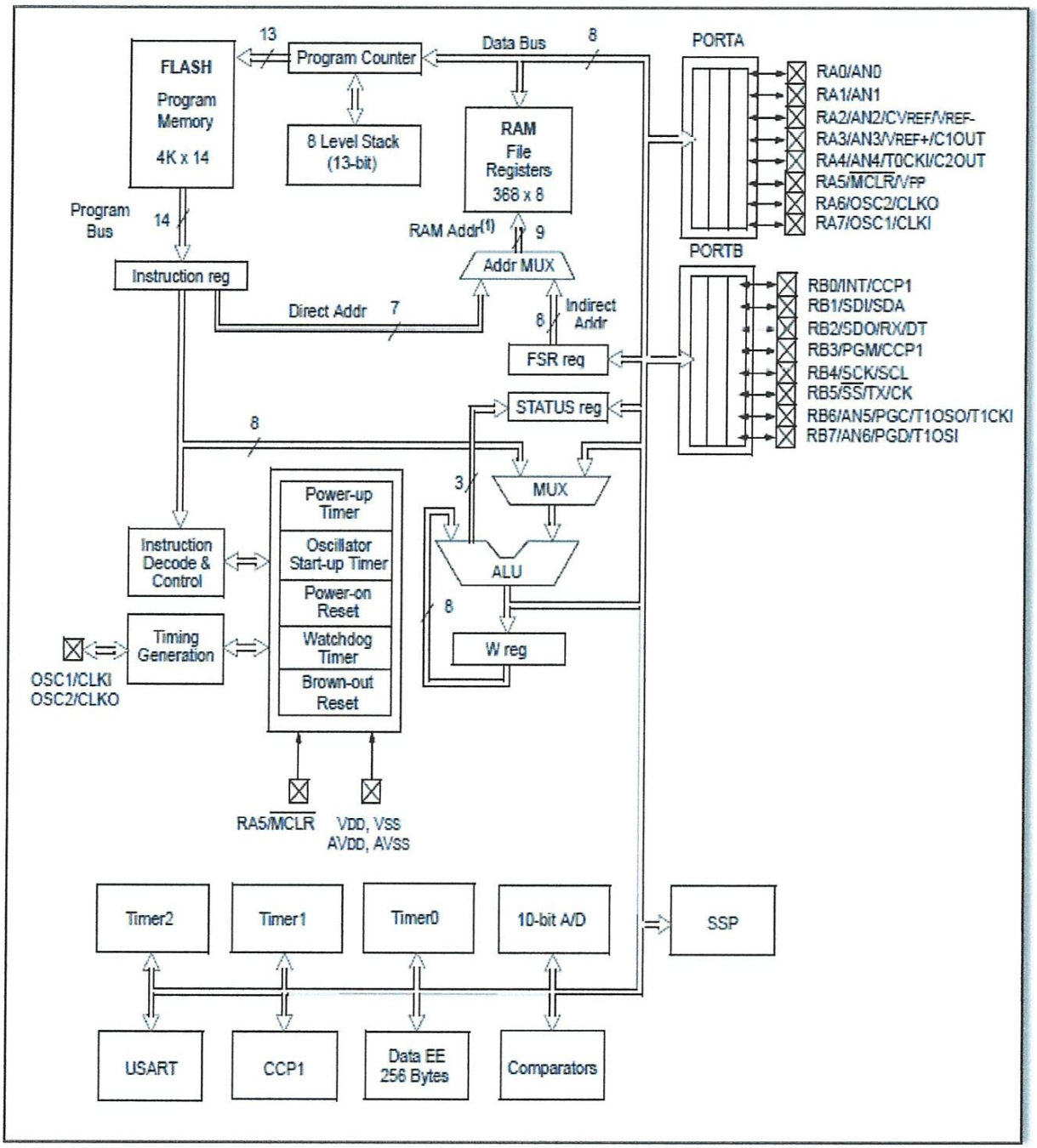


Figure 1.5 : la structure interne du PIC 16F88

III.1.2. Brochage du PIC 16F88 :

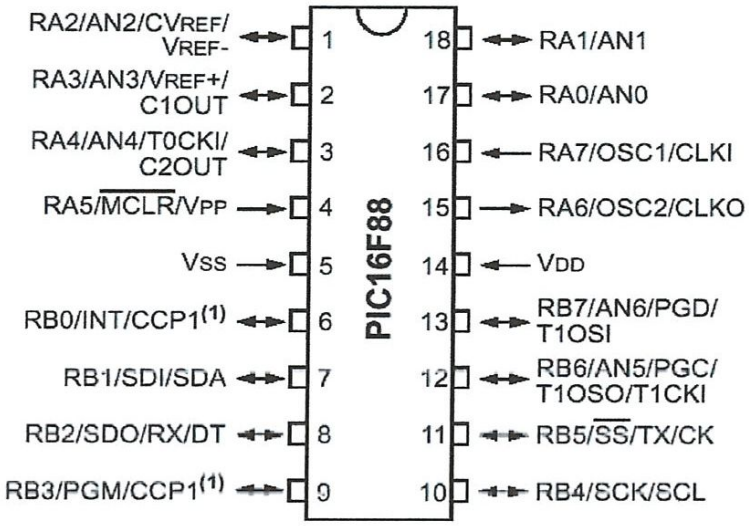


Figure 1.6 : Brochage du 16F88

III.1.3. Organisation du 16F88 :

La mémoire du 16F88 est divisée en 3 parties Pour mieux le connaître regardons de près son schéma fonctionnel.

A. la mémoire programme :

La mémoire programme est constituée de 14 bits. C'est dans cette zone que nous allons écrire notre programme.

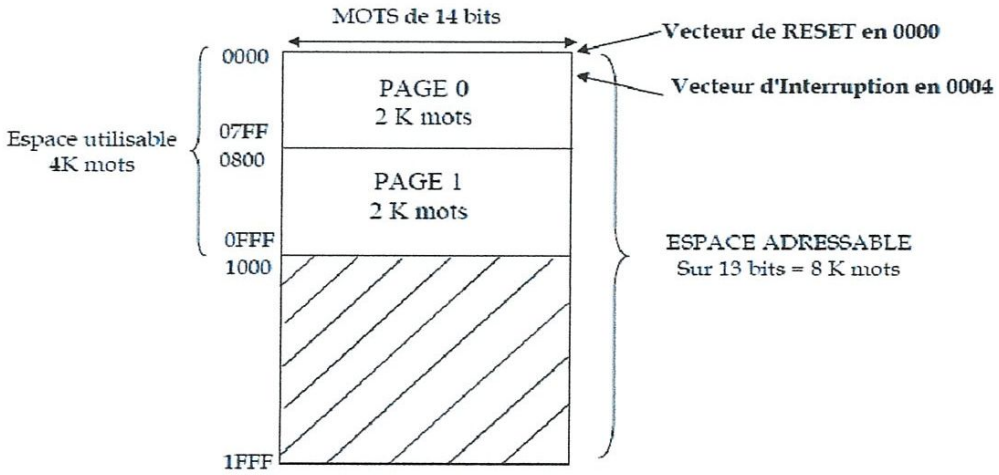


Figure 1.7 : Organisation de la mémoire programme

B. La mémoire EEPROM :

La mémoire EEPROM (**E**lectrical **E**rasable **P**rogrammable **R**ead **O**nly **M**emory), 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.

C. La mémoire Ram et organisation :

La 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 16F88 est de 368 octets.
Pour accéder à la RAM, on dispose de deux modes d'adressage :

C.1. Adressage DIRECT :

Le code opérande d'une instruction en mode d'adressage direct, contrairement au code immédiat, contient l'adresse d'une donnée en mémoire (au lieu de contenir la donnée).

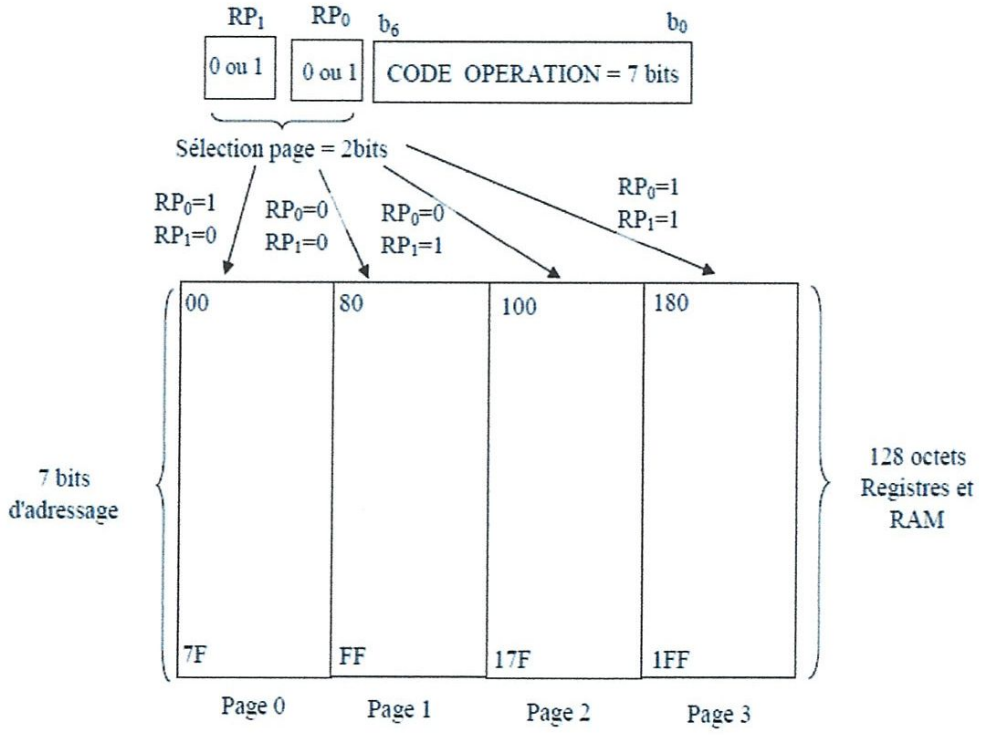


Figure 1.8 : Structure d'adressage direct

C.2. Adressage INDIRECT :

L'adresse de l'opérande est cette fois écrite avec un pointeur d'adresse.
L'adressage indirect utilise le registre du microcontrôleur : FSR.
Pour accéder à la donnée contenue dans la case mémoire X, il suffit d'écrire dans le registre FSR l'adresse X.

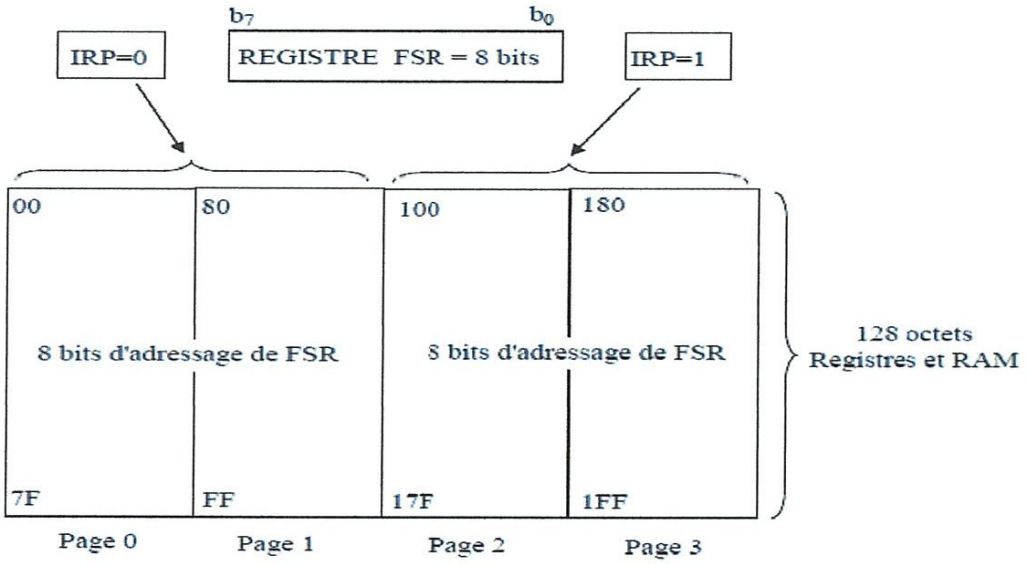


Figure 1.9 : Structure d'adressage indirect.

D. Watchdog :

Cette fonction est 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) le registre interne grâce à l'instruction clrwdt (clear watchdog). Si le programme tourne par exemple dans une boucle sans fin (qui est un bug dans le programme) le fonctionnement de watchdog va permettre de remettre à 0 le micro contrôleur afin de relancer le programme.

E. Les TIMERS :

Un timer permet de compter le temps, ou des impulsions. Le PIC 16F88 possède 3 timers, chaque un possède des particularités précises.

❖ Timer 0

C'est le timer de base et se configure via le registre OPTION_REG, il est sur 8 bits. Le timer 0 est un compteur qui peut être initialisé à une valeur donnée via le registre TMR0 (charger la valeur désirée dans ce registre). Lorsque l'interruption

du timer 0 est activée, le passage de 0xFF à 0x00 du timer provoque l'activation du flag du timer0.

❖ **Timer 1**

Second timer, il est sur 16 bits. Ce registre se configure via T1CON. Tout comme le timer 0, le timer 1 est peut être initialisé en chargeant la valeur désirée dans les 2 registres dédiés:
TMR1H pour l'octet supérieur et TMR1L pour l'octet inférieur.

❖ **Timer 2**

Troisième timer du PIC 16F88 et second timer 8 bits. Il possède également, en plus d'un pré-diviseur, un post-diviseur. Il peut être initialisé via le registre TMR2. La période du timer 2 peut être configurée via PR2 (1octet).

F. Les ports entrée/sortie :

On dispose de 16 broches d'entrées/sorties, chacune configurables soit en entrée soit en sortie (PORTA, PORTB).
Les E/S se configurent via le registre TRISA, TRISB.

F.1. Particularité du port A :

Ce port bidirectionnel est constitué de 8 bits. Le registre de direction correspondant est TRISA. Quand on écrit un "1" dans TRISA, le bit correspondant du PORTA est configuré en ENTREE, et le driver de sortie est placé en haute impédance.
Si on écrit un "0", le port devient une SORTIE, et le contenu du latch correspondant est chargé sur la broche sélectionnée.
Les bits de 0 à 4 du Port A peuvent servir d'E/S numériques ou bien être les 5 premières entrées du convertisseur analogique/numérique.
C'est par le registre ANSEL que sera fait le choix. Le bit 5 peut être soit une entrée digitale soit la commande MCLR qui est le reset. C'est par le mot de configuration que le choix de cette broche sera fait.
Les bits 6 et 7 peuvent être soit des E/S numériques soit les broches servant à relier le quartz de l'oscillateur. Quand on veut récupérer ces broches, il faudra faire fonctionner le PIC sur son oscillateur interne.

F.2. Particularités du port B :

Il comporte 8 bits. Le registre de direction correspondant est TRISB.
Si on écrit un "1" dans le registre TRISB, le bit correspondant du PORTB est configuré en ENTREE, et le driver de sortie correspondant passe en haute

impédance. Si on écrit un "0", le contenu du Latch de sortie correspondant est recopié sur la broche de sortie. Chaque broche du PORT B est munie d'un tirage au +VDD que l'on peut mettre ou non en service en mode entrée uniquement. On active cette fonction par la mise à "0" du bit 7 dans le registre OPTION.
Au reset, le tirage est désactivé, Il est inactif quand le port est configuré en sortie. Les 4 broches PB7 PB6 PB5 et PB4 provoquent une interruption sur un changement d'état si elles sont configurées en ENTREE. Cette possibilité d'interruption sur un changement d'état associé à la fonction de tirage configurable sur ces 4 broches, permet l'interfaçage facile avec un clavier. Cela rend possible le réveil du PIC en mode SLEEP par un appui sur une touche du clavier. On doit remettre à zéro le Flag de cette interruption (RBIF = bit 0 du registre INTCON) dans le programme d'interruption

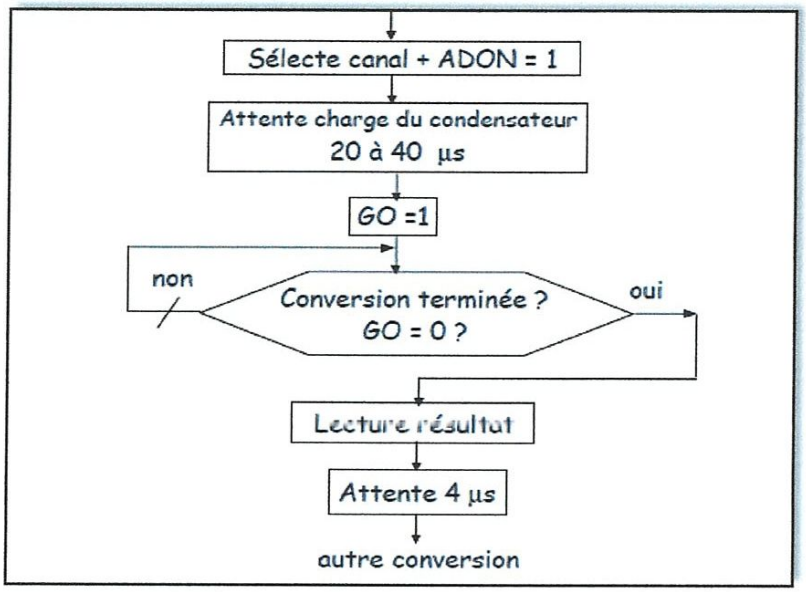
G. Le convertisseur :

Le CAN est un périphérique intégré dans le PIC 16F88.
La fonction conversion analogique-numérique consiste à transformer une grandeur électrique en une grandeur numérique exprimée sur N bits (en nombre binaire).
Notre 16F88 travaille avec un convertisseur analogique/numérique qui permet un échantillonnage sur 10 bits. Le signal numérique peut donc prendre 1024 valeurs possibles.
Les 5 premières entrées sont sur le Port A en RA0, RA1, RA2, RA3 et RA4, et le 2 dernières sur le Port B en RB6 et RB7.
Les tensions de référence haute et basse peuvent être choisies par programmation parmi: VDD ou la broche PA3 pour VREF + et VSS ou la broche PA2 pour VREF- .

