République Algérienne Démocratique Et Populaire

Ministère de l'Enseignement Supérieur
Et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou
Faculté de Génie Electrique et Informatique
Département Electronique

Projet de Fin d'Etude

Pour L'obtention du Diplôme d'Ingénieur d'Etat en Electronique
Option : Communication

Thème

Etude et réalisation d'une carte de commande PWM, PFM universelle

Présenté par :
GUELLIL Assam

Encadreur :
Mr: LAKHLEF Ahcene

Soutenue devant le jury :

Mr: ABDELLI	Président
Mr: ACHOUR	Examinateur
Mr: MAIFI	Examinateur

Année Universitaire 2009-2010

Remerciement

Travers ce modeste travail, je tiens à remercier vivement mon promoteur MR. LAKHLEF pour ses conseils précieux et pour toutes les commodités et aisances qu'il m'a apportées durant mon étude et réalisation de ce projet.

Mes remerciements les plus vifs s'adressent aussi aux messieurs le président et les membres de jury d'avoir accepté d'examiner et d'évaluer mon travail.

J'exprime également ma gratitude tous ceux qui ont contribué de près ou de loin à réaliser du présent travail.

Table des matières

Introduction générale

Au début du XXIème siècle la technologie a connue une évolution exponentielle, surtout dans Le domaine de l'électronique…

L'être humain ne cesse de recourir à l'électronique qui est en perpétuelle évolution et lui devient de plus en plus serviable. La réalisation d'une unité de commande fait souvent appel à des fonctions d'électronique. Dans la technologie classique, chaque fonction est réalisée en faisant intervenir des circuits complexes et difficiles à mettre en œuvre. Avec l'avènement des circuits programmables comme les microcontrôleurs, les DSP et les FPGA, avec ces circuits les fonctions de l'électronique classique sont devenues plus simples à réaliser et nécessitent moins de temps et d'efforts pour leur concrétisation.

Notre travail consiste à réaliser une carte de commande à base d'un PIC 16F877 qui génère un signal PWM, PFM dont la largeur d'impulsion ou la fréquence sont contrôlés par plusieurs moyens de l'environnement extérieur.

La diversité des signaux générés et celle des moyens de manipulation extérieurs rendent cette carte d'une utilité considérable dans plusieurs applications industrielles, citant l'électronique de puissance et la commande des machines.

Ce document est constitué de quatre chapitres :

- Le premier chapitre intitulé « *Généralités sur la modulation PWM, PFM* » : est consacré à des définitions, la représentation temporelle et fréquentielle de ces signaux.
- Le deuxième chapitre intitulé « *Description de l'unité de commande* » : dans ce chapitre on détaillera le fonctionnement et le rôle de chaque composant constituant de la carte.
- Le troisième chapitre intitulé « *Programme de commande* » : où le programme de commande est expliqué, en montrant son interaction avec l'environnement extérieur.
- Le quatrième chapitre intitulé « *Simulation, réalisation et test* » : dans ce chapitre on mit en œuvre notre travail et on simule l'exécution du programme de commande et aussi des applications dont ont besoin à de tels signaux.
- En fin notre travail se termine par une *conclusion générale*.

Dans ce projet on évoquera deux types de modulation d'impulsion :

- La modulation de largeur d'impulsion (PWM).
- La modulation de fréquence d'impulsion (PFM).

I.1.Définitions :

a) La modulation PWM :

C'est la plus classique. Sa génération revient à des décisions de type tout ou rien. En donnant une impulsion rectangulaire pendant une durée appelée Ton et rien pendant une durée Toff, et la somme de Ton et Toff donne la période du signal T (Figure 1.1).
Les avantages sont la très grande simplicité et le temps de réponse minimal aux perturbations Dans ce type de modulation, c'est la largeur d'impulsion qui varie pour une fréquence donnée.

b) La modulation PFM :

La même technique utilisée pour la PWM toute en gardant la largeur d'impulsion fixe (Ton constant), C'est la fréquence du signal qui varie (Toff variable).

c) La modulation PTM :

C'est une modulation PWM dont la largeur d'impulsion est une fonction du temps ça veut dire, Ton= f(t), et Toff= T- Ton, pour une T fixe.

I.2. Représentation temporelle d'un signal PWM/PFM :

Un signal modulé en PWM/PFM/PTM est défini par sa période T est son rapport cyclique α, tel que α est défini comme suit :

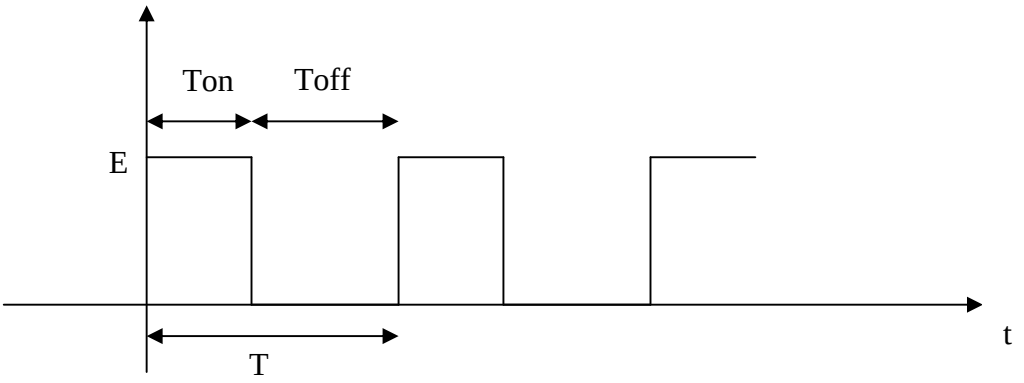


Figure 1.1 : la représentation d'un signal modulé en PWM/PFM.

$$\alpha = \frac{T_{on}}{T}$$

La définition de ce rapport est d'une grande importance, car autour de lui s'effectue tout les calculs de la commande par la PWM/PFM/PTM

I.3. Représentation fréquentielle d'un signal PWM/PFM :

La représentation fréquentielle d'un signal périodique implique son développement en série de Fourrier pour qu'il divulgue son contenu spectral.

Soit $x(t)$ un signal périodique de période T, son développement en série de Fourrier est donné par la formule suivante :

$$x(t) = \sum_{n=-\infty}^{\infty} C_n e^{j\frac{2\pi}{T}nt}, \quad C_n = \frac{1}{T}\int_0^T x(t)e^{-j\frac{2\pi}{T}nt}dt, \; C_0 = \frac{1}{T}\int_0^T x(t)dt$$

C_n : Appelés les coefficients de Fourrier.

C_0 : La valeur moyenne du signal $x(t)$.

Un signal modulé en PWM/PFM a le développement en série de Fourrier suivant :

$$C_n = E\frac{\sin(\pi n\alpha)}{n\pi}e^{-j\pi n\alpha} \quad , \quad C_0 = \alpha E$$

Les amplitudes des spectres $|C_n|$ de $x(t)$ sont représenté dans les figures suivants :

- Pour $\alpha = \frac{1}{4}$ (a), $\alpha = \frac{1}{8}$ (b) :

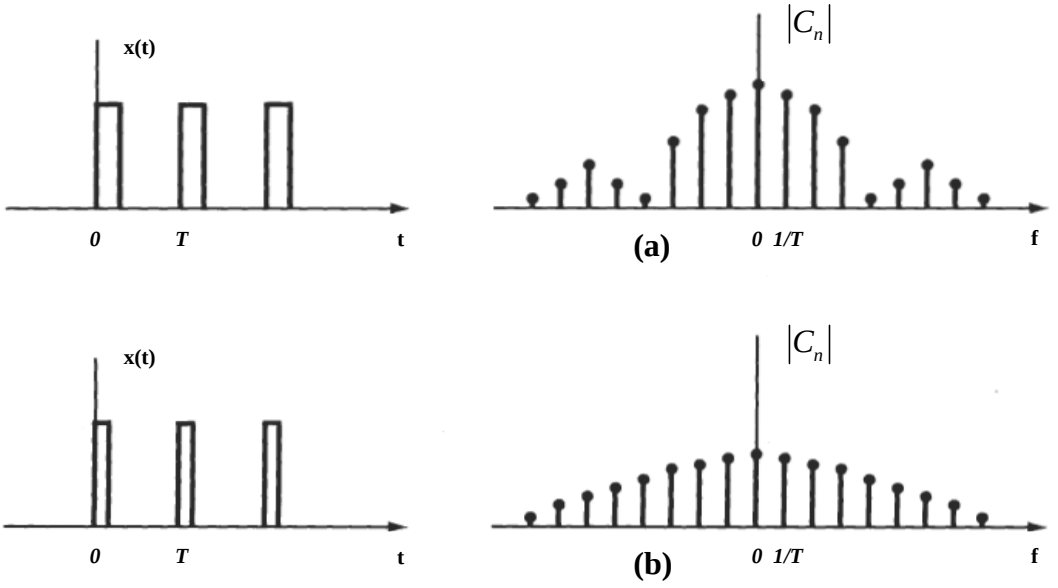


Figure 1.2 : la représentation spectrale d´un signal modulé en PWM/PFM.

Le signal modulé en PMW/PFM contient une composante continue de fréquence zéro $C_0 = \alpha E$, et des harmonies dont la fréquence et le multiple de la fréquence fondamentale $f = 1/T$.

Les harmonies du signal à étudier ne sont pas porteuses d'énergie, mais elles jouent un rôle perturbateur.

I.4. L'effet d'un signal PWM/PFM sur un système de 1er ordre :

Un signal modulé en PWM/PFM doit être appliqué à un système de 1er ou 2ème ordre tant qu'ils se comportent comme un filtre passe-bas pour qu'il puisse filtrer les harmonies inutiles d'ordre supérieur à zéro.

Un système de premier ordre est donné par sa fonction de transfert suivante :

$$G(s) = \frac{K}{1+T.s}$$

K : le gain du système.

T : constante de temps.

Pour mieux clarifier le comportement d'un système de 1er ordre vis-à-vis un signal en PWM/PFM/PTM, on doit mettre en évidence sa réponse fréquentielle.

$$\begin{cases} |G(j\omega)|_{db} = \dfrac{K}{\sqrt{1+(T\omega)^2}} \\ \varphi = -\tan^{-1}\omega T \end{cases}$$

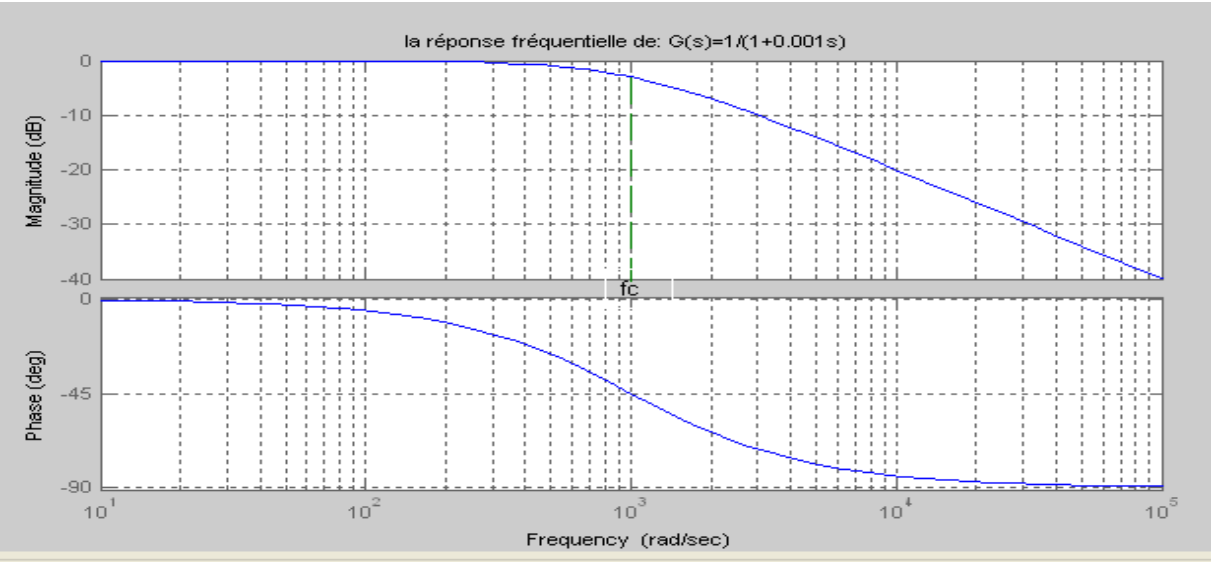


Figure 1.3 : La réponse fréquentielle d'un système de 1er ordre.

On remarque à partir de la réponse fréquentielle (Figure 1.3) que les systèmes de 1er ordre atténuent fortement les harmonies dont la fréquence est supérieure à sa fréquence de coupure f_c .

En exploitant cette propriété pour choisir la fréquence f du signal qui doit être supérieure à f_c , afin d'atténuer les harmonies d'ordre supérieur à zéro, donc que la composante continue ne le subit pas.

$$\boxed{f \succ f_c}$$

On prend à titre d'exemple :

- Un système de fréquence de coupure $f_c = 1KH$.
- Le développement en série de Fourrier d'ordre 1 d'un signal x(t) modulé en PWM pour ($f_1 = 0.5KH \prec f_c$) et ($f_2 = 10KH \succ f_c$).

Pour n=1 : $x(t) = \alpha E + \frac{E}{\pi}\sin(\pi\alpha)e^{j(\omega t - \frac{\pi}{2})} + \frac{E}{\pi}\sin(\pi\alpha)e^{-j(\omega t - \frac{\pi}{2})} = \alpha E + 2\frac{E}{\pi}\sin(\pi\alpha).\sin(\omega t)$

Pour $\alpha = \frac{1}{2}$: $x(t) = \frac{1}{2}E + 2\frac{E}{\pi}\sin(\omega t)$

❖ **Simulation avec Matlab :**

Prenant l'exemple précédant pour E=1/2, ce qui rend : $x(t) = 1 + 1.273\sin(\omega t)$.

On applique ce signal à l'entrée d'un système de 1er ordre ($f_c = 1KHZ$, K=1) et on obtient la réponse y(t) dont la transformée de Laplace y(s) :

$$y(s) = \frac{1}{s}(1 + 1.27.\frac{\omega^2 s}{s^2 + \omega^2})(\frac{1}{1+Ts}) = \frac{1}{s}(\frac{s^2 + 1.27\omega s + \omega^2}{Ts^3 + s^2 + \omega^2 Ts + \omega^2})$$

- Simulation pour $\omega = 2\pi f_1$:

```
>> num1=[0 1 1.27*(2*3.14*500) (2*3.14*500)^2];
>> den1=[0.001 1 0.001*(2*3.14*500)^2 (2*3.14*500)^2];
>> step(num1,den1)
>> ylabel('y(t)')
>> title('la réponse y(t) pour f1<fc')
>> axis([0 0.015 0 1.5])
```

La réponse y(t) pour $f < f_c$ montre l'effet de la première harmonie du signal modulé en PWM sur la sortie.

Donc la perturbation due à cette harmonie est forte.