Le convertisseur du PIC 16F88 convient pour l'acquisition de signaux lents (par exemple, suivi de la température ambiante d'une salle à partir d'un capteur LM335).

Les 4 registres utilisés par le module convertisseur A/D sont :
- ADRESH: MSB des 10 bits du résultat.
- ADRESL: LSB des 10 bits du résultat.
- ADCON0: registre de contrôle n°0 du convertisseur.
- ADCON1: registre de contrôle n°1 du convertisseur.

Exemple d'un programme :



H. 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 stabilité du montage est limitée.

Notre PIC peut fonctionner sous les mêmes modes classiques que les 16F84 ou 16F877, c'est à dire:

- Mode LP : Avec des quartzs de fréquence basse (200 KHz max).
- Mode XT: Avec des quartzs de fréquence max 4 MHz.
- Mode HS: Avec des quartzs de fréquence max 20 MHz.
- Mode RC: Avec un condensateur et une résistance extérieure.

Il possède de nouveaux modes:

- Mode ECIO: Avec une horloge externe entrant sur PA6.
- Mode INTIO1 : Oscillateur interne. Fosc/4 sur PA6 et PA7 libre pour I/O.
- Mode INTIO2 : Oscillateur interne. PA6 et PA7 libres pour des I/O.

Le choix du mode sera fait par les 3 bits Fosc0 Fosc1 et Fosc2 du mot de configuration.

H.1. Le bloc oscillateur interne

Le 16F88 possède un bloc oscillateur qui produit un signal de 31,25 KHz et un signal de 8 MHz. Le premier sert à piloter le "watchdog". Le signal à 8 MHz qui passe à travers un pré-diviseur, peut servir d'horloge système. L'oscillateur étant divisé par 4, on aura une horloge cycle maximum de 2 MHz soit un temps de cycle de 500 ns. Le choix du pré-diviseur pour le signal de 8 MHz est fait par les 3 bits IRCF0 IRCF1 et IRCF2 du registre OSCCON.

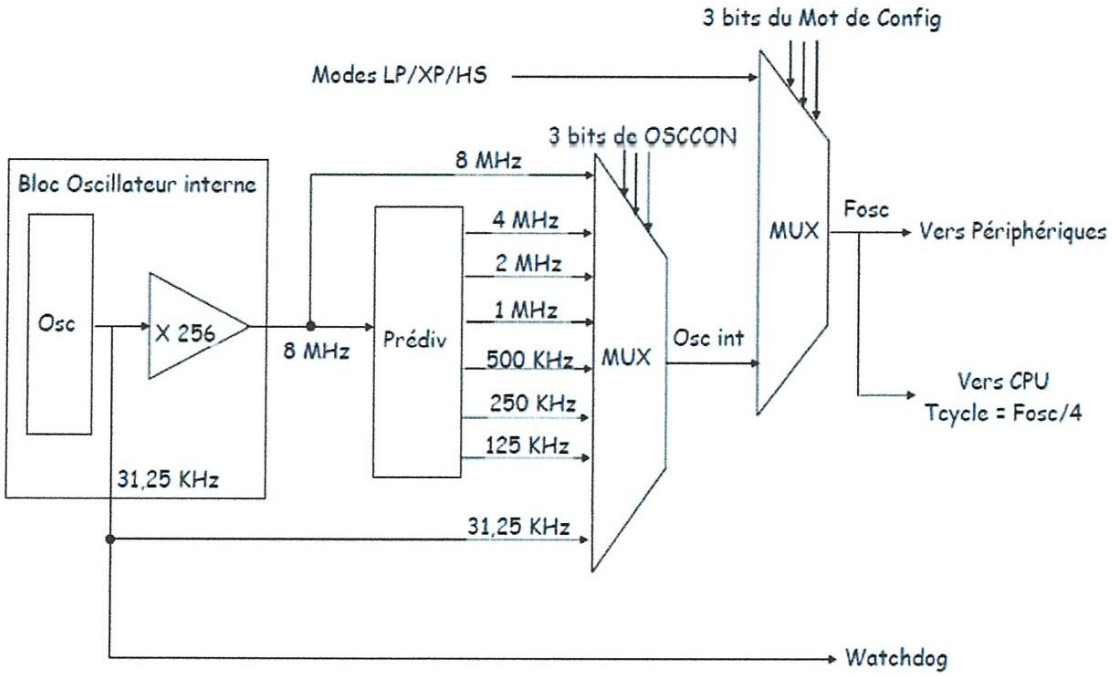


Figure 1.10 : structure de l'oscillateur interne

III.2. Le module MSSP et le bus I2C

III.2.1 Le module MSSP :

Le MSSP est un des deux modules de communication série du PIC 16F88. Il permet d'échanger des données en mode synchrone avec d'autres circuits qui peuvent être des microcontrôleurs, des mémoires EEPROM série, des convertisseurs A/N, des modules d'affichage…

Il peut fonctionner selon deux modes : le mode **SPI** (Serial Peripheral Interface) et le mode **I²C** (Inter-Integrated Circuit).

III.2.2. Présentation du bus I2C :

Le bus I²C (Inter-Integrated Circuit) est un bus populaire développé par la société

Philips dans les années 1980.

Le bus I2C est un **bus série synchrone** bifilaire :

- **SDA** (Serial Data Line) : ligne de données bidirectionnelle
- **SCL** (Serial Clock Line) : horloge de synchronisation bidirectionnelle

N.B : Cela fait 3 fils en comptant la masse.

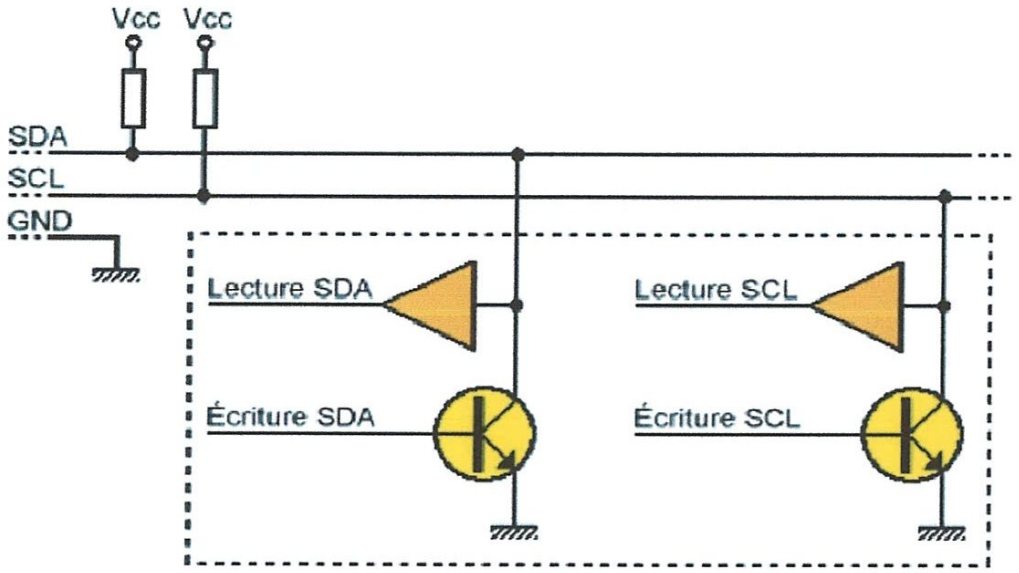


Figure 1.11 : support physique de bus I²C

Le bus I²C est contrôlé par un maître (c'est généralement un microcontrôleur, par exemple un PIC 16F88). Le bus I²C peut avoir plusieurs maîtres (on parle alors de mode multi-maîtres), mais à un instant donné, un seul maître contrôle le bus I²C. Le bus I²C possède un ou plusieurs esclaves (un bus I2C sans esclaves ne sert à rien). **Un esclave est identifié par une adresse unique** (sur 7 bits ou parfois 10 bits).

III.2.3. Le module MSSP en mode I2C :

Le module MSSP du PIC peut être configuré en master ou en esclave. Il utilise les broches RC3/SCL (horloge) et RC4/SDA (données). Ses broches doivent être configurées en entrée. L'accès au module en lecture et écriture se fait à l'aide du registre tampon SSPSR auquel nous n'avons pas directement l'accès.

Le contrôle du module se fait à l'aide des registres :

- SSPCON : registre de control (BANK0)
- SSPCON2 : registre de control (BANK1)
- SSPSTAT : registre d'état (BANK1)
- SSPADD : registre d'adresse (BANK1)

III.2.4. Le module SSP du PIC 16F88

Le module SSP (Synchronous serial port) du PIC 16F88 gère le bus I2C, mais seulement en mode esclave. C'est la principale différence avec le module MSSP (Master Synchronous serial port) du PIC 16F87X, qui gère également (au sens hardware) le mode maître et le mode multi-maîtres. Cependant, le module SSP du PIC 16F88 permet d'implanter de manière logicielle le mode I2C maître ou multi-maîtres.

IV. CONCLUSION

Au niveau du ce chapitre, nous avons présenté les microcontrôleurs en général et le PIC 16F88 particulièrement qui est le PIC avec lequel on travail dans ce projet ; nous avons cité quelques caractéristiques importantes ainsi que leur module de communication entre la carte d'acquisition réalisée (qui sera présentée plus tard) et l'ordinateur.

I. INTRODUCTION

Nous présenterons dans ce chapitre quelques principes généraux qui permettraient de mettre en œuvre des phénomènes intervenant dans l'acquisition de données, les filtres analogiques, les filtres numériques, la conversion analogique numérique et la liaison RS232

II. SYSTEME D'ACQUISITION DE DONNEES :

II.1. Définition :

Le système d'acquisition des données (S.A.D) est un ensemble d'éléments matériel et logiciel destiné à recueillir des données physiques par l'intermédiaire de capteurs.

II.1.1. Une Chaine d'acquisition comporte :

- Une source d'information (Un patient par exemple).
- Les capteurs capables de transformer l'information physiologique en une grandeur électrique.
- Les circuits de mise en forme réalisant les fonctions d'amplification, de filtrage, de calibration etc…
- La carte d'acquisition des données proprement dite permettant l'interfaçage homme machine.
- Un support logiciel réalisant l'acquisition, l'affichage, le traitement et la transmission des données conformément a un protocole de communication.

II.1.2. Schéma bloc d'une chaine d'acquisition :

Une chaine d'acquisition peut se représenter selon le Schéma bloc suivant :

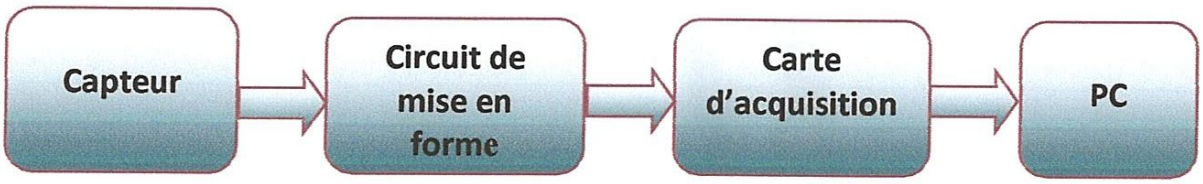


Figure 2.1 : Schéma bloc d'une chaine d'acquisition

II.1.3. Carte d'acquisition de donnée :

C'est un ensemble de circuits électroniques ou interface d'entrée/sortie, destiné à traiter l'information venant des capteurs: échantillonnage, digitalisation, etc.

II.1.4. Schéma bloc d'une carte d'acquisition :

Une carte d'acquisition numérique peut se représenter selon la figure suivante :

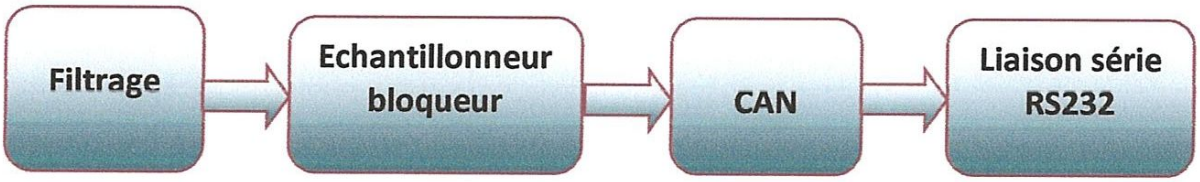


Figure 2.2: Structure d'une carte d'acquisition

III. LA CONVERSION ANALOGIQUE NUMERIQUE :

III.1. Identification de la fonction C.A.N :

On appelle Convertisseur Analogique Numérique (**C.A.N.**) (ou Analogic Digital Conversion **A.D.C.** en anglais) tout dispositif électronique qui transforme une grandeur analogique d'entrée $\mathbf{u_e}$ en un nombre binaire de sortie N proportionnel à cette grandeur u_e.

III.2. Symbole d'un C.A.N

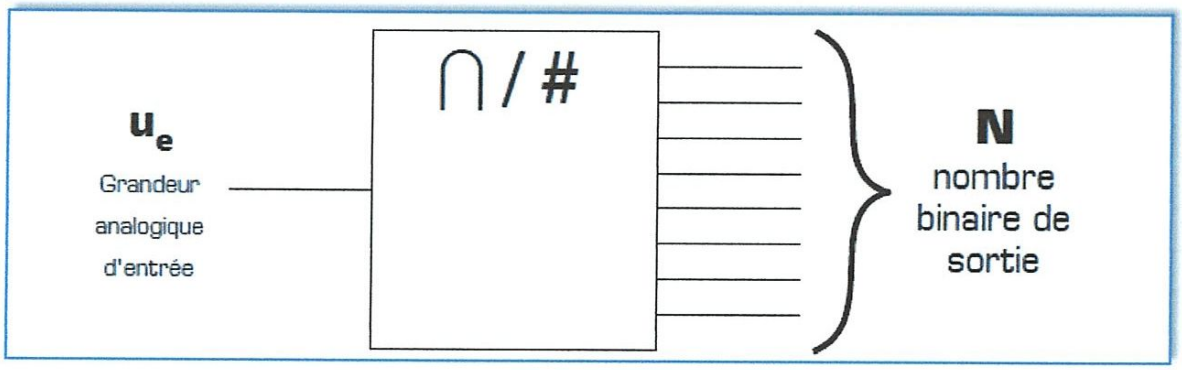


Figure 2.3 : Symbole de la fonction Conversion Analogique / Numérique

Le signe ∩ indique que la grandeur est de type **analogique** (il s'agit ici de l'entrée).
Le signe # indique que la grandeur est de type **numérique** (il s'agit ici de la sortie).

III.3. Principe de fonctionnement :

L'obtention d'un signal numérique à partir d'un signal analogique (exemple : température) nécessite de mettre en œuvre une chaîne de conversion analogique-numérique.

Le convertisseur analogique-numérique a pour fonction de faire correspondre à un signal analogique un signal numérique de sortie.

Cette chaîne comporte typiquement la structure fonctionnelle suivante :

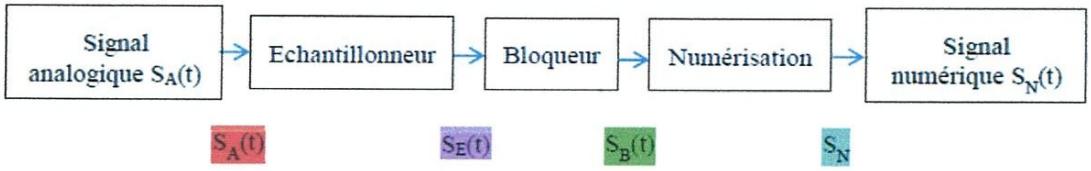


Figure 2.4 : Principe du chaîne de conversion analogique-numérique

Les fonctions réalisées par chaque élément sont les suivantes :

- Echantillonneur : Acquérir la grandeur analogique à un instant t ;
- Bloqueur : Maintenir cette grandeur pendant la durée de la numérisation ;
- Numérisation : Convertir le signal analogique bloqué en un nombre binaire (sur N bits).