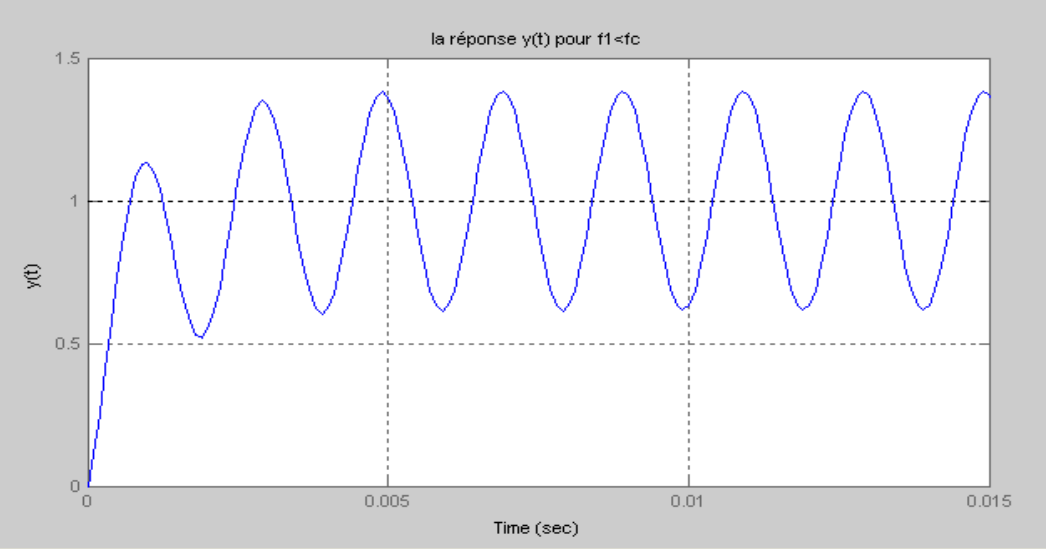


Figure 1.4 : la réponse du système de 1er pour $f < f_c$.

- Simulation pour $\omega = 2\pi f_2$:

```
>> num2=[0 1 1.27*(2*3.14*10000) (2*3.14*10000)^2];
>> den2=[0.001 1 0.001*(2*3.14*10000)^2 (2*3.14*10000)^2];
>> step(num2,den2)
>> ylabel('y(t)')
>> title('la réponse y(t) pour f1>fc')
>> axis([0 0.015 0 1.5])
```

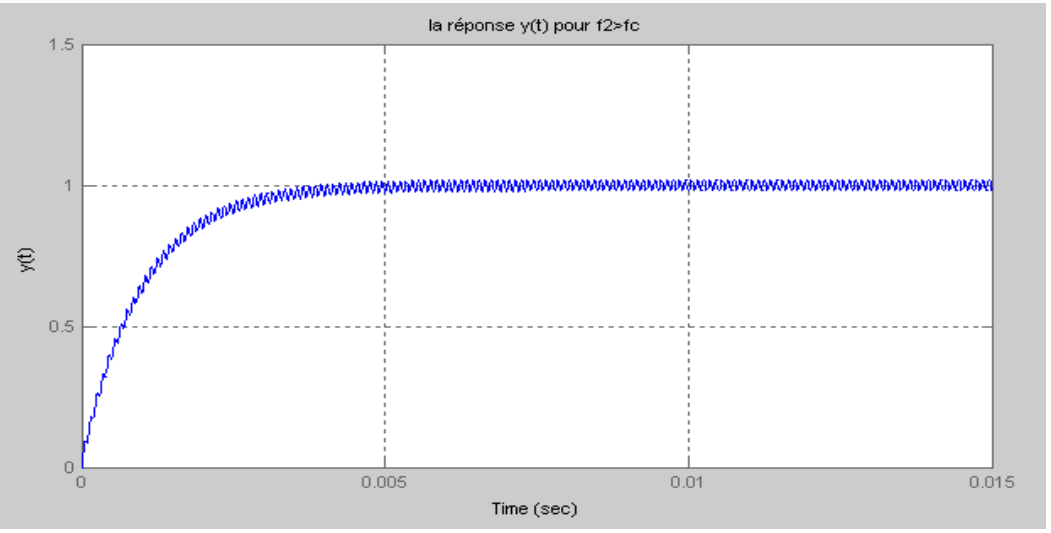


Figure 1.5 : la réponse du système de 1er pour $f > f_c$.

La réponse y(t) pour $f > f_c$ montre l'effet de la première harmonie du signal modulé en PWM

sur la sortie.

Donc la perturbation due à cette harmonie est énormément faible.

Les autres harmonies d'ordre supérieur sont d'amplitudes faibles et de fréquences multiples de la fréquence fondamentale du signal, donc plus loin de la fréquence de coupure du système ce qui rend leur effet pratiquement négligeable.

Si on néglige l'effet de la première harmonie, le système ne voit à son entrée que la composante continue C_0 qu'est un échelon dont l'amplitude est proportionnelle au rapport cyclique α .

La figure suivante montre la similitude entre l'application d'un échelon unitaire et le signal x(t) du développement de 1er ordre du signal modulé en PWM à l'entrée du système de 1er ordre (l'exemple précédent).

- Simulation pour $\omega = 2\pi f_2$ et un échelon unitaire :

```
>> hold on
>> num3=[0 1];
>> den3=[0.001 1];
>> step(num3,den3)
>> title('la réponse y(t) pour f2>fc et lechelon unitaire')
```

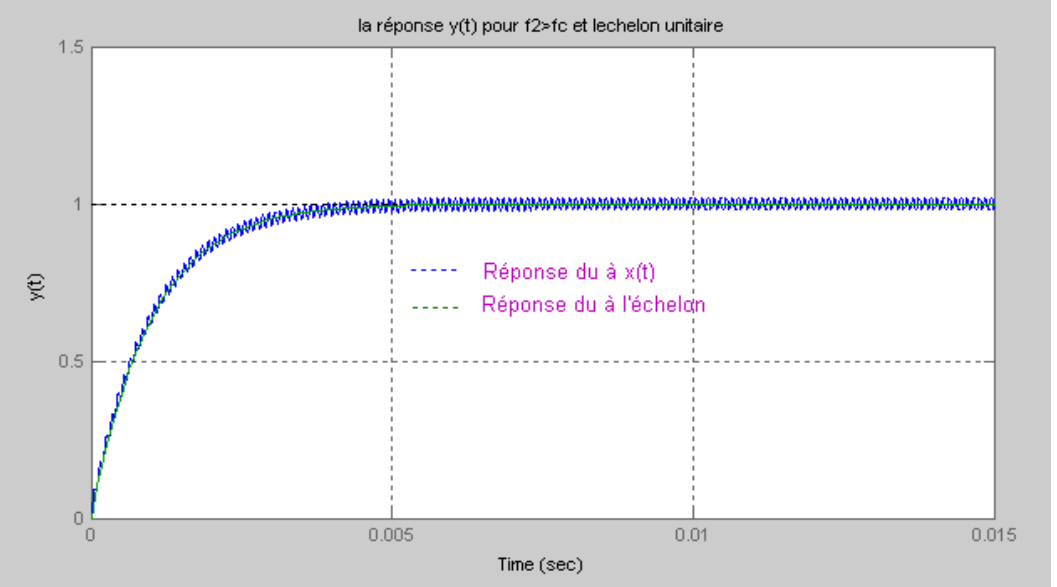


Figure 1.6 : la réponse du système de 1er pour $f > f_c$ et pour un échelon unitaire.

On conclut de cette étude qu'on peut commander un système passe-bas par un signal modulé en PWM /PFM au lieu d'utiliser un échelon.

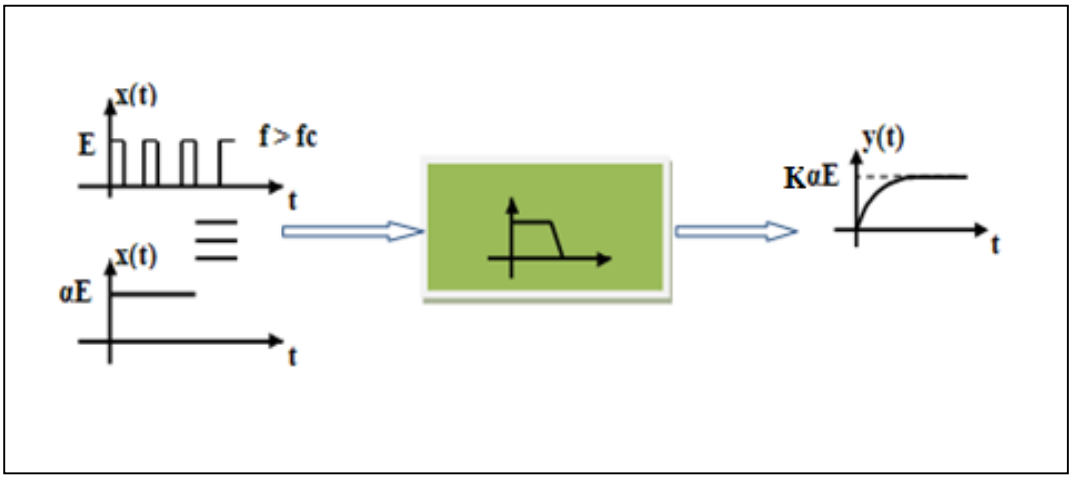


Figure 1.7 : l'équivalence entre l'utilisation d'un signal PWM et un échelon.

Dans la plupart des cas on est contraint de joindre au système à commander un filtre passe – bas pour filtrer les harmonies.

I.5. L'utilisation du signal PWM/PFM pour la régulation :

Souvent, on se trouve dans des cas de régulation complexe, où on à besoin au traitement numérique des signaux de mesure pour générer la commande correspondante ,en utilisant des circuits intégrés programmable (microcontrôleur, DSP, FPGA…) selon la complexité de l'algorithme à traiter, ces derniers sont connus par leurs principe de fonctionnement basé sur la commutation (0/1 ou 0/5V), sont plus aptes à générer au même temps des signaux PWM/PFM, on appelle ce type de régulation : la régulation modulée.

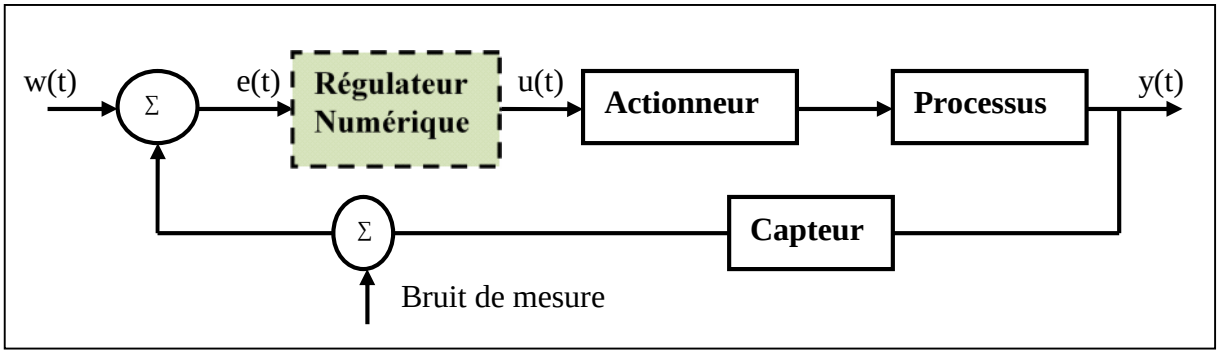


Figure 1.8 : Shéma fonctionnelle d'un système de régulation numérique.

Cette propriété nous donne l'alternative de programmer la durée d'un niveau logique d'une sortie du circuit calculateur et par conséquent la programmation du rapport cyclique α , ce qui nous permet d'obtenir l'amplitude désirée après le filtrage des harmonies.

I.5.1.Principe de fonctionnement d'un régulateur classique :

Un régulateur numérique classique est menu généralement d'un convertisseur analogique/numérique (A/N) pour convertir le signal à traiter à un signal numérique .plus un processeur de traitement qui s'occupe de l'exécution d'un algorithme implémenté dedans en exploitant les octets du à la conversion A/N comme argument d'entrée et en donnant à la sotie des octets qui résultent à l'exécution de cette algorithme. Plus un convertisseur numérique/analogique pour convertir le signal numérique traité à un signal analogique utilisable dans la commande d'un processus (Figure 1.9). Ces éléments peut être intégrés ou non selon la technologie de construction du calculateur.

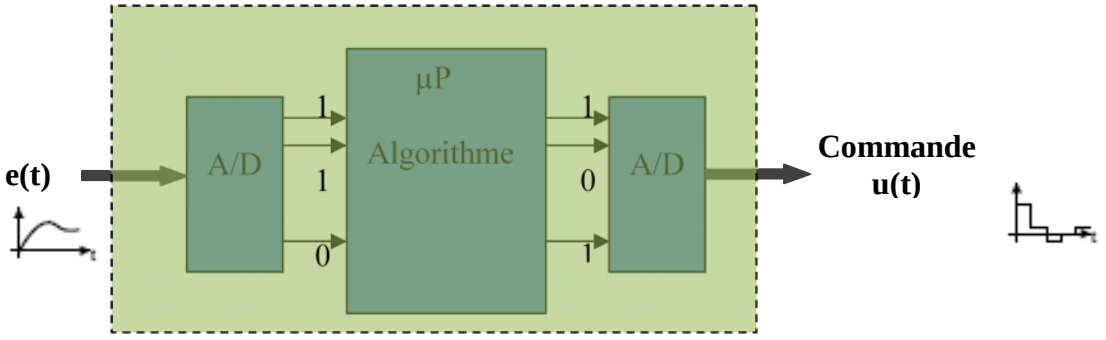


Figure1.9 : Régulateur numérique classique.

Le temps de traitement T_T du signal e(t) jusqu'à l'apparition de la commande correspondante u(t) est la somme du temps de conversion A/N plus le temps d'exécution de l'algorithme plus le temps de conversion N/A.

$$T_T = T_{CAN} + T_e + T_{CNA}$$

I.5.2. Principe de fonctionnement d'un régulateur PWM/PFM :

Un régulateur numérique classique est menu généralement d'un convertisseur analogique/numérique (A/N) pour convertir le signal à traiter à un signal numérique .plus un processeur de traitement qui s'occupe de l'exécution d'un algorithme implémenté dedans qui exploite les octets du à la conversion A/N et manipule le paramètre temps (timer du processeur) par la temporisation des niveaux logiques (haut et bas) d'un sortie afin d'obtenir un signal PWM de rapport cyclique désiré.

A la sotie du filtre passe-bas on aura une commande :

$$u(t) = \alpha(t).E \qquad \alpha(t) \in [0,1]$$

E : l'amplitude du signal PWM/PFM.

$$\alpha(t)=\frac{T_{on}}{T_{on}+T_{off}}$$

T_{on} : La période pendant laquelle la sortie du calculateur est au niveau haut.

T_{off} : La période pendant laquelle la sortie du calculateur est au niveau bas

Ces deux paramètres T_{on} et T_{off} sont les paramètres de sortie de l'algorithme.

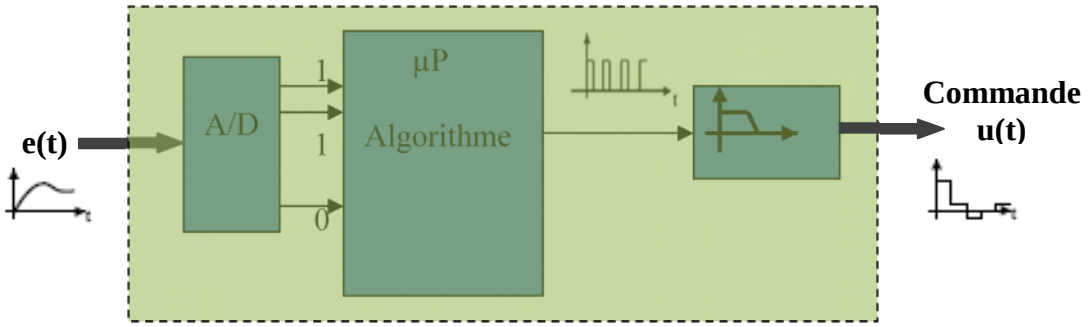


Figure1.10 : Régulateur numérique PWM.