III.3.1. Echantillonnage :

L'échantillonneur a pour rôle de prélever périodiquement la valeur du signal analogique. Il réalise une « photographie » du signal qui « fige » son image jusqu'à la « prise » suivante.

L'intervalle de temps entre 2 instants consécutifs d'échantillonnage constitue la période d'échantillonnage, notée Te, du système de conversion analogique numérique.

Le choix de cette période est imposée à la fois par la nature du signal que l'on veut numériser mais aussi par le temps que met le convertisseur analogique numérique pour réaliser sa conversion.

Usuellement, on utilise la notion de fréquence d'échantillonnage telle que :

$$F_E = \frac{1}{Te}$$

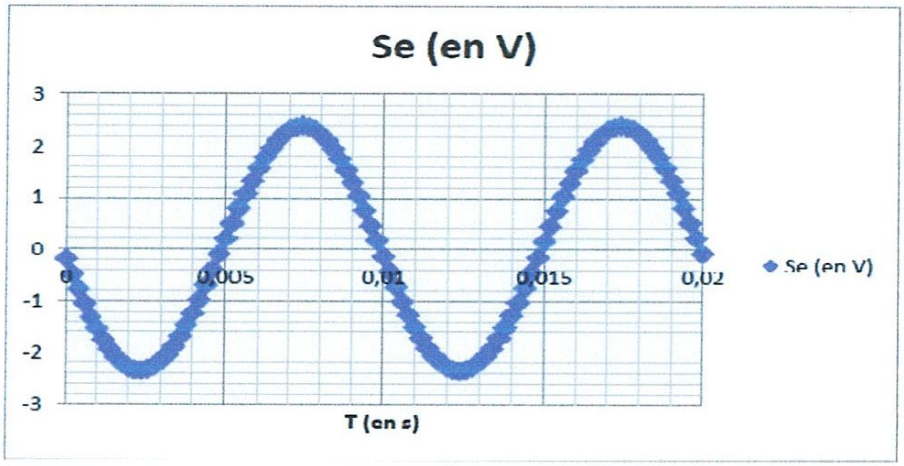


Figure 2.5 : exemple de signal échantillonné

La figure ci-dessus représente l'allure du signal échantillonné avec une période d'échantillonnage de 200 µs. A chaque instant (multiple de 200 µs), une valeur du signal analogique est acquise (comme une photo).
D'une manière générale, un signal véhiculant une information contient un certain nombre de fréquences correspondant à son spectre. Par nature, pour chaque fréquence, l'amplitude est aléatoire. Il est donc nécessaire de considérer une bande de fréquence dans laquelle se trouvent de façon pertinente les informations relatives au signal. On fixe ainsi une fréquence maximale du signal analogique et par conséquent on accepte de « perdre » une partie de l'information.
Ce choix doit être pertinent en trouvant un compromis entre rapidité de conversion et qualité de numérisation.

III.3.2. Bloqueur :

Le bloqueur a pour fonction de maintenir constant le signal échantillonné SE(t) afin de permettre au convertisseur analogique numérique de le numériser. Ce signal est maintenu constant à la valeur SB(t) jusqu'à l'échantillonnage de la valeur suivante.
Actuellement les bloqueurs agissent comme une mémoire qui garde constante la valeur échantillonnée (SB(t)=constante), on parle alors de bloqueur d'ordre 0.

III.3.3. Numérisation :

Principe :

La numérisation est la transformation d'un signal analogique issu du bloqueur en un signal numérique. Elle est réalisée par un convertisseur analogie

numérique qui convertit le signal bloqué en un nombre. La sortie du convertisseur est raccordée ensuite à un système numérique (ordinateur, analyseur de spectre numérique…)
Cette numérisation réalise la quantification du signal analogique bloqué en lui associant un nombre. Ce nombre est compris entre **0** et $\mathbf{2^N-1}$ valeurs possibles où **N** est le nombre de bits en sortie du convertisseur. Ainsi pour 8 bits, on a $\mathbf{2^8}$ valeurs différentes possibles en sortie soit 256 valeurs différentes. Les nombres associés vont alors de la valeur 0 jusqu'à 255.

IV. LE FILTRAGE ANALOGIQUE :

D'une manière générale, un filtre est un système linéaire dont le rôle est de modifier la composition spectrale d'u signal sans y ajouter de nouvelles composantes. Il permet le renforcement ou l'atténuation d'une ou plusieurs bandes de fréquences.
Un filtre est un circuit dont le comportement dépend de la fréquence. C'est un circuit linéaire.
Si la tension d'entrée est sinusoïdale alors la sortie est sinusoïdale de même fréquence.

V. LA GESTION DE LA LIAISON RS232 :

V.1. communication série asynchrone à travers le port série RS232 :

Le protocole RS232 est apparu en 1962, et bien qu'il est vieux, il est toujours beaucoup utilisé en industrie pour la communication entre un ordinateur via un port série et un système électronique. Les communications sont en Full duplex, c'est-à-dire capable de fonctionner dans les deux sens, ainsi la carte d'acquisition peut envoyer des informations et en recevoir.
Les liaisons séries permettent la communication entre deux systèmes numériques en limitant le nombre de fils de transmission. La liaison série aux normes de RS232 est utilisée dans tous les domaines de l'informatique. Elle est de type asynchrone, c'est-à-dire qu'elle ne transmet pas le signal d'horloge.
Le schéma fonctionnel est le suivant :

Figure 2.6 : schéma fonctionnel d'une liaison asynchrone de la norme RS232

La transmission série nécessite au moins 2 fils de communication, l'un pour la transmission (Tx) et l'autre pour la réception (Rx) et un fil de masse.

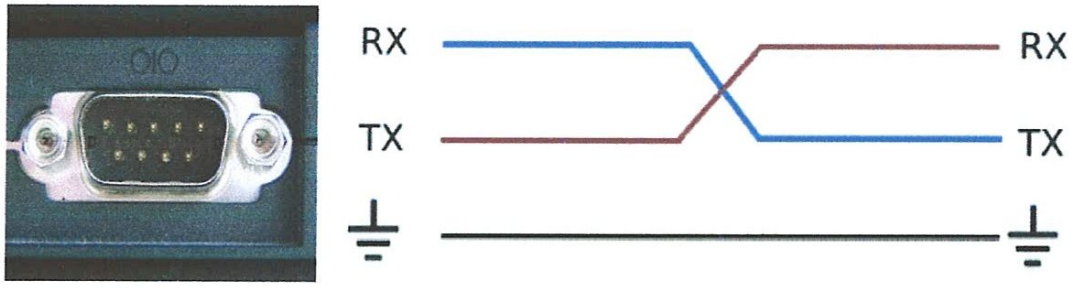


Figure 2.7 : la liaison RS232 croisé

V.2. protocole de transmission :

Afin que les éléments communicants puissent se comprendre, il est nécessaire d'établir un protocole de transmission. Ce protocole devra être le même pour les deux éléments afin que la transmission fonctionne correctement.
Paramètre rentrant en jeu :

- Longueur des mots : 7 ou 8 bits.
- La vitesse de transmission : les différentes vitesses de transmission son réglables a partir de 110 bauds (bits par seconde) de la façon suivante : 110 bds, 150 bds, 300 bds, 600 bds, 1200 bds, 2400 bds, 4800 bds, 9600 bds.
- Parité : le mot transmis peut être suivi ou non d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission.
- Bit de start : la ligne au repos est à l'état logique 1 pour indiquer qu'un mot va être transmis, la ligne passe a l'état bas avant de commencer le transfert. Ce bit permet de synchroniser l'horloge du récepteur.
- Bit de stop : après la transmission, la ligne est positionnée au repos pendant 1, 2 ou 1,5 période d'horloge selon le nombre de bits de stop.

Le bit de start apparait en premier dans la trame puis les données (poids faible en premier), la parité éventuelle et le (les) bit(s) de stop.

VII. CONCLUSION

Ce chapitre a été essentiellement consacré à la description théorique du système d'acquisition de données. Ces aspects théoriques sons les outils nécessaires pour une compréhension correcte de la carte d'acquisition pour les signaux lents utilisée dans notre projet et que nous allons détailler dans le chapitre suivant.

Chapitre 3

I. INTRODUCTION :

Dans notre projet on va essayer de réaliser une carte d'acquisition à 7 entrées pour signaux lents, qui peut être utilisée pour l'acquisition de température, de pressions atmosphériques, etc…

- La fréquence d'acquisition par voie est de 1Hz maximum (c'est-à-dire 60 acquisition maximum par minute).
- Le Convertisseur Analogique/Numérique (module interne du PIC 16F88) qui à les caractéristiques suivantes :
  - Plage de tension analogique : 0,000V à 5,000V.
  - Résistance interne de la source de tension à échantillonnée : 5 K maximum.
  - Résolution numérique : 10 bits (5mv).

N.B. la carte se connecte à un ordinateur (et cela se fait via une liaison RS232).

II. SCHEMA ELECTRIQUE DE LA CARTE D'ACQUISITION :

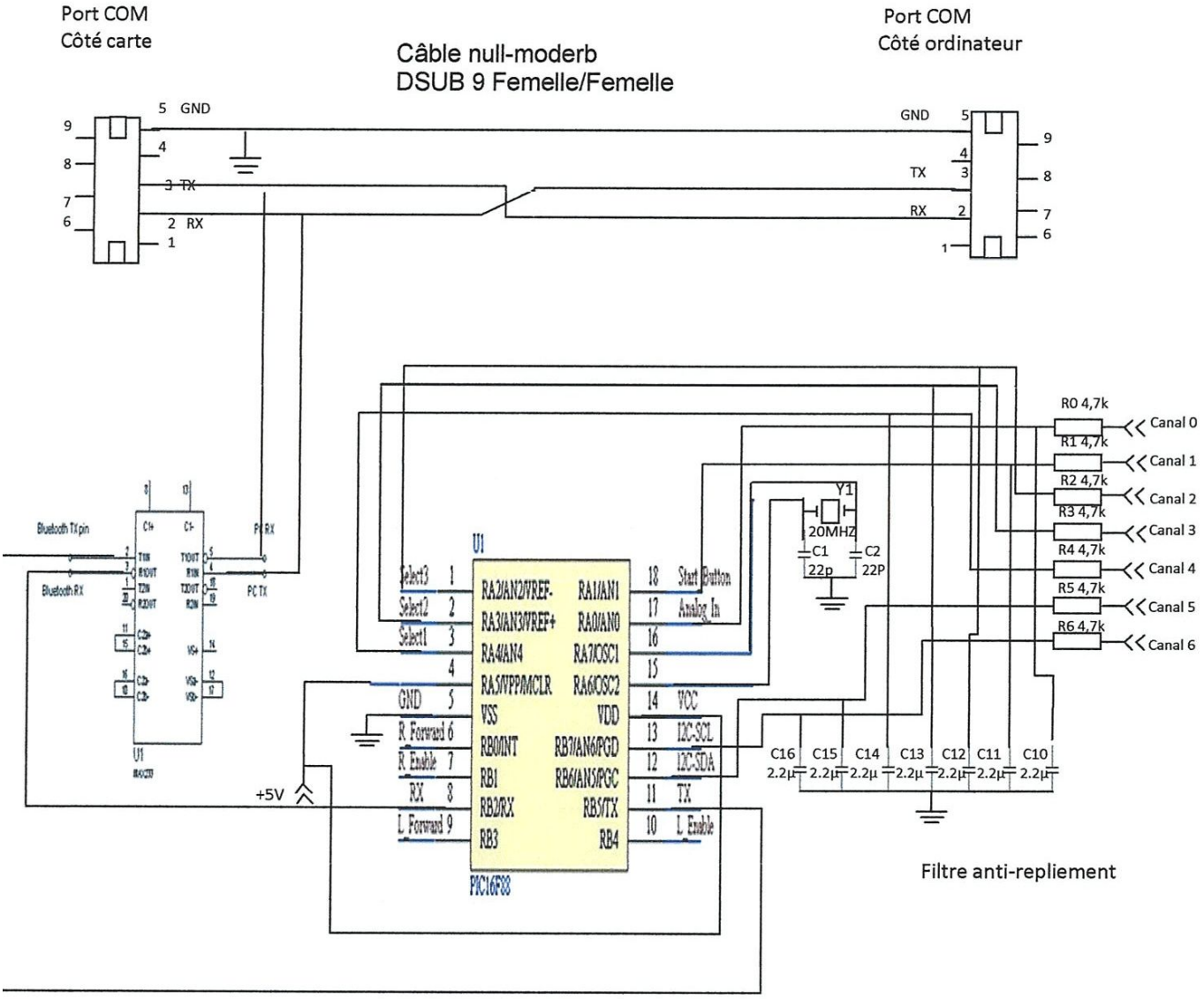


Figure 3.1 : schéma électrique de la carte

III. PRINCIPE DE FONCTIONNEMENT :

III.1. Les filtres anti-repliements :

Un filtre anti-repliement est intercalé entre la tension à mesurer (qui provient généralement d'un capteur) et le canal d'entrée de l'ADC du PIC.

Il s'agit ici d'un simple filtre analogique passe-bas

Fréquence de coupure à 3 dB :

R=4,7 KΩ

C=2,2 µf

$$f_c = \frac{1}{2\pi RC} \approx 15\ \text{IIz}$$

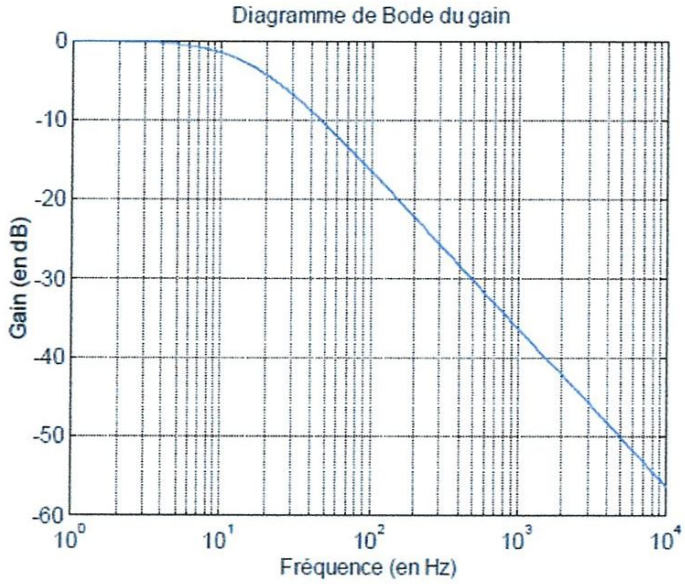


Figure 3.2 : diagramme de BODE de fréquence de coupure

III.2. Le convertisseur analogique - numérique (ADC) :

ADC = **A**nalog to **D**igital **C**onverter = **C**onvertisseur **A**nalogique - **N**umérique (CAN en français).

Le rôle du convertisseur analogique - numérique est de transformer une tension analogique en un nombre binaire (proportionnel à la tension analogique).

Le microcontrôleur PIC 16F88 a l'avantage de posséder un module ADC interne, contrairement au PIC 16F84A ou 16F628A (il faut alors adjoindre un ADC externe, ce qui complique le schéma électrique et la programmation du PIC).

III.2.1. Caractéristiques principales du module ADC du PIC 16F88 :

• Plage de tension analogique : 0,000 V à 5,000 V.
• Résistance interne de la source de tension à échantillonner : 5000 ohms maximum.
• Résolution numérique : 10 bits (5 mV).
• type : à approximations successives.
• nombre de canaux : 7 (multiplexés).
• période d'échantillonnage : environ 42 µs.

→ Principaux registres relatifs au module ADC :
• **ANSEL** (adresse 0x9B, banque 1)
• **ADCON0** (adresse 0x1F, banque 0)
• **ADCON1** (adresse 0x9F, banque 1)
• **ADRESH** (adresse 0x1E, banque 0)
• **ADRESL** (adresse 0x9E, banque 1)

✓ **Etape 1 : Configuration :**

A. Choix des canaux d'entrées :

On peut utiliser jusqu'à 7 canaux d'entrées (mais à un instant donné, un seul canal est relié à l'ADC).