Le temps de traitement T_T totale du signal e(t) jusqu'à l'apparition de la commande correspondante u(t) est la somme du temps de conversion A/N plus le temps de l'exécution de l'algorithme.

$$\boxed{T_T = T_{CAN} + T_t}$$

La composante continue du signal PWM ne subit aucun retard (déphasage) au niveau du filtre (Voir La Figure1.3).

I.5.3. Avantages d'un régulateur PWM/PFM :

- L'utilisation d'une seule sortie du processeur de traitement au lieu de 8, 16… dans le cas d'un régulateur numérique classique.
- Temps de traitement plus court par rapport à celui d'un régulateur numérique classique (Pas de conversion numérique / analogique).

II.1.Introduction :

Notre unité de commande sert à commander tous les dispositifs hacheur à deux ou quatre quadrant ou comme régulateur dans des systèmes de régulation par la modulation de largeur d'impulsion PWM et par modulation de fréquence PFM. Elle constituée de deux partie essentielles, partie commande qui comporte le microcontrôleur comme élément de base qui s'occupe de la réalisation des différentes fonctions (conversion, comptage, variation de la fréquence et de largeur d'impulsion), et l'autre partie c'est la partie d'isolation et amplification qui se présente comme interface intermédiaire entre la partie commande et la partie opérative, elle est constituée d'un photocoupleur suivi d'un amplificateur de puissanceL293B.

Pour permettre à l'opérateur de suivre l'évolution des données en temps réel, l'unité est équipée d'un afficheur LCD et d'un clavier plus un connecteur D9M pour connecter avec le PC via le port série.

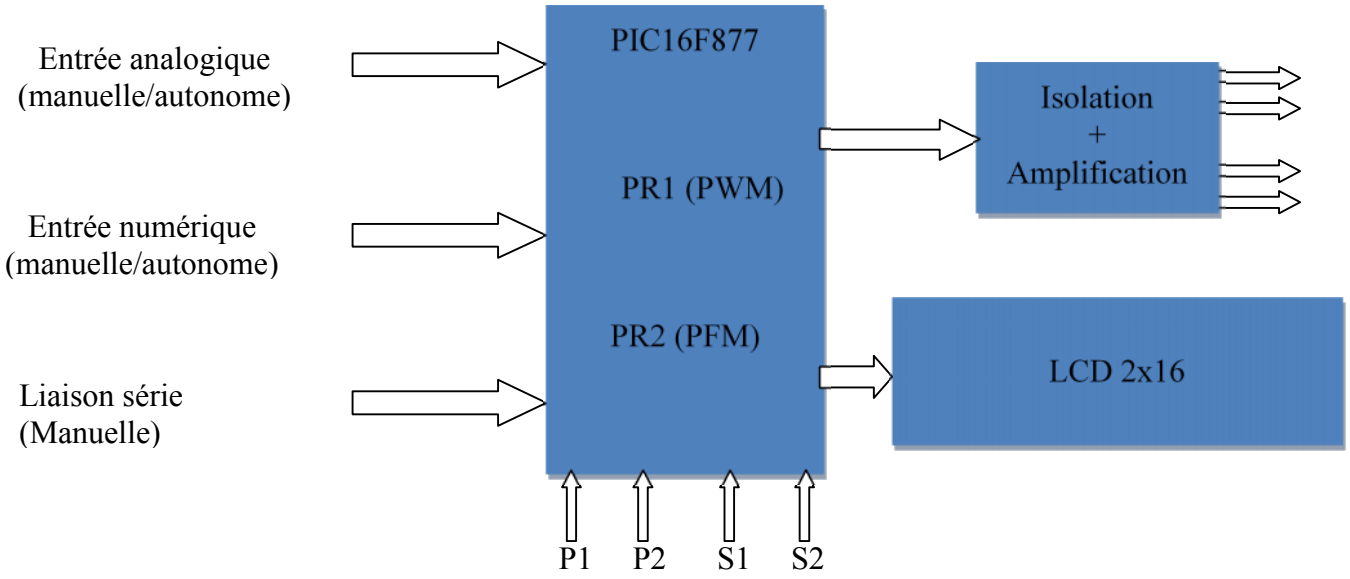


Figure 2.1 : schéma fonctionnelle de la carte.

II.2.Principe de fonctionnement de la carte :

Le microcontrôleur exploite l'une des trois entrées (analogique, numérique, PC) pour exécuter l'un des programmes (PR1, PR2), implémenter dedans en utilisant l'interrupteur (S1 te S2) pour choisir l'entrée, et l'interrupteur (P2) pour choisir le programme à exécuter.

PR1 : est un programme qui génère un signal PWM.

PR2 : est un programme qui génère un signal PFM.

Donc le choix s'effectue sur le programme à exécuter et sur la source des données.
Il y a un pin qui est utilisé comme sortie du signal, et l'afficheur LCD 2x16 est connecté sur le port *B* pour afficher les paramètres (fréquence, rapport cyclique…) du signal.

II.3. les composants principaux de la carte et leurs rôles :

Cette carte compris principalement les éléments suivants :

- Microcontrôleur 16F877.
- Circuit intégré Max232.
- Afficheur 2x16 caractères.
- Circuit d'isolation photo-galvanique PC847.
- Potentiomètres.
- Connecteurs.
- Interrupteurs.
- Connecteur DB9-mâle.
- DIP-8.

II.3.1. Le Microcontrôleur 16F877:

II.3.1.1.Généralités :

Un microcontrôleur est un composant électronique autonome doté d'un microprocesseur, mémoire RAM, mémoire permanente, interface d'E/S parallèles et série, interface d'E/S analogiques, timer pour gérer le temps et d'autre modules plus au moins sophistiqués selon la taille des microcontrôleurs.

Il est généralement moins puissant qu'un microprocesseur en terme de rapidité ou taille mémoire, il se contente le plus souvent d'un bus de 8 ou 16 bits. Ceci en fait un composant très bon marché parfaitement adapté pour piloter les applications embarquées dans de nombreux domaines d'application.

Les microcontrôleurs, quelque soient leurs constructeurs, ont des architecture très similaires et sont constitués de modules fondamentaux assurant les même fonctions :
UAL, ports d'E/S, interfaces de communication série, interface d'E/S analogiques, timer et horloge temps réels. On peut dire que seul le langage de programmation constitue la différence majeure entre deux microcontrôleurs venant de deux constructeurs différents.

II.3 .1.2. Présentation du microcontrôleur PIC 16F877 :

II.3.1.2.1. Eléments constructifs du PIC 16F877 :

Le PIC 16F877 est produit par le constructeur Microchip, il fait parti de la famille des pic Middle-range (les mots de la mémoire programme sont codés sur 14 bits) et possède un jeu d'instruction réduit (35 instructions), son boitier et de type PDIP 40.

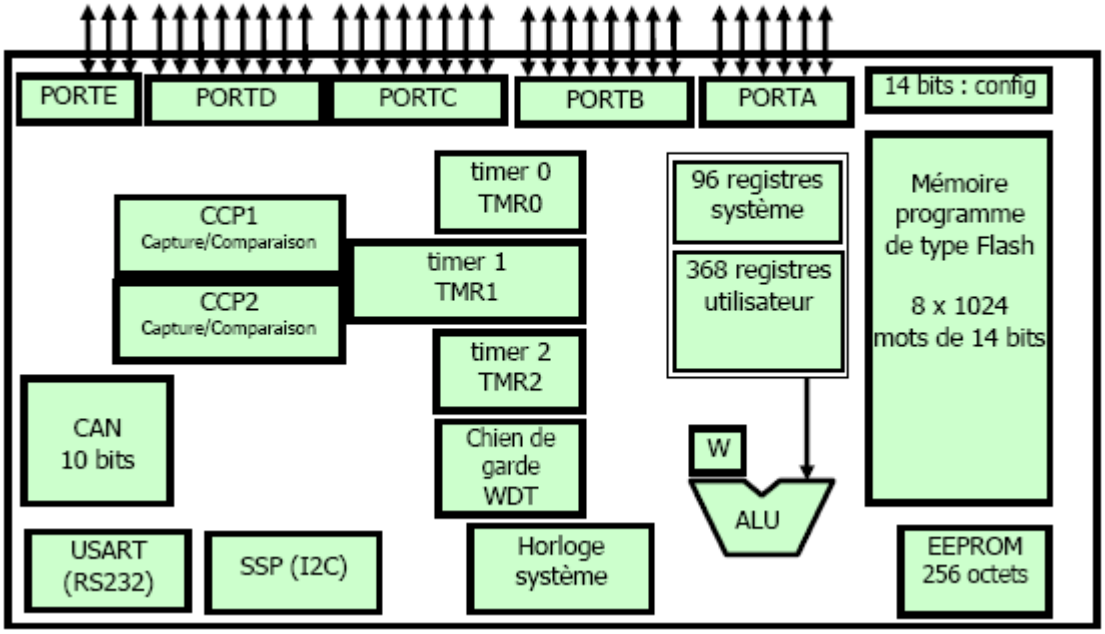


Figure 2.2 : Eléments constructif du PIC 16F877.

Sa mémoire programme est composée de trois type de mémoire (Figure 2.2):

- Une mémoire programmable (Flash), de 8k mots de 14 bits.
- Une mémoire EEPROM de 256 octets.
- Une mémoire vive (RAM) de 386 octets.

Il possède 40 broches, 33 d'entre elles sont des entées/sorties (PORTA, PORTB, PORTC, PORTD, PORTE), dont 8 peuvent être utilisées comme entées analogiques (RA0, RA1, RA2, RA3, R5, RE0, RE1, RE2) (Figure 2.3).

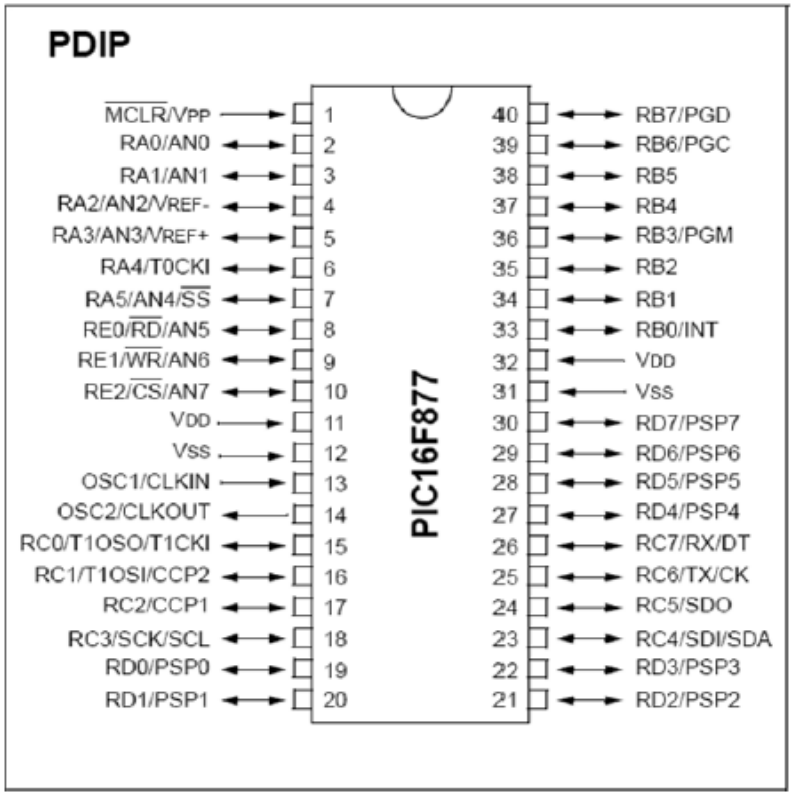


Figure 2.3 Brochage du PIC 16F877.

Il possède également un convertisseur analogique/numérique. Et deux modules de transmission série USART (Universal Synchronous Asynchronous Receiver Transmitter), et MSSP (Master synchronous Serial Port). Et trois timer pour gérer le temps, et des entrées qui provoquent des interruptions (RB0, RB4 à RB7).

Tout les PIC sont accompagnés d'un circuit d'initialisation et un circuit d'horloge.

a) circuit d'initialisation :

Le circuit de reset externe est nécessaire lors de la mise sous tension, en effet, à ce moment là, le microcontrôleur n'est pas prêt à exécuter des instructions instantanément.

L'entrée MCLR (Broche1) permet d'initialiser le microcontrôleur après la mise sous tension. Une réinitialisation est correctement effectuée si l'entrée MCLR est mise au niveau bas.

Le circuit RC empêche la tension sur la broche 1 d'attendre la valeur de la tension d'alimentation pendant un temps correspondant au temps de charge de la capacité, afin que le circuit interne de reset automatique détecte la tension d'alimentation. Soit le temps de charge

τ=1ms. Le bouton poussoir permet le reset du pic pendant son fonctionnement. En fixant la valeur de la capacité C à 0.1µF, on aura R=10KΩ.

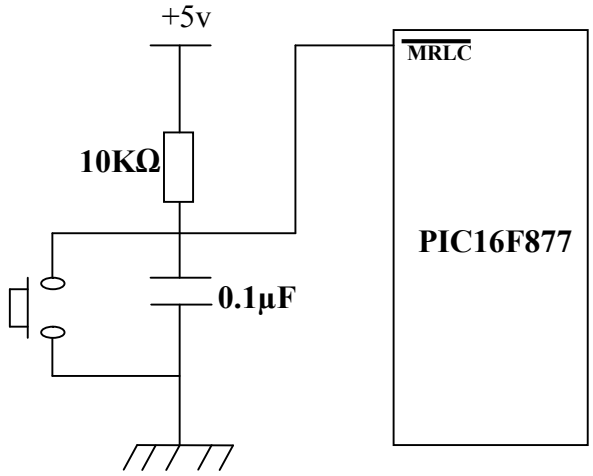


Figure 2.4 Circuit de reset du microcontrôleur 16F877.

b) Circuit d'horloge :

Un signal d'horloge est nécessaire pour piloter le circuit de contrôle et gérer les séquences du microcontrôleur. La fréquence de ce signal subit une division par quatre, fournissant ainsi un signal d'horloge interne, c'est cette base de temps qui est utilisée pour donner le temps d'un cycle de T est donné par : 4/Fosc.

Dans cette application, nous avons utilisé un quartz de 20 MHZ, nous aurons un donc T=0.2µs, ce qui donne 5 millions cycles par seconde.

Le PIC16F877 exécute pratiquement (sauf les sauts) une instruction par le cycle, ce qui nous donne une puissance de l'ordre de 5MIPS (5 Million d'instructions par seconde). Comme la montre la figure 2.5, Le quartz est branché sur les broches OSC1/CLKIN et OSC2/CLKOUT, avec de capacité de 15pF recommander pour la stabilité de l'oscillateur.