Numéro de canal	Broche
0	RA0/AN0
1	RA1/AN1
2	RA2/AN2/VREF-
3	RA3/AN3/VREF+
4	RA4/AN4
5	RB6/AN5
6	RB7/AN6

Tableau 3.1 : choix des canaux

Supposons que l'on ait besoin de 4 des 7 canaux (canal 0, 1, 2 et 5). Il faut configurer les 4 broches correspondantes comme **entrée analogique** :
On commence par configurer ces 4 broches en **entrée** avec les registres spéciaux **TRISA** et **TRISB** (banque 1), puis on configure les entrées comme entrée de type analogique avec le registre spécial **ANCEL** (banque 1).

bsf STATUS, RP0 ; passage en banque 1
bsf TRISA, 0 ; configuration de la broche RA0 en entrée
bsf TRISA, 1 ; configuration de la broche RA1 en entrée
bsf TRISA, 2 ; configuration de la broche RA2 en entrée
bsf TRISB, 6 ; configuration de la broche RB6 en entrée
movlw B'01000111' ; W = B'01000111'
movwf NSEL ; (ANSEL) = B'01000111'

• **bit 7** du registre ANSEL = X : non implémenté
• **bit 6** du registre ANSEL = 0 : configuration de la broche **RB7**/AN6 comme **entrée/sortie numérique**
• **bit 5** du registre ANSEL = 1 : configuration de la broche RB6/**AN5** comme **entrée analogique** (canal 5)
• **bit 4** du registre ANSEL = 0 : configuration de la broche **RA4**/AN4 comme **entrée/sortie numérique.**
• **bit 3** du registre **ANSEL** = 0 : configuration de la broche **RA3**/AN3/VREF+ comme **entrée/sortie numérique**
• **bit 2** du registre ANSEL = 1 : configuration de la broche RA2/**AN2**/VREF- comme **entrée analogique** (canal 2)
• **bit 1** du registre ANSEL = 1 : configuration de la broche RA1/**AN1** comme **entrée analogique** (canal 1)
• **bit 0** du registre ANSEL = 1 : configuration de la broche RA0/**AN0** comme **entrée analogique** (canal 0)

B. Choix des tensions de référence :

Quatre combinaisons sont possibles.

• Elles dépendent de 2 bits du registre spécial **ADCON1** (banque 1) :

VCFG1 (bit 5 du registre ADCON1)	VCFG0 (bit 4 du registre ADCON1)	Tension analogique d'entrée (référence haute)	Tension analogique d'entrée (référence basse)	Remarques
0	0	VDD	VSS	
0	1	VDD	RA2/AN2/VREF-	La broche RA2 doit être configurée comme entrée analogique : **bsf TRISA, 2** **bsf ANSEL, 2** Le canal 2 n'est plus disponible
1	0	RA3/AN3/VREF+	VSS	La broche RA3 doit être configurée comme entrée analogique : **bsf TRISA, 3** **bsf ANSEL, 3** Le canal 3 n'est plus disponible
1	1	RA3/AN3/VREF+	RA2/AN2/VREF-	Les broches RA2 et RA3 doivent être configurées comme entrée analogique : **bsf TRISA, 2** **bsf TRISA, 3** **bsf ANSEL, 2** **bsf ANSEL, 3** Les canaux 2 et 3 ne sont plus disponibles

Tableau 3.2 : choix des tensions de référence

VDD = broche d'alimentation positive du microcontrôleur PIC 16F88

VSS = broche de masse (0 V) du microcontrôleur PIC 16F88

Exemple:

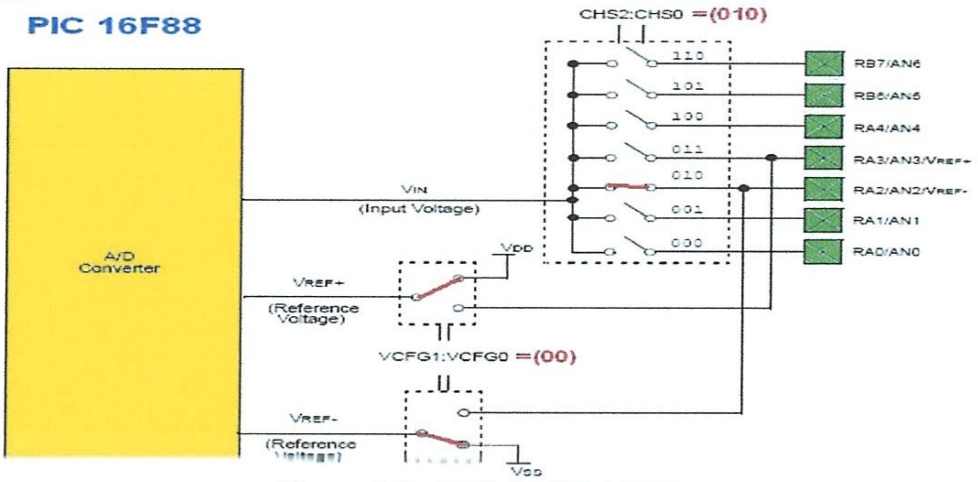


Figure 3.3 : ADC du PIC 16F88

Le microcontrôleur PIC 16F88 est alimenté avec une tension stabilisée de +5,00 V (VDD).

bsf STATUS, RP0 ; passage en banque 1

bcf ADCON1, VCFG0 ; VCFG0 = 0

bcf ADCON1, VCFG1 ; VCFG1 = 0

Avec VCFG1 = 0 et VCFG0 = 0, la plage d'entrée du convertisseur ADC est [VSS à VDD] = [0 V à 5,00 V].

Le convertisseur a une résolution de 10 bits, soit 2^10 = 1024 pas de 4,88 mV (5V /1024).

C. Choix du format du résultat de la conversion :

Le convertisseur ADC fournit un nombre binaire naturel de 10 bits (b9 b8 b7 b6 b5 b4 b3 b2 b1b0).

b0 = LSB (bit de poids faible)

b9 = MSB (bit de poids fort)

Deux registres (2 x 8 bits) sont donc nécessaires pour stocker le résultat de la conversion (10 bits). Ce sont les registres :

- **ADRESH** (en banque 0)
- **ADRESL** (en banque 1)

Deux formats sont disponibles suivant la valeur du bit ADFM (bit 7 du registre spécial **ADCON1**, situé en banque 1) :

ADFM (bit 7 du registre spécial ADCON1)	Instruction	Registre ADRESH	Registre ADRESL	Exemple 0
0	**bcf ADCON1, ADFM**	(b9 b8 b7 b6 b5 b4 b3 b2)	(b1 b0 0 0 0 0 0 0)	D'758' = 0x2F6 (ADRESH) = 0xBD (ADRESL) = 0x80
1	**bsf ADCON1, ADFM**	(0 0 0 0 0 0 b9 b8)	(b7 b6 b5 b4 b3 b2 b1 b0)	D'758' = 0x2F6 (ADRESH) = 0x02 (ADRESL) = 0xF6

Choix du format du résultat de la conversion

ADFM = A/D Result Format select bit.

D. Choix de la fréquence d'horloge du convertisseur ADC :

Le choix de la fréquence d'horloge du convertisseur ADC se fait avec 2 bits du registre spécial **ADCON0** (banque 0) et 1 bit du registre spécial **ADCON1** (banque 1) :

ADCS2 (bit 6 du registre ADCON1)	ADCS1 (bit 7 du registre ADCON0)	ADCS0 (bit 6 du registre ADCON0)	F AD Fréquence d'horloge du convertisseur ADC	T AD Période d'horloge du convertisseur ADC
0	0	0	FOSC / 2	2 x T OSC
0	0	1	FOSC / 8	8 x T OSC
0	1	0	FOSC / 32	32 x T OSC
0	1	1	FRC	TRC
1	0	0	FOSC / 4	4 x T OSC
1	0	1	FOSC / 16	16 x T OSC
1	1	0	FOSC / 64	64 x T OSC
1	1	1	FRC	TRC

Tableau 3.3 : Choix de fréquence d'horloge d'ADC

• **F OSC** est la fréquence de l'horloge du microcontrôleur PIC 16F88.

Par exemple, avec un oscillateur à quartz externe de 20 MHz et (ADCS2 ADCS1 ADCS0) = (010) :

F OSC = 20 MHz (T OSC = 50 ns)

F AD = F OSC / 32 = 625 kHz

T AD = 1,6 µs.

• **F RC** est la fréquence de l'oscillateur RC interne du microcontrôleur PIC 16F88.

• Pour assurer un fonctionnement correct du convertisseur ADC, Microchip indique que la période **T AD doit être comprise entre 1,6 µs et 6,4 µs.**

Pour une conversion plus rapide, on prendra (si possible) : **TAD = 1,6 µs**

Cela impose une contrainte sur la fréquence de l'horloge du microcontrôleur PIC 16F88 :

ADCS2 (bit 6 du registre ADCON1)	ADCS1 (bit 7 du registre ADCON0)	ADCS0 (bit 6 du registre ADCON0)	F OSC Maximum
0	0	0	1,25 MHz (T AD = 1,6 µs)
0	0	1	5 MHz (T AD = 1,6 µs)
0	1	0	20 MHz (T AD = 1,6 µs)
1	0	0	2,5 MHz (T AD = 1,6 µs)
1	0	1	10 MHz (T AD = 1,6 µs)
1	1	0	20 MHz (T AD = 3,2 µs)

Tableau 3.4 : Fréquence maximum d'horloge du PIC 16F88

E. Mise en service de l'interruption du convertisseur ADC :

Uniquement dans le cas où vous voulez utiliser la technique des interruptions :

bcf STATUS, RP0 : passage en banque 0

bcf PIR1, ADIF : on efface le drapeau de l'interruption du convertisseur ADC

bsf STATUS, RP0 : passage en banque 1

bsf PIE1, ADIE : autorisation de l'interruption du convertisseur ADC

bsf PEIE, INTCON : autorisation des interruptions des périphériques (dont le module ADC)

bsf GIE, INTCON : autorisation globale des interruptions

- **Le drapeau ADIF est automatiquement mis à 1 à la fin de la phase de conversion.**

→ Une interruption est alors déclenchée.

Dans la routine d'interruption, on pourra lire et traiter le résultat de la conversion (registres **ADRESH** et **ADRESL**). On n'oubliera pas d'effacer le drapeau ADIF.

✓ **Etape 2- Mise en service du convertisseur ADC :**

Pour cela, il faut mettre à 1 le bit ADON du registre **ADCON0** :

bcf STATUS, RP0 ; passage en banque 0

bsf ADCON0, ADON ; ADON = 1

Remarque : il faut mettre le bit ADON à 0 pour rendre hors service le convertisseur ADC :

bcf ADCON0, ADON ; ADON = 0

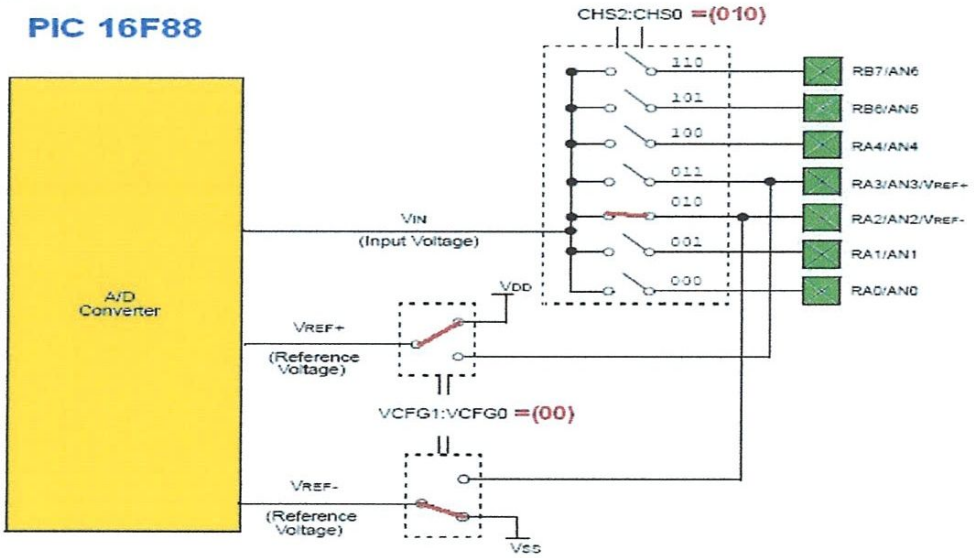
✓ **Etape 3- Sélection du canal à échantillonner :**

La sélection du canal à échantillonner se fait avec 3 bits du registre spécial **ADCON0** (banque0)

CHS2 (bit 5 du registre ADCON0)	CHS1 (bit 4 du registre ADCON0)	CHS0 (bit 3 du registre ADCON0)	Canal sélectionné
0	0	0	canal 0 (broche RA0)
0	0	1	canal 1 (broche RA1)
0	1	0	canal 2 (broche RA2)
0	1	1	canal 3 (broche RA3)
1	0	0	canal 4 (broche RA4)
1	0	1	canal 5 (broche RA6)
1	1	0	canal 6 (broche RA7)

Tableau 3.5 : Sélection du canal à échantillonner

Exemple :



Exemple sur la sélection du canal à échantillonner

Pour échantillonner le canal 2 :

bcf STATUS, RP0 ; passage en banque 0

bcf ADCON0, CHS2 ; CHS2 = 0

bsf ADCON0, CHS1 ; CHS1 = 1

bcf ADCON0, CHS0 ; CHS0 = 0

Remarque : Pour échantillonner plusieurs canaux (canal 0, 1, 2 et 5), on utilise la technique du multiplexage : Echantillonnage du canal 0, puis échantillonnage du canal 1, puis échantillonnage du canal 2, puis échantillonnage du canal 5, puis échantillonnage du canal 0, etc…

✓ **Etape 4- Attente pendant la phase d'acquisition :**

Pendant la phase d'acquisition vous devez programmer une temporisation de 19,7 µs (environ).

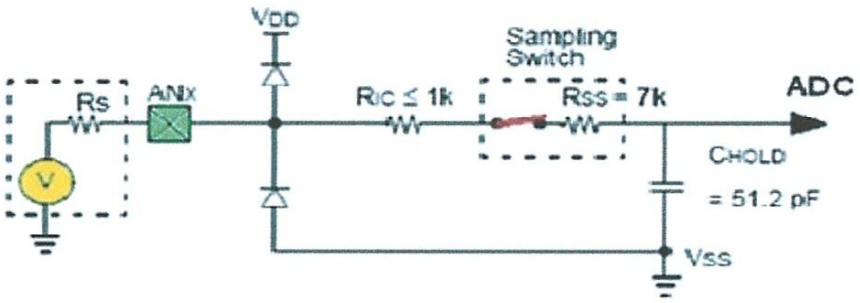


Figure 3.5 : Echantillonneur-bloqueur de l'ADC

En effet, la phase d'acquisition consiste à charger le condensateur de maintien C **HOLD** avec la tension présente à l'entrée (V). Mais la durée de charge du condensateur augmente avec la résistance interne RS de la source de tension.

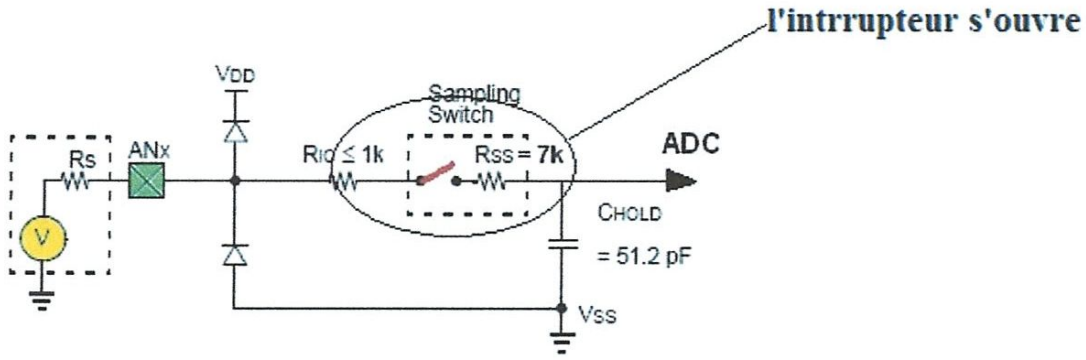
✓ **Etape 5- Lancement de la phase de conversion :**

Une fois la phase d'acquisition terminée, vous pouvez lancer la phase de conversion en mettant à 1 le bit GO du registre **ADCON0** :

bcf STATUS, RP0 ; passage en banque 0

bsf ADCON0, GO ; GO = 1

La conversion dure 12 x T AD soit **19,2 µs** dans le meilleur des cas **(T AD = 1,6 µs).**



Lancement de la phase de la conversion

Pendant la phase de conversion, l'interrupteur électronique est ouvert.
Le condensateur de maintien C HOLD conserve alors à ses bornes la tension V.
Cette tension est constante (aux fuites près) pendant toute la phase de conversion.

✓ **Etape 6- Attente de la fin de la phase de conversion :**

A la fin de la conversion, **le drapeau ADIF de l'interruption du convertisseur ADC est automatiquement mis à 1.** Si vous utilisez cette interruption, le programme saute immédiatement vers la routine d'interruption.
Si vous n'utilisez pas la technique des interruptions, il faut tester le bit GO du registre **ADCON0**.
En effet, **le bit GO est automatiquement remis à 0 en fin de conversion**.

✓ **Etape 7 - Lecture du résultat de la conversion :**

On peut maintenant lire et traiter le résultat de la conversion (registres **ADRESH** et **ADRESL**)

✓ **Etape 8 - Attente avant une nouvelle acquisition :**

Microchip indique qu'il faut attendre 2 x T AD (3,2 µs) entre la fin de la phase de conversion et un nouveau cycle d'acquisition.
En interne, après 2 x T AD, l'interrupteur électronique de l'échantillonneur-bloqueur est fermé, et le processus d'acquisition commence.