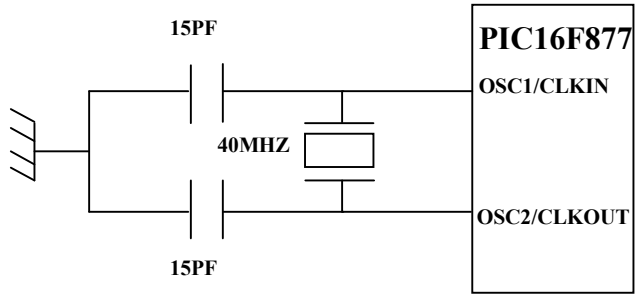


Figure 2.5 Circuit d'horloge du PIC 16F877.

c) La conversion analogique numérique dans le PIC 16F877 :

c.1) Le module de conversion analogique/numérique du PIC16F877 :

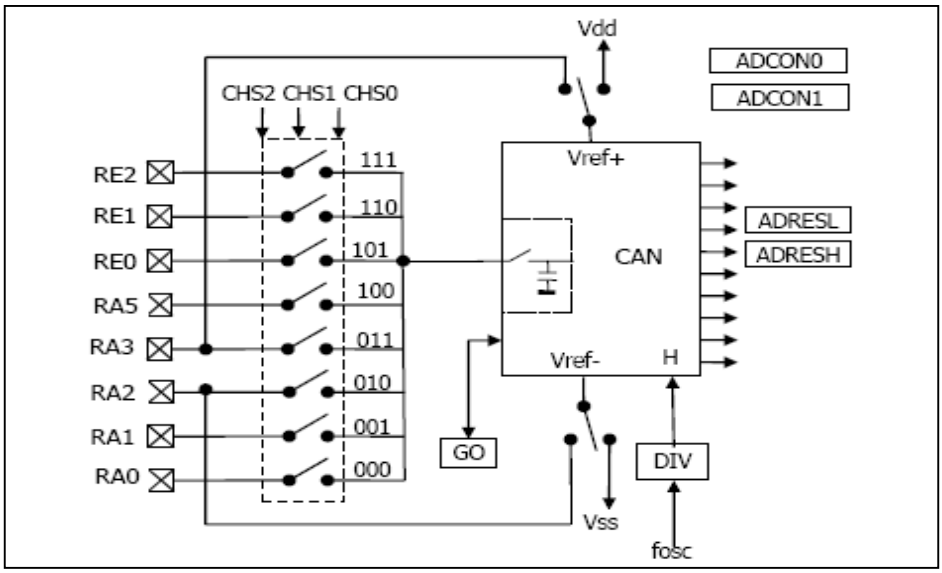


Figure 2.6 : Module de conversion analogique/numérique du PIC 16F877.

Ce module est constitué d'un convertisseur Analogique Numérique 10 bits dont l'entrée Analogique peut être contactée sur l'une des 8 entrées analogique externe. On dit qu'on a un CAN à 8 canaux. Les entrées analogiques doivent être configurées en entrée à l'aide des registres TRISA et/ou TRISE. L'échantillonneur bloqueur est intégré, il est constitué d'un interrupteur d'échantillonnage et d'une capacité de blocage de 120pF.

Les tensions de références permettant de fixer la dynamique du convertisseur. Elles peuvent être choisies parmi Vdd, Vss, Vr+, Vr-.

c.2) control du module de conversion analogique/numérique :

Le control du module se fait par les deux registres ADCON0, ADCON1.

- **ADCCON0 :**

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	--------	ADON

ADCS1:ADCS1 : Choix de l'horloge de conversion donc du temps de conversion.

00 : Fosc/2

01 : Fosc/8

10 : Fosc/32

11 : Oscillateur RC dédié au CAN.

CHS2 :CHS0 : choix de l'entrée analogique.

000 = Channel 0, (RA0)

001 = Channel 1, (RA1)

010 = Channel 2, (RA2)

011 = Channel 3, (RA3)

100 = Channel 4, (RA5)

101 = Channel 5, (RE0)

110 = Channel 6, (RE1)

111 = Channel 7, (RE2)

GO/DONE : une conversion démarre quand on place ce bit à 1. A la fin de la conversion, il est remis automatiquement à zéro.

ADON : Ce bit permet de mettre le module AN en service.

- **ADCON1 :**

ADFM	------	------	------	PCFG3	PCFG2	PCFG1	PCFG0

ADFM : justification à droite ou à gauche du résultat dans les registre ADRESH et ADRESL.

	ADRESH	ADRESL
1 : justifié à droite	000000XX	XXXXXXXX.
0 : justifié à gauche	XXXXXXXX	XX000000.

PCFG3:PCFG0 : configuration des E/S et des tensions de références. Les 5 broches de PORTA et les 3 de PORTE peuvent être configurés soit en E/S digitales, soit en entrées analogiques. RA2 et RA3 peuvent aussi être configurées en entrée de référence.

c.3) Déroulement d'une Conversion :

Le PIC dispose d'un échantillonneur bloqueur intégré constitué d'un interrupteur S, d'une capacité de maintien C=120 pF et d'un convertisseur Analogique numérique 10 bits. Pendant la conversion, la tension Ve à l'entrée du convertisseur A/N doit être maintenue constante. Au départ il faut commencer par faire l'acquisition du signal en fermant l'interrupteur S, ceci se fait à l'aide du registre ADCON0, soit au moment de la validation du module par le bit ADON soit après un changement de canal si ADON est déjà positionné. Après la fin de l'acquisition, on peut démarrer une conversion en positionnant le bit GO_DONE, l'interrupteur S s'ouvre

pour assurer le blocage de la tension. La conversion commence, elle est réalisée en 12 TAD, à la fin, le bit GO_DONE repasse à 0, le drapeau ADIF passe à 1 et le résultat est chargé dans les registres ADRESL et ADRESH. Le module met 2 TAD supplémentaires pour fermer L'interrupteur S ce qui démarre une nouvelle phase d'acquisition pendant laquelle la tension Ve rejoint la tension analogique d'entrée Va. Le temps d'acquisition dépend de la constante de temps RC, R étant la somme des résistances entre le module de conversion et la source de la tension analogique. Après la fin de l'acquisition, on peut démarrer une nouvelle conversion et ainsi de suite.

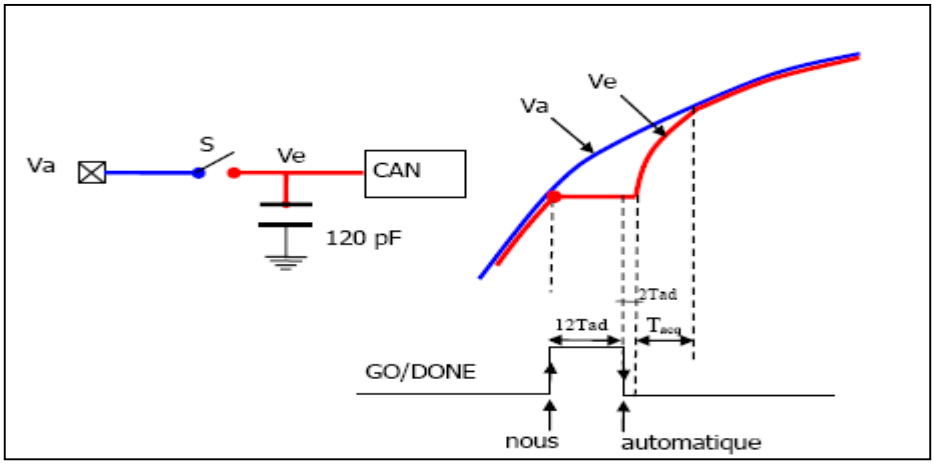


Figure 2.7: déroulement de la conversion dans le convertisseur A/N du PIC 16F877.

c.4) Temps de conversion :

Le temps de conversion est égal à 12 TAD.

TAD est le temps de conversion d'un bit, il dépend de la fréquence du quartz et du prédiviseur (div) choisi : TAD = div x 1/fosc. Le choix de div doit être ajusté pour que TAD soit ≥ à 1,6 µs.

\Quartz Div\	20Mhz	5Mhz	4Mhz	2Mhz
2	0.1µs	0.4µs	0.5µs	1µs
8	0.4µs	1.6µs	2µs	4µs
32	1.6µs	6.4µs	8µs	16µs

Tableau 2.1 : Temps de conversion d'un bit TAD (les cases rouges sont hors la plage d'utilisation).

Avec un quartz de 4 MHz, il faut choisir div=8 ce qui donne TAD = 2 µs soit un temps de conversion : TCONV = 24 µs.

c.4) Temps d'acquisition :

Temps d'acquisition = TACQ = Tc + CT +2 µs.

Tc : temps de charge du condenseur = (Ric+Rss+Rs) C Ln(2047).

Ric = Résistance d'interconnexions, elle est inférieure à 1k.