✓ **Etape 9 - Nouvelle acquisition**

On saute à **l'étape 3** (pour un changement de canal). Autrement, on recommence à **l'étape 4.**

III.3. Remarques et conseils :

A. Calcul de la période d'échantillonnage (TE) :

19,7 µs (phase d'acquisition)
+ 12 x T AD = 19,2 µs (phase de conversion)
+ 2 x T AD = 3,2 µs (temps mort)
TE = 42,1 µs (avec T AD = 1,6 µs)

B. Echantillonnage de signaux lents :

Le convertisseur du PIC 16F88 convient pour l'acquisition de signaux lents (par exemple, suivi de la température ambiante d'une salle à partir d'un capteur LM335).

On pensera à intercaler entre le capteur et le canal d'entrée de l'ADC du PIC, un **filtre antirepliement** (filtre analogique passe-bas de fréquence de coupure FE 2). Il est également conseillé d'effectuer un **filtrage numérique** (passe-bas) de façon à enlever le maximum de bruit (en particulier le bruit de ronflement du secteur 50 Hz). Ce filtrage numérique est réalisé de manière logicielle par le PIC (pendant la phase de traitement du résultat de la conversion).

C. Fréquence d'échantillonnage et nombre de canaux :

Supposons que le module ADC travaille avec une fréquence d'échantillonnage de F E = 8 kHz. On obtient ainsi 8000 échantillons par seconde. Si n canaux sont utilisés, cela fait 8000/n échantillons par seconde et par canal. Ainsi, avec 7 canaux, chaque canal est échantillonné à seulement **F E / n** = 1,14 kHz. Ce n'est pas gênant pour l'acquisition de signaux lents (on pourra connecter 7 capteurs de température par exemple).

III.4. Le filtrage numérique :

Un filtrage numérique est effectué de façon à enlever le maximum de bruit. Ce filtrage numérique est réalisé de manière logicielle par le microcontrôleur PIC 16F88 (pendant la phase de traitement du résultat de la conversion de l'ADC).

Caractéristique du filtre numérique :

• Période d'échantillonnage nominale : Te = 390,625 µs
• Fréquence d'échantillonnage nominale : Fe = 1 / Te = 2,560 000 kHz

Le filtre numérique calcule la moyenne arithmétique sur 256 échantillons (filtre numérique à moyenne glissante).

- Choix de la fréquence d'échantillonnage du filtre :

La fréquence d'acquisition est de 1 Hz maximum par voie.
Comme il y a 7 voies, cela laisse 140 ms entre deux acquisitions. Il faut que **nTe** soit un multiple entier de la période du secteur (20 ms), tout en restant inférieur à 140 ms :
On prendra : **nTe = 100 ms** (soit 5 fois la période du secteur).
On a intérêt à choisir n le plus grand possible.
Te est limitée par la rapidité du convertisseur ADC interne du microcontrôleur PIC 16F88.
Dans notre cas, la durée de conversion est de 42 µs.
On prendra : **n = 256** (puissance de 2)
D'où :

Te = 100 ms / 256 = **390,625 µs** (ce qui est bien supérieur à 42 µs)
Fe = 2560,000 Hz
L'horloge du microcontrôleur PIC 16F88 est cadencée par un quartz de fréquence nominale
20 MHz : 1 cycle correspond donc à 0,2 µs.
390,625 µs = 1953,125 cycles
On arrondit à 1953 cycles :
Te = 390,6 µs

IV. LA LIAISON RS232 :

La communication de cette carte avec le PC se fait à travers une connexion série (DB9), reliée au microcontrôleur par ses broches 2 et 3 (RX, TX) via un circuitMAX232 dont le rôle est l'adaptation des signaux TTL/CMOS.

IV.1. Principe d'adaptation PIC- RS232 :

Passons maintenant au principe d'adaptation entre le PIC et le port série Rs232.

Les cartes électroniques à base de microcontrôleurs fonctionnent très souvent avec des niveaux TTL soit 0-5Volt, 0V pour le niveau 0 et 5Volt pour le niveau 1. Brancher donc directement une ligne RS232 sur un microcontrôleur n'aurait donc aucun sens et pourrait aussi endommager le système en imposant des tensions de 25volt.

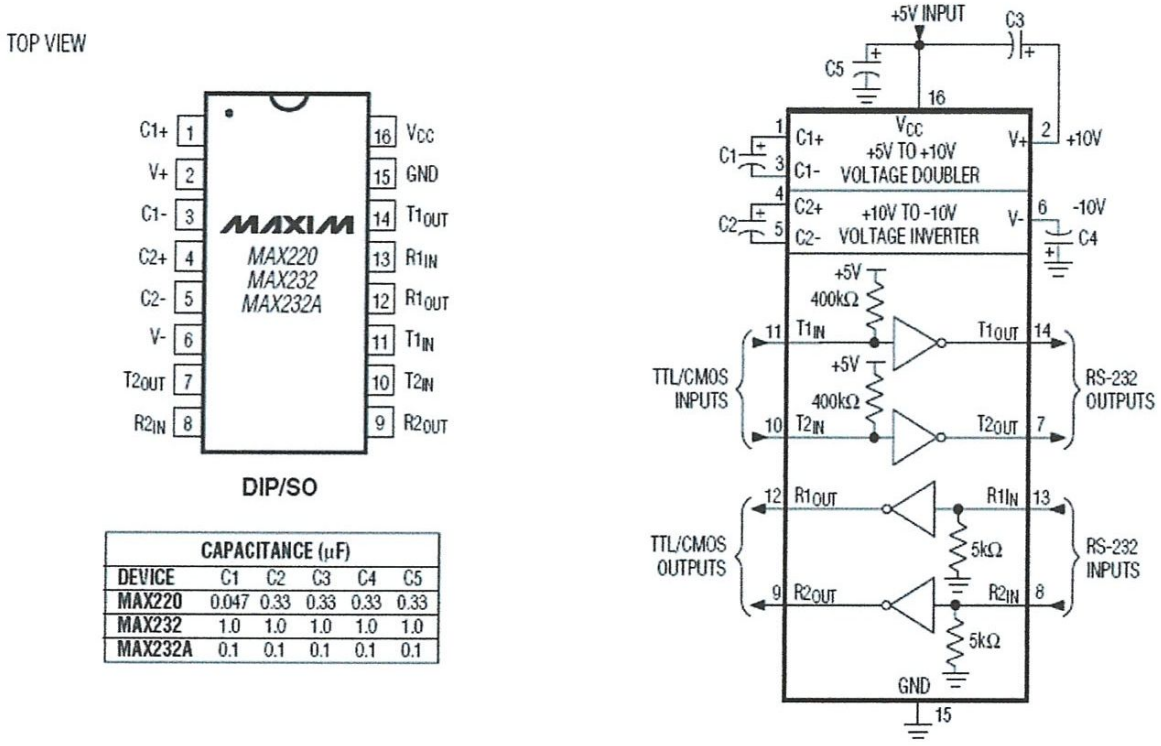
Pour rendre compatible une ligne RS232 avec une carte de ce type il existe un composant très simple d'utilisation c'est le max232. En regardant son schéma interne ci-dessous (Figure 6), nous constatons directement qu'il est premièrement doté d'un convertisseur de tension, au travers des capacités C1 et C3 il génère une tension de 10Volt depuis les 5Volt (doubleur de tension), et au moyen des capacités C2 et C4 il génère une tension de -10Volt a partir de la tension de 10Volt. Il est bien sur évident que la puce est munie de tout un système, avec un oscillateur, des diodes et ... afin d'intégré ce convertisseur DC-DC. Il existe une version de cette puce, le max233, où les capacités sont intégrées directement dedans, mais nous ne rentrerons pas dans ce détail. La valeur des capacités va dépendre de la version de la puce :

Nous aurons donc besoin d'un circuit chargé de convertir les niveaux des signaux entre PIC et PC. La broche TX du PIC émettra en 0V/5V et sera convertie en +12V/-12V vers notre PC.

La broche RX du PIC recevra les signaux en provenance du PC, signaux qui seront converti du +12V/-12V en 0V/5V par notre circuit de pilotage du bus. Notons que la liaison étant fullduplex, émission et réception sont croisées, chaque fil ne transitant l'information que dans un seul sens.

Nous utiliserons le célèbre circuit MAX232 pour effectuer cette adaptation de niveaux. Ce circuit contient un double convertisseur à double direction. Autrement dit, il dispose de :

- 2 blocs, dénommes T1 et T2, qui convertissent les niveaux entres en 0V/5V en signaux sortis sous +12V/-12V. En réalité, on n'a pas tout a fait +12V et -12V, mais plutôt de l'ordre de +8,5V/-8,5V ce qui reste dans la norme RS232.
- 2 blocs, dénommes R1 et R2, qui convertissent les niveaux entres en +12V/-12V en signaux sortis sous 0V/5V.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 3.6 : Circuit intégré MAX232

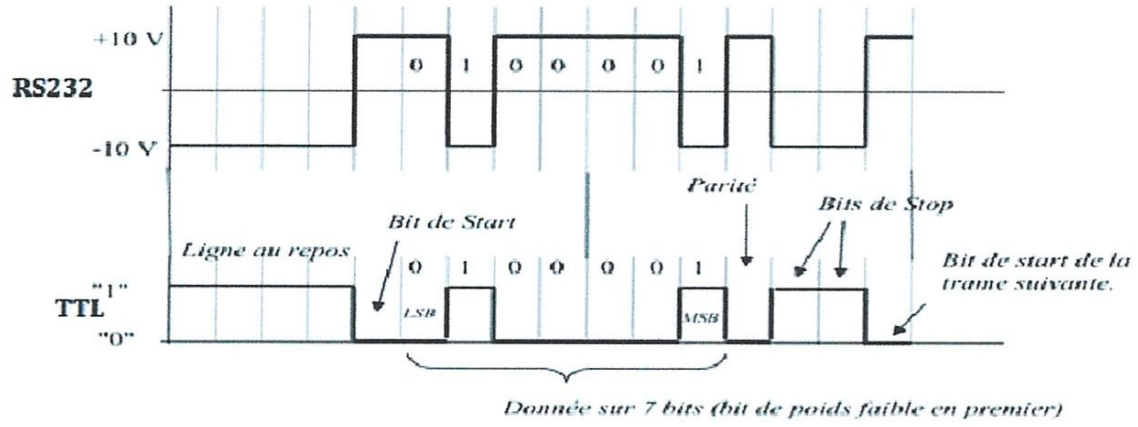


Figure 3.7 : Conversion TTL – RS232

IV.2. Protocole de communication entre l'ordinateur et la carte :

Seul l'ordinateur prend la parole (maître).
Le microcontrôleur PIC 16F88 se contente de répondre (esclave).
L'ordinateur envoie 1 octet (via la liaison RS232) toute les 140 ms.
Cet octet contient le numéro du canal à échantillonner :

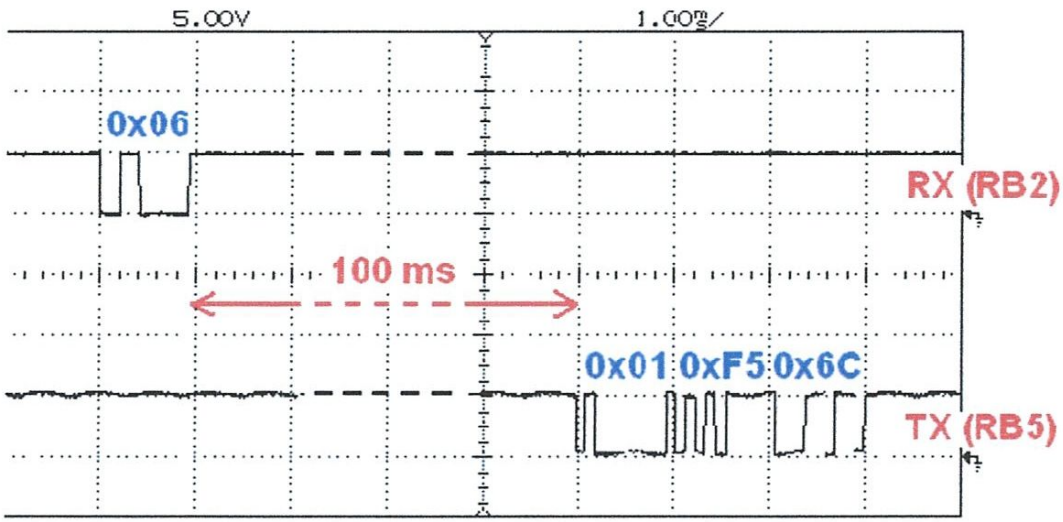
- canal 0 -> 0x00
- canal 1 -> 0x01
- canal 2 -> 0x02
- canal 3 -> 0x03
- canal 4 -> 0x04
- canal 5 -> 0x05
- canal 6 -> 0x06

Le processus est cyclique : voie 0 puis voie 1 ... puis voie 6 puis voie 0 ...
Chaque voie est ainsi échantillonnée toutes les 7 x 140 = 980 ms (d'où une fréquence d'acquisition d'environ 1 Hz). Une fois l'octet reçu, le PIC 16F88 sélectionne le canal indiqué, et lance une série de 256 conversions (une toute les 390,6 µs).Cela prend : 256 x 390,6 µs = 100 ms.
Le microcontrôleur effectue la moyenne des 256 conversions (filtrage numérique).
Sachant que la résolution du convertisseur ADC est 10 bits, la somme nécessite un nombre de 18 bits. La moyenne correspond à la somme avec la virgule décalée de 8 bits vers la gauche.
Par exemple :

Somme brute (18 bits) : 10 10010010 01000101 = 168 517 (en décimal). On décale la virgule de 8 positions : 10 10010010, 01000101
Partie entière (10 bits) : 10 10010010 = 658 (en décimal)
01000101 = 1/4 + 1/64 + 1/256 = 0,26953125
En définitive : 168 517 / 256 = 658,26953125
Notez que la résolution est améliorée : on passe de 10 bits à 18 bits.
On pratique, on restera prudent, et on considérera que la résolution est d'au moins 12 bits (soit une résolution de l'ordre de 5 V / 4096 ≈ 1 mV). La moyenne est envoyée telle quelle vers l'ordinateur sous la forme de 3 octets :

- 1er octet : (000000 D9 D8)
- 2ème octet : (D7 ... D0)
- 3ème octet : (D-1 ... D-8)

Exemple d'oscillogramme :



L'ordinateur envoie vers le PIC un octet : 0x06 <=> voie 6
100 ms plus tard, le PIC renvoie 3 octets :
0x01 0xF5 0x6C <=> 0x01F56C <=> 128 364 (en décimal)
128 364 / 256 = 501,42 <=> (501,42 / 1023) x 5 volts = **2,451 volts**
40 ms plus tard (140 - 100 = 40 ms), l'ordinateur envoie un nouvel octet (0x00) pour demander l'échantillonnage de la voie 0, et le cycle recommence ...

V. LISTE DE MATERIEL:

- Microcontrôleur PIC 16F8.
- 1 circuit intégré MAX233A (interface RS232C<-->TTL/CMOS).
- 7 résistances 4,7k.
- 7 condensateurs de 2200nf (ou 1000 nF).
- 1 quartz 20MHz.
- 1 condensateur électrochimique de 100µF (filtrage).
- 1 condensateur électrochimique de 10µF (filtrage).
- 1 condensateur électrochimique de 1µF (filtrage).
- 4 condensateurs de 100nF (filtrage).
- 2 condensateurs de 22pF.
- 1 source d'alimentation continue +12V (ou une pile de 9V).
- 1 régulateur 7805 (boîtier TO220).

La carte d'alimentation :

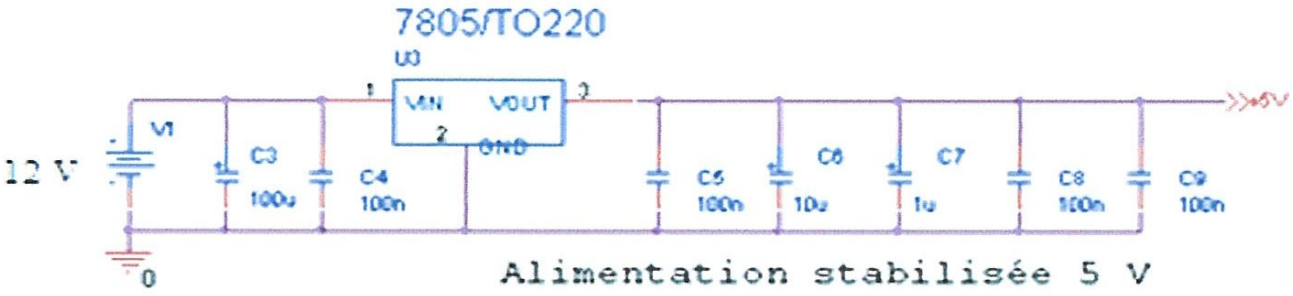


Figure 3.8 : schéma électrique de la carte d'alimentation

Notre carte doit être alimentée par une alimentation stabilisée qui fournit 5V a la sortie pour alimenter notre PIC.

La carte d'alimentation réalisée contient :

Un régulateur de tension 7805 : qui fournit une sortie 5V pour une tension d'entrée de 12v (on peut utiliser une pile de 9V). Les condensateurs C3 et C4 de (100nF) permettent de filtrer la tension d'entrée pour le régulateur, et les condensateurs C5 (100nF), C6 (10µF), C7 (1µF), C8 (100nF), C9 (100nF) permettent de lisser la sortie du régulateur pour la charge.

VI. SIMULATION AVEC ISIS-PROTEUS :

Avant de passer à la réalisation pratique de notre système nous avons eu recours à la simulation des différentes parties du système.
Pour cela on utilisé le logiciel ISIS qui est un très bon logiciel de simulation en électronique.
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.
La simulation permet d'ajuster et de modifier le circuit comme si on manipulait un montage réel. Ceci permet d'accélérer le prototypage et de réduire son coût.
Il faut toujours prendre en considération que les résultats obtenus de la simulation sont un peu différents de celles du monde réel, et ce dépend de la précision des modèles SPICE1 des composants et de la complication des montages.

1 **SPICE** (***S**imulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis*) est un logiciel de simulation généraliste de circuits électroniques analogiques. Il permet la simulation au niveau du composant (résistances, condensateurs, transistors) en utilisant différents types d'analyses.
ISIS est la composante de Proteus qui permet la création de schémas et la simulation électrique. La grande force de **ISIS** est de pouvoir simuler le comportement d'un microcontrôleur (PIC, Atmel, 8051, ARM, HC11...) et de son interaction avec les composants qui l'entourent.

II. CONCLUSION :

Dans ce chapitre on a vue les différents éléments constituants notre carte 'acquisition 7 voies pour les signaux lents, leurs fonctionnement ainsi que le principe e la transmission série et celui de l'adaptation en ligne.

III. RESUME :

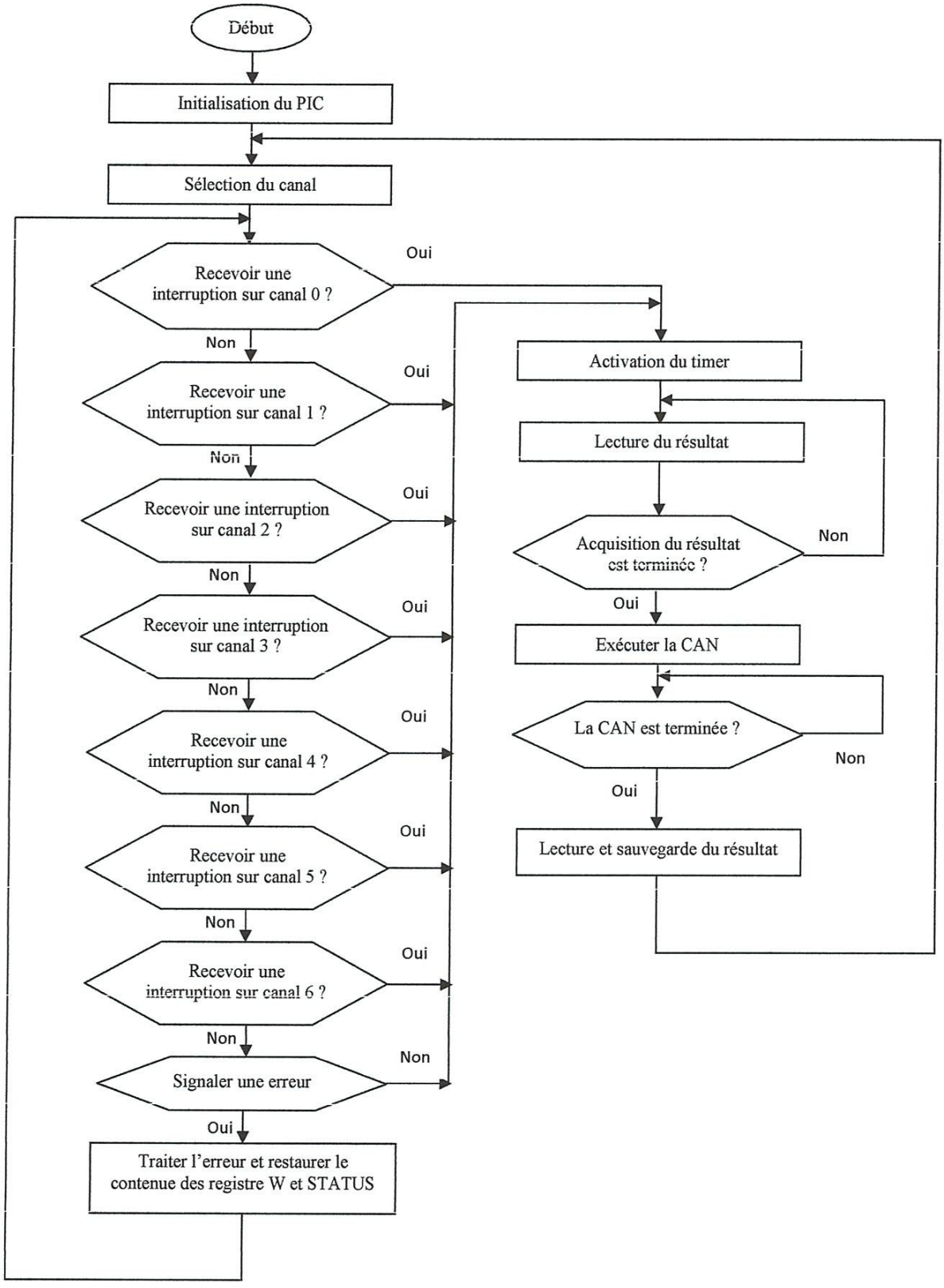
L'objectif de ce mémoire est la réalisation d'une carte d'acquisition 7 voies ıalogiques pour signaux lents.

Les fonctions essentielles de cette carte électronique réalisée sont : la conversion, filtrage analogique et numérique. Les résultats obtenus de la conversion de chaque ɔie sont affichées sur PC à partir d'une application de windows ***ADC108.exe.*** Cette ɔplication fonctionne sous windows XP.

Il ya d'autre possibilité de sauvegarder les données sur un fichier textes. 'avantage principal de ce projet c'est qu'on peut acquérir par exemple la température un ou plusieurs patients, selon le programme de notre carte qu'il a été créée avec ɛnvironnement de la programmation en assembleur en utilisant le fameux logiciel PLAB (dans notre cas on a utilisé MPLAB version 8)

Ce logiciel nous permettra de créer un programme, de l'assembler, et de le ɯuler, et c'est ce programme là qui nous permet de définir le fonctionnement de notre ırte.

I. L'ORGANIGRAMME QUI REPRESENTE LE FONCTIONNEMENT DE L'APPLICATION :



II. LE PROGRAMME EN ASSEMBLEUR :

```
; Carte d'acquisition 7 entrées analogiques 0 à 5 V
; Fréquence d'échantillonnage 2560 Hz (Te = 390,6 µs)

; Communication via RS232
; Utilisation de l'USART du 16F88 avec interruptions
; 9600 bauds/s      8 bits de données      Pas de bit de parité
; 1 bit de STOP     Pas de contrôle de flux

; Utilisation du module ADC 10 bits avec interruption de fin de conversion

; (C) Fabrice Sincère, octobre 2007
; http://perso.orange.fr/fabrice.sincere
; microcontrôleur PIC 16F88
; langage : assembleur
; développé avec Microchip MPLAB IDE

        Errorlevel-302 ; Supprime le message "Ensure that bank bits are
correct"

        List p=16F88    ; processeur utilisé
        #include <p16F88.inc>

        ;Program Configuration Register 1
        __CONFIG    _CONFIG1, _CP_OFF & _CCP1_RB0 & _DEBUG_OFF &
_WRT_PROTECT_OFF & _CPD_OFF & _LVP_OFF & _BODEN_ON & _MCLR_ON & _PWRTE_ON &
_WDT_OFF & _HS_OSC
                ;bits de configuration :
                ; Code protection OFF
                ; CCP1 function on RB0
                ; In-Circuit Debugger OFF
                ; FLASH Program Memory Write protection OFF
                ; Data EE Memory Code Protection OFF
                ; Low Voltage Programming OFF
                ; Brown-out Reset ON
                ; RA5/MCLR pin function is MCLR
                ; Power-up Timer ON
                ; Watchdog Timer OFF
                ; HS oscillator (quartz 20 MHz)
```

```
        ;Program Configuration Register 2
        __CONFIG    _CONFIG2, _IESO_OFF & _FCMEN_OFF
                ;bits de configuration :
                ;Internal External Switch Over mode OFF
                ;Fail-Safe Clock Monitor OFF

;xxxxxx
; macro
;xxxxxx

bank1   macro           ; passage en banque 1
                bsf STATUS,RP0
                bcf STATUS,RP1
                endm

bank0   macro           ; passage en banque 0
                bcf STATUS,RP0
                bcf STATUS,RP1
                endm

;xxxxxxxxxxxxxxxxxxxxxxxxxx
; Déclaration des variables
;xxxxxxxxxxxxxxxxxxxxxxxxxx

        CBLOCK 0x070            ; début de la zone des registres d'usage
général du 16F88
                                        ; (banque quelconque : 0,1,2
ou 3)
                                        ; 0x070 - 0x07F : 16
variables

        STATUS_TEMP : 1         ; sauvegarde du registre STATUS (routine
d'interruption)
        W_TEMP : 1      ; sauvegarde du registre W  (routine d'interruption)

        octet_rx : 1            ; octet reçu (broche RX de l'UART)
                                ; contient le numéro de canal à
échantillonner(0x00 à 0x06)
```

```
        octet1_tx : 1   ; 1er octet à transmettre : contient les 2 bits de
poids fort
                        ; de la sommme des 256 mesures (0 0 0 0 0 0 b17 b16)
        octet2_tx : 1   ; 2ème octet à transmettre : (b15 ... b8)
        octet3_tx : 1   ; 3ème octet à transmettre : (b7 ... b0)
        nb_octet_transmis : 1  ; compteur du nombre d'octets transmis (0, 1,
2 ou 3)

        nb_mesures : 1 ; compteur du nombre de conversion réalisées (0x01 à
0x00 = 256)

        adresh_bak : 1  ; sauvegarde du registre ADRESH
        adresl_bak : 1  ; sauvegarde du registre ADRESL

        ENDC

;xxxxxxxxxxxxxxxxxxxx
; Démarrage sur reset
;xxxxxxxxxxxxxxxxxxxx

        org 0x0000
        goto initialisation

; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; Routine d'interruption
; 4 sources d'interruption :
; - réception (UART)
; - émission (UART)
; - module ADC (interruption de fin de conversion)
; - timer1 (période d'échantillonnage de 390,6 µs)
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

        org 0x0004                          ; vecteur d'interruption

        movwf W_TEMP
        swapf STATUS,W
        movwf STATUS_TEMP               ; sauvegarde du registre W puis du
registre STATUS


    bank1
```

```
    btfss PIE1, RCIE
    goto int1
    bank0
    btfsc PIR1, RCIF
    goto reception
int1
        bank1
    btfss PIE1, TXIE
    goto int2
    bank0
    btfsc PIR1, TXIF
    goto emission
int2
        bank1
    btfss PIE1, ADIE
    goto int3
    bank0
    btfsc PIR1, ADIF
    goto adc
int3
        bank1
    btfss PIE1, TMR1IE
    goto int4
    bank0
    btfsc PIR1, TMR1IF
    goto timer1
int4
        goto restauration

        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ; traitement de l'interruption de réception de l'USART
        ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
reception

    bank0
    movf RCREG , W ; N.B. le flag RCIF est remis à 0 par une lecture du
registre RCREG
    movwf octet_rx ; les 8 bits de données sont transférés dans (octet_rx)

        ; on teste s'il s'agit du canal 0
    movlw .0 ; W = .0
```

```
    subwf octet_rx , W ; W = (octet_rx) - .0
    btfss STATUS, Z ; test du bit Z
    goto canal1 ; Z = 0 c'est-à-dire (octet_rx) != .0
    ; (octet_rx) = .0 (canal 0)
        ; Sélection du canal 0
        bcf ADCON0 , CHS2 ; CHS2 = 0
        bcf ADCON0 , CHS1 ; CHS1 = 0
        bcf ADCON0 , CHS0 ; CHS0 = 0
        goto activation_timer1

canal1
        ; on teste s'il s'agit du canal 1
        movlw .1 ; W = .1
    subwf octet_rx , W ; W = (octet_rx) - .1
    btfss STATUS, Z ; test du bit Z
    goto canal2 ; Z = 0 c'est-à-dire (octet_rx) != .1
    ; (octet_rx) = .1 (canal 1)
        ; Sélection du canal 1
        bcf ADCON0 , CHS2 ; CHS2 = 0
        bcf ADCON0 , CHS1 ; CHS1 = 0
        bsf ADCON0 , CHS0 ; CHS0 = 1
        goto activation_timer1

canal2
        movlw .2 ; W = .2
    subwf octet_rx , W ; W = (octet_rx) - .2
    btfss STATUS, Z ; test du bit Z
    goto canal3 ; Z = 0 c'est-à-dire (octet_rx) != .2
    ; (octet_rx) = .2 (canal 2)
        ; Sélection du canal 2
        bcf ADCON0 , CHS2 ; CHS2 = 0
        bsf ADCON0 , CHS1 ; CHS1 = 1
        bcf ADCON0 , CHS0 ; CHS0 = 0
        goto activation_timer1

canal3
        movlw .3 ; W = .3
    subwf octet_rx , W ; W = (octet_rx) - .3
    btfss STATUS, Z ; test du bit Z
    goto canal4 ; Z = 0 c'est-à-dire (octet_rx) != .3
    ; (octet_rx) = .3 (canal 3)
```

```
        ; Sélection du canal 3
        bcf ADCON0 , CHS2 ; CHS2 = 0
        bsf ADCON0 , CHS1 ; CHS1 = 1
        bsf ADCON0 , CHS0 ; CHS0 = 1
        goto activation_timer1

canal4
        movlw .4 ; W = .4
    subwf octet_rx , W ; W = (octet_rx) - .4
    btfss STATUS, Z ; test du bit Z
    goto canal5 ; Z = 0 c'est-à-dire (octet_rx) != .4
    ; (octet_rx) = .4 (canal 4)
        ; Sélection du canal 4
        bsf ADCON0 , CHS2 ; CHS2 = 1
        bcf ADCON0 , CHS1 ; CHS1 = 0
        bcf ADCON0 , CHS0 ; CHS0 = 0
        goto activation_timer1

canal5
        movlw .5 ; W = .5
    subwf octet_rx , W ; W = (octet_rx) - .5
    btfss STATUS, Z ; test du bit Z
    goto canal6 ; Z = 0 c'est-à-dire (octet_rx) != .5
    ; (octet_rx) = .5 (canal 5)
        ; Sélection du canal 5
        bsf ADCON0 , CHS2 ; CHS2 = 1
        bcf ADCON0 , CHS1 ; CHS1 = 0
        bsf ADCON0 , CHS0 ; CHS0 = 1
        goto activation_timer1

canal6
        movlw .6 ; W = .6
    subwf octet_rx , W ; W = (octet_rx) - .6
    btfss STATUS, Z ; test du bit Z
    goto canal_erreur ; Z = 0 c'est-à-dire (octet_rx) != .6
    ; (octet_rx) = .6 (canal 6)
        ; Sélection du canal 6
        bsf ADCON0 , CHS2 ; CHS2 = 1
        bsf ADCON0 , CHS1 ; CHS1 = 1
        bcf ADCON0 , CHS0 ; CHS0 = 0
        goto activation_timer1
```

```
canal_erreur
        ; (octet_rx) a une valeur non valide
        ; (liste des valeurs valides : 0, 1, 2, 3, 4, 5 et 6)
        goto overrun

        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ; Activation du TIMER1 (16 bits)
        ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
activation_timer1

        bcf PIR1 , TMR1IF ; on efface le drapeau de l'interruption
        clrf nb_mesures
        clrf octet1_tx
        clrf octet2_tx
        clrf octet3_tx
        clrf TMR1H

        bank1
        bsf PIE1 , TMR1IE ; autorisation de l'interruption du TIMER1 (16
bits)
        bank0

        bsf PIR1 , TMR1IF ; on force à 1 le drapeau de l'interruption
                                        ; => interruption provoquée
        goto overrun

        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
overrun
        ; test d'une erreur d'overrun
        bank0
        btfss RCSTA , OERR
        goto restauration
        ; traitement de l'erreur d'overrun
        bcf RCSTA , CREN ; on efface le bit OERR
        bsf RCSTA , CREN ; on relance la réception
        goto restauration



        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
        ; Traitement de l'interruption du Timer1
        ; on arrive ici toutes les 390,6 µs (période d'échantillonnage)
        ; 390,6 µs = 1953 cycles (avec un quartz de 20MHz)
        ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
timer1

        incf nb_mesures , f ; incrémentation du compteur

        ; le timer1 est incrémenté de 0xF866 = 63590 (65536 - 1953 + 7
instructions)
        bcf T1CON , TMR1ON              ; Stops TIMER1
        movlw 0x66
        addwf TMR1L , f ; (TMR1L) = (TMR1L) + 0x66
        btfsc STATUS, C ; test du bit C (Carry)
        incf TMR1H , f
        movlw 0xF8
        addwf TMR1H , f ; (TMR1H) = (TMR1H) + 0xF8
        bsf T1CON , TMR1ON              ; Enables TIMER1

        ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ; Attente pendant la phase d'acquisition (environ 19,7 µs)
        ; 19,7 µs  = 98 cycles (avec un quartz 20 MHz)
acquisition
        movlw .158 ; 256 - 98
        movwf TMR0 ; (TMR0) = .158
        bcf INTCON , TMR0IF ; on efface le drapeau du timer0
attente
        btfss INTCON , TMR0IF
        goto attente

        ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ; Lancement de la phase de conversion de l'ADC
        bsf ADCON0 , GO ; GO = 1

        bcf PIR1 , TMR1IF ; on efface le drapeau de l'interruption Timer1
        goto restauration



        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
        ; Traitement de l'interruption de fin de conversion du module ADC
        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
adc
        ; Attente avant une nouvelle acquisition (2 TAD = 3,2 µs)
        ; 3,2 µs  = 16 cycles (avec un quartz 20 MHz)
        movlw .240 ; 256 - 16
        movwf TMR0 ; (TMR0) = .240
        bcf INTCON , TMR0IF ; on efface le drapeau du timer0
attente1
        btfss INTCON , TMR0IF
        goto attente1

        ; lecture et sauvegarde du résultat de la conversion
    movf ADRESH , W ; W = (ADRESH)
    movwf adresh_bak ; (adresh_bak) = (ADRESH)
        bank1
        movf ADRESL , W ; W = (ADRESL)
    movwf adresl_bak ; (adresl_bak) = (ADRESL)

        bank0
        ; sommme =
        ; (octet1_tx) (octet2_tx) (octet3_tx) = nombre de 18 bits
        ; (000000 b17 b16 ... b0)
        ; resultat de conversion =
        ; (adresh_bak) (adresl_bak) = nombre de 10 bits
        ; (000000 d9 ... d0)
        ; somme = somme + resultat de conversion

        movf adresl_bak , W ; W = (adresl_bak)
        addwf octet3_tx , f ; (octet3_tx) = (octet3_tx) + adresl_bak
        btfsc STATUS, C ; test du bit C (Carry)
        goto som0
        goto som1
som0
        incf octet2_tx , f ; (octet2_tx) = (octet2_tx) + 0x01
        btfsc STATUS, Z ; test du bit Z (Zero)
        incf octet1_tx , f ; (octet1_tx) = (octet1_tx) + 0x01
som1
        movf adresh_bak , W ; W = (adresh_bak)
        addwf octet2_tx , f ; (octet2_tx) = (octet2_tx) + adresh_bak
        btfsc STATUS, C ; test du bit C (Carry)
```

```
        incf octet1_tx , f ; (octet1_tx) = (octet1_tx) + 0x01

        bcf PIR1 , ADIF ; on efface le drapeau de l'interruption du module
ADC

        ; on teste si le compteur = 0x00 (= 256 ème mesure)
    movf nb_mesures , f ; (nb_mesures) = (nb_mesures)
    btfss STATUS , Z ; test du bit Z
    goto restauration ; Z = 0 c'est-à-dire (nb_mesures) != 0x00
        ; 256 ème mesure
        bank1
        bsf PIE1 , TXIE ; autorisation de l'interruption d'émission de
l'USART
        bcf PIE1 , TMR1IE ; interdiction de l'interruption du TIMER1 (16
bits)
        bank0
        goto restauration

        ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ; Traitement de l'interruption d'émission de l'USART
        ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
emission

        ; 3 octets à transmettre
        incf nb_octet_transmis , f     ; on incrémente (nb_octet_transmis)

        movlw D'1'
        subwf nb_octet_transmis , W
        btfss STATUS , Z
        goto emm2
        ; émission du 1er octet
        movf octet1_tx  , W

        ;bank0
        movwf TXREG ; transmission
                              ; N.B. le flag TXIF est remis à 0 par une
écriture dans le registre TXREG
        goto int1  ; pour transmettre le plus rapidement possible le 2ème
octet

emm2
```

```
        movlw D'2'
        subwf nb_octet_transmis , W
        btfss STATUS , Z
        goto emm3
        ; émission du 2ème octet
        movf octet2_tx  , W

        ;bank0
        movwf TXREG ; transmission
                              ; N.B. le flag TXIF est remis à 0 par une
écriture dans le registre TXREG
        goto int1  ; pour transmettre le plus rapidement possible le 3ème
octet

emm3
        ; émission du 3ème octet
        movf octet3_tx  , W
        ;bank0
        movwf TXREG ; transmission
                              ; N.B. le flag TXIF est remis à 0 par une
écriture dans le registre TXREG

        bank1
        bcf PIE1 , TXIE ; interdiction de l'interruption d'émission de
l'USART
        bank0
        clrf nb_octet_transmis
        goto restauration

;xxxxxxxxxxx
restauration

        swapf STATUS_TEMP,W           ; restauration des registres STATUS
puis W
        movwf STATUS
        swapf W_TEMP,f
        swapf W_TEMP,W

        retfie

;xxxxxxxxxxxxxxx
```

```
; Initialisation
;xxxxxxxxxxxxxxx

initialisation

        bank0
        clrf PORTA      ; mise à 0 des sorties du port A
        clrf PORTB              ; mise à 0 des sorties du port B

        bank1

        movlw B'11011000'
        movwf OPTION_REG
        ; bit 7 (/RBPU) = 1 : valeur par défaut (option non utilisée)
        ; bit 6 (INTEDG) = 1 : valeur par défaut (option non utilisée)
        ; bit 5 (T0CS) = 0 : l'horloge interne est l'horloge du timer0
        ; bit 4 (T0SE) = 1 : valeur par défaut (option non utilisée)
        ; bit 3 (PSA) = 1
        ; bit 2 (PS2) = 0
        ; bit 1 (PS1) = 0
        ; bit 0 (PS0) = 0
        ; Prescaler du Timer0 = 1:1
        ; Watchdog sans prescaler (1:1)
        ; le timer0 déborde toutes les 256*0,2 µs = 51,2 µs

        movlw B'11111111'
        movwf TRISA
        ; bit 0 du port A (RA0) = 1 : configuration en entrée (canal 0 du
module ADC)
        ; bit 1 du port A (RA1) = 1 : configuration en entrée (canal 1 du
module ADC)
        ; bit 2 du port A (RA2) = 1 : configuration en entrée (canal 2 du
module ADC)
        ; bit 3 du port A (RA3) = 1 : configuration en entrée (canal 3 du
module ADC)
        ; bit 4 du port A (RA4) = 1 : configuration en entrée (canal 4 du
module ADC)
        ; bit 5 du port A (RA5) = X : configuration en entrée, par exemple
        ; bit 6 du port A (RA6) = X : configuration en entrée, par exemple
        ; bit 7 du port A (RA7) = X : configuration en entrée, par exemple
```

```
        movlw B'11111111'
        movwf TRISB
        ; bit 0 du port B (RB0) = X : configuration en entrée, par exemple
        ; bit 1 du port B (RB1) = X : configuration en entrée, par exemple
        ; bit 2 du port B (RB2) = 1 : configuration en entrée (RX : USART)
        ; bit 3 du port B (RB3) = X : configuration en entrée, par exemple
        ; bit 4 du port B (RB4) = X : configuration en entrée, par exemple
        ; bit 5 du port B (RB5) = 1 : configuration en entrée (TX : USART)
        ; bit 6 du port B (RB6) = 1 : configuration en entrée (canal 5 du
module ADC)
        ; bit 7 du port B (RB7) = 1 : configuration en entrée (canal 6 du
module ADC)

        movlw B'01111111'
        movwf ANSEL
        ; bit 7 du registre ANSEL = 0 : non implémenté
        ; bit 6 du registre ANSEL = 1 : configuration de la broche RB7/AN6
comme entrée analogique (canal 6)
        ; bit 5 du registre ANSEL = 1 : configuration de la broche RB6/AN5
comme entrée analogique (canal 5)
        ; bit 4 du registre ANSEL = 1 : configuration de la broche RA4/AN4
comme entrée analogique (canal 4)
        ; bit 3 du registre ANSEL = 1 : configuration de la broche
RA3/AN3/VREF+ comme entrée analogique (canal 3)
        ; bit 2 du registre ANSEL = 1 : configuration de la broche
RA2/AN2/VREF- comme entrée analogique (canal 2)
        ; bit 1 du registre ANSEL = 1 : configuration de la broche RA1/AN1
comme entrée analogique (canal 1)
        ; bit 0 du registre ANSEL = 1 : configuration de la broche RA0/AN0
comme entrée analogique (canal 0)

        ; Configuration du module ADC

        ; tension de référence haute : VDD (5 V)
        ; tension de référence basse : VSS (0 V)
    bcf ADCON1 , VCFG0          ; VCFG0 = 0
    bcf ADCON1 , VCFG1          ; VCFG1 = 0
        ; Choix du format du résultat de la conversion
        bsf ADCON1 , ADFM
                ; ADRESH = (0 0 0 0 0 0 b9 b8)
                ; ADRESL = (b7 b6 b5 b4 b3 b2 b1 b0)
```

```
    ; Choix de la fréquence d'horloge du convertisseur ADC
            ; F AD = F OSC / 32 = 625 kHz
            ; T AD = 1,6 µs
    bcf ADCON1 , ADCS2 ; ADCS2 = 0
    bank0
    bsf ADCON0 , ADCS1 ; ADCS1 = 1
    bcf ADCON0 , ADCS0 ; ADCS0 = 0
    bank1

    ; configuration de la liaison RS232
movlw D'129'
movwf SPBRG ; (SPBRG) = D'129'

    movlw B'00100100'
    movwf TXSTA
    ; bit 7 (CSRC) = 0 (non utilisé : 0 par exemple)
    ; bit 6 (TX9) = 0 : 8 bits de transmission
    ; bit 5 (TXEN) = 1 : autorise la réception
    ; bit 4 (SYNC) = 0 : mode asynchrone
    ; bit 3 = 0 (non implémenté)
    ; bit 2 (BRGH) = 1 : mode asynchrone haute vitesse
    ; bit 1 (TRMT) = 0 (en lecture seule)
    ; bit 0 (TX9D) = 0 (non utilisé : 0 par exemple)

    bank0
    movlw B'10010000'
    movwf RCSTA
    ; bit 7 (SPEN) = 1 : utilisation du port série
    ; bit 6 (RX9) = 0 : 8 bits de réception
    ; bit 5 (SREN) = 0 (non utilisé : 0 par exemple)
    ; bit 4 (CREN) = 1 : autorise la réception
    ; bit 3 (ADDEN) = 0 (non utilisé : 0 par exemple)
    ; bit 2 (FERR) = 0 (en lecture seule)
    ; bit 1 (OERR) = 0 (en lecture seule)
    ; bit 0 (RX9D) = 0 (non utilisé : 0 par exemple)

    clrf PORTA      ; mise à 0 des sorties du port A
    clrf PORTB              ; mise à 0 des sorties du port B
    clrf nb_octet_transmis
    clrf nb_mesures
```

```
        bcf PIR1 , ADIF ; on efface le drapeau de l'interruption du module
ADC

    bank1
    bsf PIE1 , RCIE ; autorisation de l'interruption de réception de
l'USART
        bcf PIE1 , TXIE ; interdiction de l'interruption d'émission de
l'USART
        bsf PIE1 , ADIE ; autorisation de l'interruption du module ADC
        bsf INTCON , PEIE ; autorisation des interruptions des
périphériques
    bsf INTCON, GIE ; autorisation globale des interruptions

        bank0
        ; Mise en service du convertisseur ADC
        bsf ADCON0 , ADON ; ADON = 1

        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ; Configuration du TIMER1 (16 bits) en mode timer
        ; Utilisation de l'interruption
        ; Débordement (0xFFFF -> 0x0000)
        ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

        bcf T1CON , T1CKPS1
        bcf T1CON , T1CKPS0     ; prescaler 1:1
        bcf T1CON , TMR1CS              ; Timer1 Clock Source = Internal
clock (FOSC/4)
        bsf T1CON , TMR1ON              ; Enables TIMER1

        goto debut_programme

;xxxxxxxxxxxxxxxxxxxxx
; Programme principal
;xxxxxxxxxxxxxxxxxxxxx

debut_programme

        goto debut_programme ; on attend une interruption

        END
```

Conclusion

CONCLUSION GENERALE :

Le travail qui nous a été confié dans le cadre de notre mémoire de master 2 en électronique est un travail de développement technologique ayant pour objectif la réalisation d'une carte d'acquisition avec 7 entrées analogiques pour des signaux lents.
Par exemple le signal de température et de pression atmosphérique sont des signaux d'évolution lente (de période très supérieure à la seconde).

Le cœur de notre carte est un microcontrôleur de la famille Microchip : le PIC 16F88.

L'utilisation du PIC nous a introduits dans le monde des microcontrôleurs : leurs mises en œuvre, leur système de développements ainsi que les différentes fonctionnalités et avenages qu'ils offrent.

Ce travail nous a permet aussi d'apprendre le logiciel ISIS pour la simulation.

Toutes ces choses sont nouvelles pour moi. Ceci m'a procuré un plaisir réel et a insufflé, à chaque fois, en moi un nouveau souffle (particulièrement pendant les jours où à chaque instant j'apprends quelque chose de nouveau) pour aller de l'avent. En effet, mon souci majeur était et est toujours de mener à bien et le plus loin possible e travail.

Dans la mesure où notre projet est fondamentalement destiné à la collecte de signaux analogiques par un microcontrôleur en vue de leur traitement à des fins de supervision, j'ai passé beaucoup de temps à étudier le microcontrôleur en particulièrement le microcontrôleur 16F88 (que j'ai utilisé pour ma carte) que je n'ai pas eu la chance de l'étudier pendant mon premier parcours à l'université. J'ai commencé par écrire au départ de tout petits programme, histoire de me familiariser avec le microcontrôleur, pour arriver en fin a comprendre mon

programme qui plus important et plus compliqué !! Et ce n'était pas une mince affaire.

Parallèlement, nous nous somme astreints à étudier également le protocole RES232 et le circuit MAX232, auquel notre application fait appel.

Personnellement, je pense que la réalisation de cette carte est très utile. Elle peut être utilisée dans diverses utilisations telles que :
Dans le domaine de la santé, cette carte peut surveiller un ou plusieurs patients (par exemple sa température en même temps que son tracé cardiaque l'ECG).
Dans le domaine industriel, cette carte peut être utilisée pour l'acquisition de la température ou la pression atmosphérique en utilisant seulement des capteurs (comme LM35 pour la température).

Mon seul souci étant d'arriver à achever un travail qui soit globalement homogène, instructif et concret.

En conclusion, je pense sincèrement avoir compris bien des choses lors de l'élaboration de ce travail, la (ou les) méthodes (s) d'approche du sujet afin de mieux définir ses différents contours.

Je souhaite profondément que ce travail soit bénéfique à d'autres étudiants. Nous avons tenu à ce qu'il soit aisément exploitable tant sur le plan de la méthodologie que du point de vue matériel et logiciel.

Nous ne prétendant pas que ce travail soit complètement achevé, nous pensons qu'il peut être certainement amélioré c'est ce que j'espère !!

Annexe

Organisation de la mémoire registres:

Il s'agit de mot de 8 bits, rangés sur 4 pages.

FIGURE 2-3: PIC16F88 REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	WDTCON	105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
	07h		87h		107h		187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(1)	18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh	Reserved(1)	18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h		91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
CCPR1H	16h		96h	General Purpose Register 16 Bytes		General Purpose Register 16 Bytes	
CCP1CON	17h		97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah		9Ah				
	1Bh	ANSEL	9Bh				
	1Ch	CMCON	9Ch				
	1Dh	CVRCON	9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes	
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h-7Fh	1F0h
	7Fh		FFh		17Fh		1FFh

- # Jeu d'instructions

A- Instructions manipulant des données immédiates :

W : registre de travail (accumulateur), taille 8 bits

k : valeur littérale, taille 8 bits

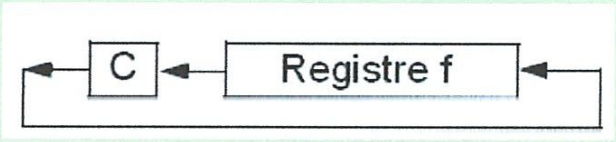
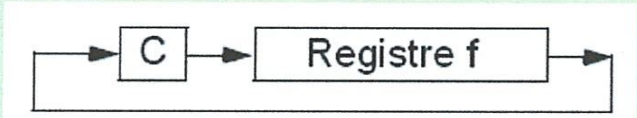
Mnémonique, opérande	Description	bit du registre STATUS affecté	nombre de cycles
MOVLW k	k (8 bits) est chargé dans (W)	-	1
ADDLW k	Additionne k (8 bits) et (W) et place le résultat dans (W)	C, DC , Z	1
SUBLW k	Soustrait W de k (8 bits) et place le résultat dans (W) k - (W) -> (W)	C, DC , Z	1
ANDLW k	Réalise un ET logique entre k (8 bits) et (W), et place le résultat dans (W)	Z	1
IORLW k	Réalise un OU logique (inclusif) entre k (8 bits) et (W), et place le résultat dans (W)	Z	1
XORLW k	Réalise un OU exclusif entre k (8 bits) et (W), et place le résultat dans (W)	Z	1

B- Instructions manipulant des données situées dans des registres :

f : registre (spécial ou d'usage général.

d : registre de destination (on peut choisir entre *le registre de travail W et le registre f)*

Mnémonique , opérande	Description	bit du registre STATUS affecté	nombre de cycles
MOVWF f	(W) est chargé dans (f)	-	1
MOVF f , d	(f) (8 bits) est chargé dans (destination)	Z	1
ADDWF f , d	Additionne le contenu du registre f (8 bits) et (W), et place le résultat dans (destination)	C, DC , Z	1
SUBWF f , d	Soustrait (W) de (f) (8 bits) et place le résultat dans (destination). (f) - (W) ->(destination)	C, DC , Z	1
ANDWF f , d	Réalise un ET logique entre (f) (8 bits) et (W), et place le résultat dans (destination)	Z	1
IORWF f , d	Réalise un OU logique (inclusif) entre (f) (8 bits) et (W), et place le résultat dans (destination)	Z	1
XORWF f , d	Réalise un OU exclusif entre (f) (8 bits) et (W), et place le résultat dans (destination)	Z	1
COMF f , d	Réalise le complément logique de (f) (8 bits), et place le résultat dans (destination)	Z	1
DECF f , d	Décrémente (f) et place le résultat dans (destination). (f) - 1 -> (destination)	Z	1
DECFSZ f , d	Décrémente (f) et place le résultat dans (destination). Si le résultat est 0, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place (soit 2 cycles)	-	1 ou 2
INCF f , d	Incrémente (f) et place le résultat dans (destination). (f) + 1 -> (destination)	Z	1
INCFSZ f , d	Incrémente (f) et place le résultat dans (destination). Si le résultat est 0, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place (soit 2 cycles)	-	1 ou 2

CLRF f	Efface le contenu du registre (f). Remarque : le bit Z est donc mis à 1.	Z	1
CLRW	Efface le contenu de l'accumulateur (W). Remarque : le bit Z est donc mis à 1.	Z	1
RLF f , d	Réalise une rotation circulaire à gauche :  Le résultat est placé dans (destination).	C	1
RRF f , d	Réalise une rotation circulaire à droite :  Le résultat est placé dans (destination).	C	1
SWAPF f , d	Les 4 bits de poids forts et les 4 bits de poids faibles de (f) sont échangés. Le résultat est placé dans (destination).	-	1
NOP	Cette instruction ne fait rien (durée 1 cycle).	-	1

C- Instructions manipulant des bits situés dans des registres :

f : registre (spécial ou d'usage général).

b : position du bit (0 à 7).

Mnémonique , opérande	Description	bit du registre STATUS affecté	nombre de cycles
BCF f , b	Mise à 0 du b ème bit du registre f	-	1

BSF f , b	Mise à 1 du b ème bit du registre f	-	1
BTFSC f , b	Si le b ème bit du registre f est égal à 0, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place (soit 2 cycles)	-	1 ou 2
BTFSS f , b	Si le b ème bit du registre f est égal à 1, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place (soit 2 cycles)	-	1 ou 2

D- Autres instructions/

L : label (étiquette).

Mnémonique, opérande	Description	bit du registre STATUS affecté	nombre de cycles
GOTO L	Branchement à l'adresse L	-	2
CALL L	Appelle un sous-programme (subroutine) situé à l'adresse L	-	2
RETURN	Retour de sous-programme	-	2
RETLW k	Retour de sous-programme, avec chargement de la valeur littérale k (8 bits) dans (W)	-	2
RETFIE	Retour de sous-programme d'interruption	-	2
CLRWDT	Efface le Watchdog	/TO, /PD	1
SLEEP	Place le microcontrôleur en mode sommeil	/TO, /PD	1

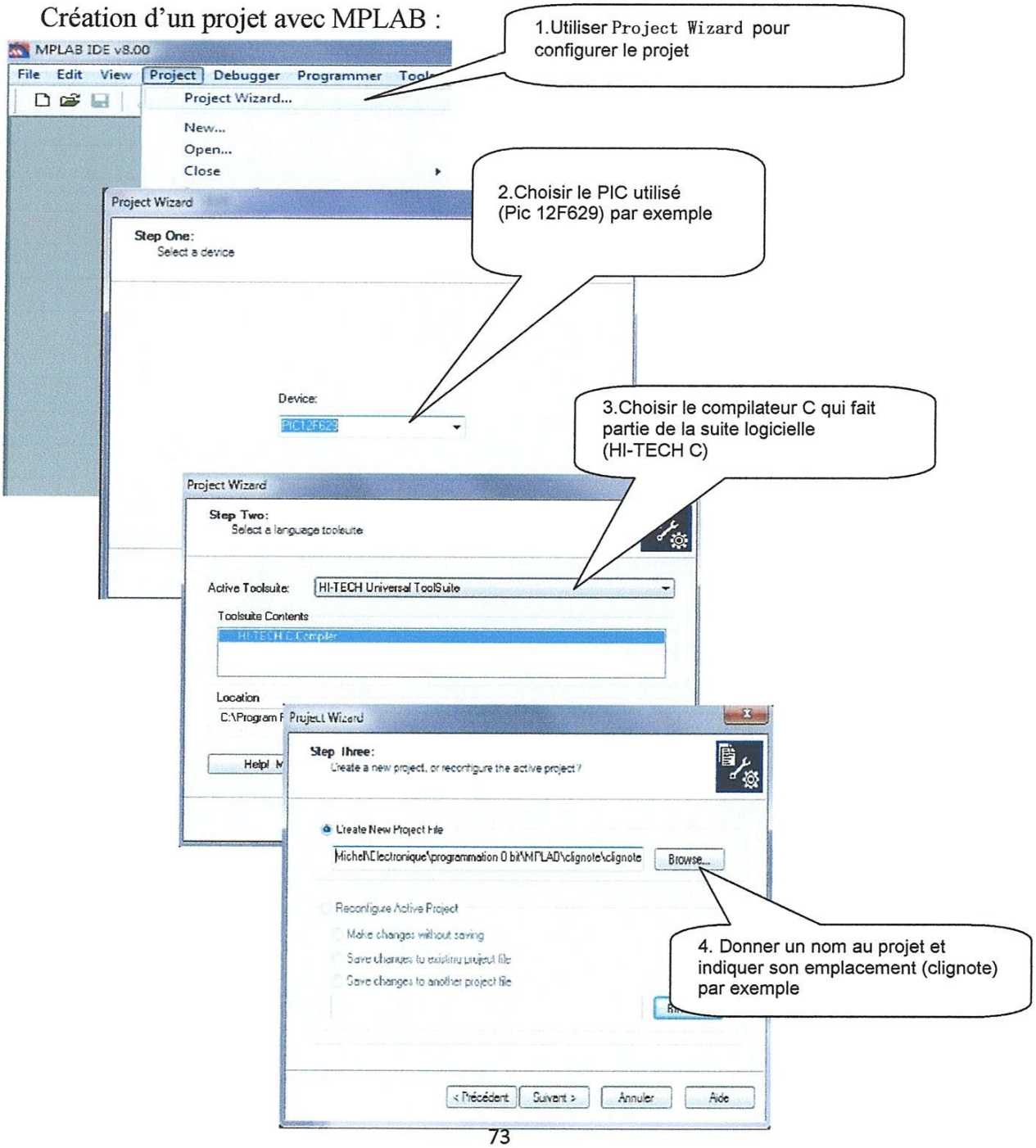
Une instruction est codée par un mot de 14 bits.

Une instruction nécessite 1 cycle, ou bien 2 cycles dans le cas d'une instruction de branchement (GOTO, CALL ...).

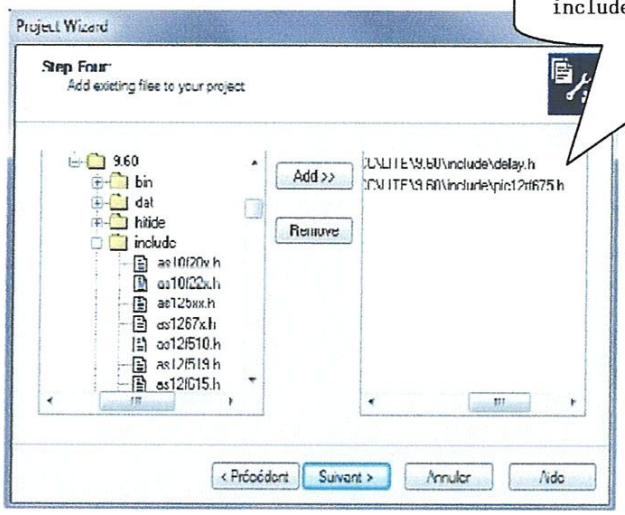
Avec une horloge à quartz de 20 MHz, un cycle correspond à $4/(20.10^6) = 200$ nanosecondes.

Le microcontrôleur peut donc exécuter jusqu'à 5 millions d'instructions par seconde (5 MIPS) !

Création d'un projet avec MPLAB :



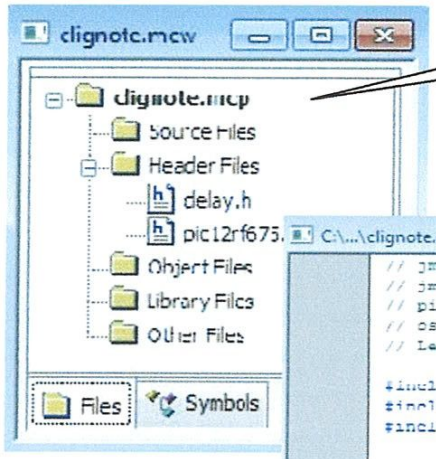
5.Placer les 2 fichiers d'entête nécessaires au programme. Ils se trouvent dans le dossier `include` et `samples` du compilateur C



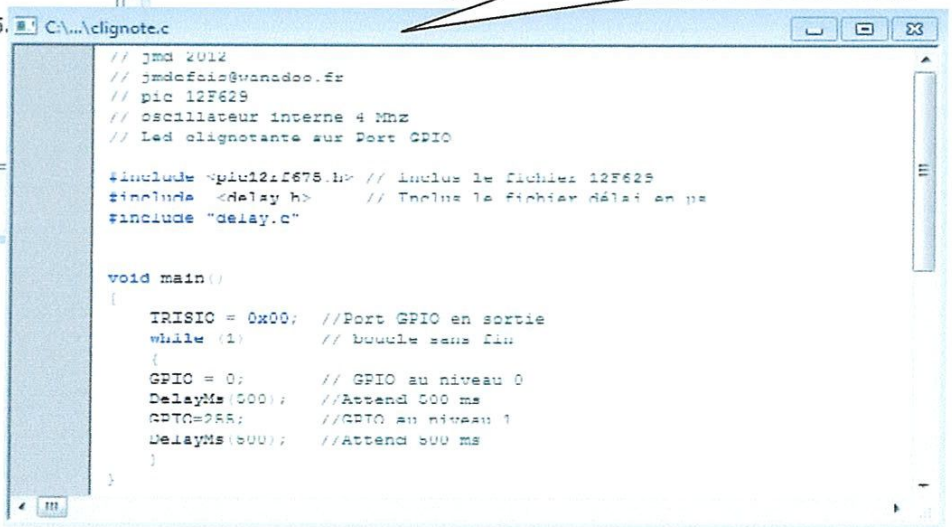
Pour cet exemple les 2 fichiers sont nécessaires :
`pic12rf675.h` il convient pour un Pic 629
`delay.h`

Ecrire le programme :

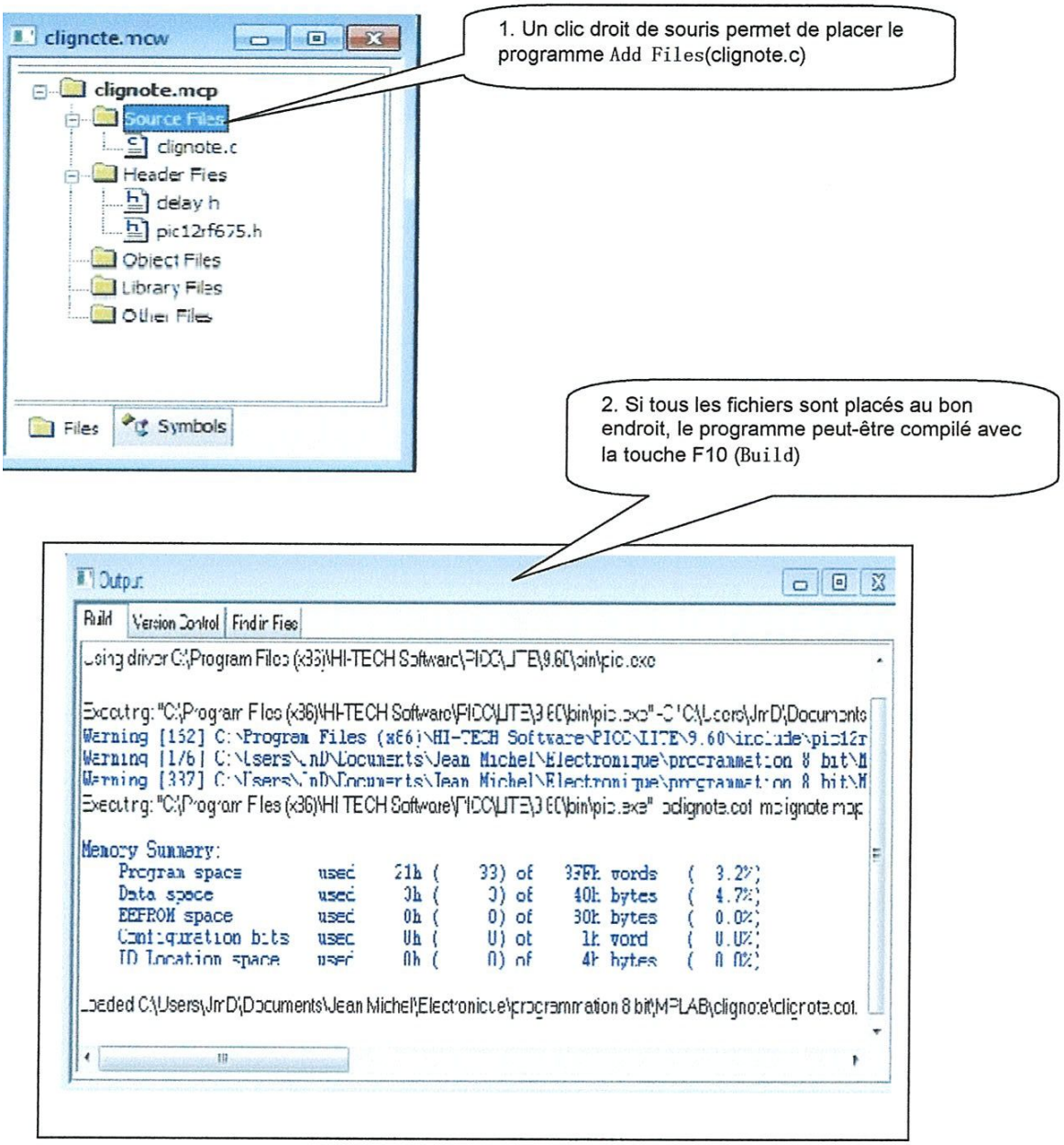
1. Après un appui sur le bouton `Terminer` et le choix dans le menu `Project` de `View` la fenêtre suivante doit apparaitre



2. Le programme peut-être écrit en ouvrant une fenêtre Menu `File` → `New` et enregistrer (clignote.c)



Compiler le programme



Si tout se passe bien le compilateur affiche une fenêtre comme ci-dessus.
Dans le cas contraire il indique des messages d'erreur et ne crée pas le fichier.Hex.
Les erreurs viennent de la syntaxe ou le compilateur ne trouve pas les fichiers d'entête (.h) ou fichier programme (.c) associés.

Bibliographie

Références bibliographiques

- Site de microchip : www.microchip.com
- http://www.technologuepro.com
- http://fabrice.sincere.pagesperso-orange.fr
- Site de mikroElectronika : www.mikroe.com
- Site de Bigonoff (cours extrêmement complet sur les PICs 16F84 et 16F87X) : www.bigonoff.org
- Abcelectronique (ressources en électronique) : www.abcelectronique.com
- http://www.electronique
- http://www.wikipedia.org
- http://www.elektronique.fr/logiciels/mplab.php
- www.sites.google.com/site/mparelectronique/home/projets-a-microcontrolleur-pic
- http://forums.futura-sciences.com/
- projret.PFE\fabrice.sincere.pagesperso-orange.fr\cm_electronique\projet_pic\aidememoire\16F88_ADC\ADC_16F88.htm
- projet_pic/aidememoire/16F876A_bus_I2C/bus_I2C_16F876A.htm#SSP

[16] projet_pic\carte acquisition 7 voies