Rss = Résistance de l'interrupteur S (Sampling Switch), elle dépend de la tension d'alimentation Vdd. Elle est égale à 7kΩ pour Vdd=5V.

Rs : Résistance interne de la source du signal analogique. Microchip recommande de ne pas dépasser 10 kΩ.

C : Capacité de blocage = 120 pF CT : Coefficient de température = (Tp -25°C) 0.05 µs/°C.

Tp = Température Processeur, voisine de 45°C en temps normal

- ✓ **Exemple :**

Ric = 1k, Rss = 7k, Rs = 2k, Tp = 45 °C :

Tc = 10k x 120pF x Ln (2047) = 9 µs.

CT = 20 x 0.05 µs = 1 µs.

TACQ = 2 + 9 + 1 µs = 12 µs.

c.5) Fréquence d'échantillonnage :

Si on veut échantillonner un signal variable, La période d'échantillonnage Te doit être supérieur ou égale à Temin = TCONV + 2 Tad + TACQ.

Avec l'exemple précité, on aura la période d'échantillonnage minimale :

Temin = 24 + 2 + 12 = 40 µs

La fréquence d'échantillonnage max est donc femax = 1/Temin = 25 kHz.

Si on tient compte de la règle de Shannon (fe ≥2 fmax), on constate que l'on peut échantillonner des signaux dont la fréquence ne dépasse pas 12 KHz.

c.6) Valeur numérique obtenue :

Quelle est la relation entre la tension analogique convertie et le nombre N recueilli dans le registre ADRES ?

Si on note : Q = pas de quantification = (Vref+ - Vref-)/1024 (configuration 10 bits).

Q = pas de quantification = (Vref+ - Vref-)/256 (configuration 8 bits).

Va = tension analogique à convertir.

N = valeur numérique obtenue.

N = valeur entière de (Va – Vref-) / Q.

Avec Vref- = masse, on obtient :

$$N = Int (Va / Q)$$

- ✓ **exemple :**

Vref+ = Vdd = 5V, Vref- = 0, Vin = 5 V, Q = 5V/256 = 0,01953125 V.

N = 5V / 0,0048828125 = 256.

Donc, les valeurs qu'on peut lire dans le registre ADRES sont de 0X00 à 0XFF.

- **N.B :**

La programmation en C nous facilite la configuration du module de conversion analogique/ numérique. Il ya des fonctions prédéfinies qui nous permettent de configurer les registre et de choisir le type de conversion à utiliser (10 bits ou 8 bits).

d) L'USART :

L'USART (Universal Synchronous Asynchronous Receiver Transmitter) est l'un des deux modules de communication série dont dispose le PIC 16F877. L'USART peut être configuré comme système de communication asynchrone full duplex ou comme système synchrone half duplex. La communication se fait sur les deux broches RC6/TX et RC7/RX qui doivent être configurés toutes les deux en ENTREE par TRISC.

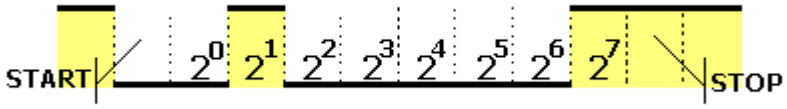
d.1) Mode Asynchrone :

Si on place le bit SYNC du registre TXSTAT à 0, l'USART fonctionne dans le mode asynchrone standard, 10 (ou 11) bits sont transmis ou reçus dans l'ordre ci-dessous :

- 1 bit de START (toujours 0).
- 8 ou 9 bits de donnée (LSB d'abord).
- 1 bits de STOP (toujours 1).

- **Exemple :**

Transmission du code **$82** avec 1 bit de stop, sans bit de parité :**$82** donne **% 1000 0010.**



- La transmission se fait sur la broche RC6/TX et la réception sur la broche RC7/RX.
- La configuration et le control du port se fait par les registres TXSTA et RCSTA.
- La vitesse de communication est fixée par le registre SPBRG et le bit TXSTA.BRGH.
- La parité n'est pas gérée d'une façon matérielle, elle peut être gérée par soft si son utilisation est nécessaire.
- L'accès au port en lecture ou écriture se fait par les registres tampon RCREG et TXREG. La transmission et la réception se font par deux registres à décalage, un pour la transmission (TSR) et un pour la réception (RSR). L'accès au registres tampon peut se faire alors que les registre à décalage sont en train de transmettre/recevoir une donnée.

-Les drapeaux PIR1.RCIF et PIR1.TXIF sont très utiles pour gérer la lecture/écriture dans le port. RCIF est positionné quand le port a terminé une réception et TXIF est positionné quand le buffer de transmission TXREG est vide.

d.2) Le port en transmission :

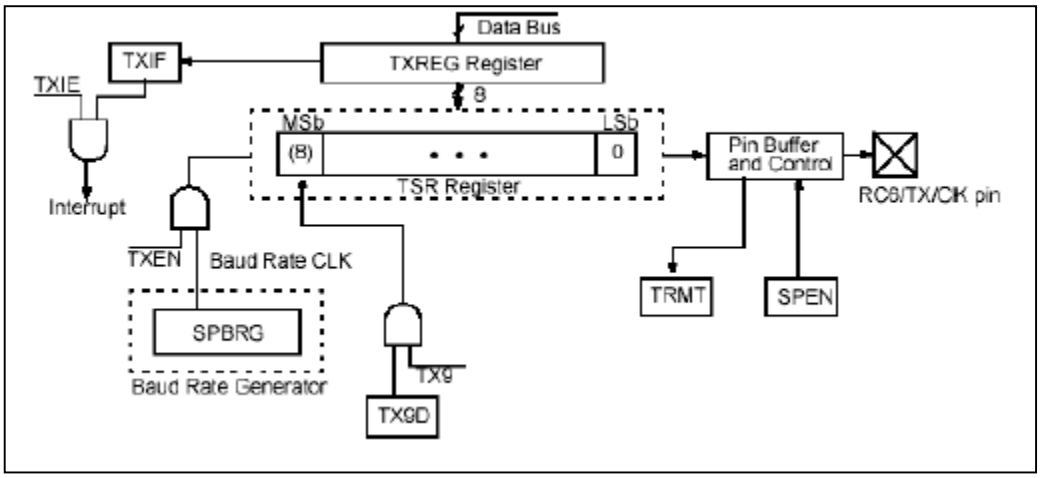


Figure 2.8 : Module de transmission/réception du PIC 16F877 en mode de transmission.

Le contrôle de la transmission se fait par le registre TXSTA :

CSRS	TX9	TXEN	SYNC	-------	BRGH	TRMT	TX9D

CSRC : non utilisé en mode asynchrone.

TX9 et TX9D : Pour utiliser le mode 9 bits il faut positionner le bit TX9. Le 9ème bit doit être écrit dans TX9D avant d'écrire les 8 bits de données dans TXREG.

TXEN : permet de valider ou interdire la transmission.

SYNC : 0 → mode asynchrone, 1 → mode synchrone.

BRGH : sélectionne le mode haut débit du générateur de baud rate.

TRMT : Indicateur de l'activité du registre à décalage de transmission TSR,
1 → TSR libre, 0 → TSR en activité.

d.2.1) Déroulement de la transmission :

Quand on écrit un octet D dans le registre TXREG, le drapeau PIR1.TXIF passe à 0, ensuite, deux situations sont possibles :

- Le registre de transmission TSR n'est pas occupé, alors la donnée D est transférée immédiatement dans TSR qui commence sa transmission bit bar bit. Le drapeau TXIF repasse à 1 pour nous dire que nous pouvons de nouveau écrire dans TXREG.
- Le registre de transmission TSR est occupé à transmettre un octet qui lui été donné auparavant. La donnée D attend dans TXREG, et le drapeau TXIF reste à 0 jusqu'à ce que TSR termine de transmettre l'octet précédent. La donnée D est alors transférée dans TSR qui commence sa transmission bit bar bit. Le drapeau TXIF repasse à 1 pour nous dire que nous pouvons de nouveau écrire dans TXREG.

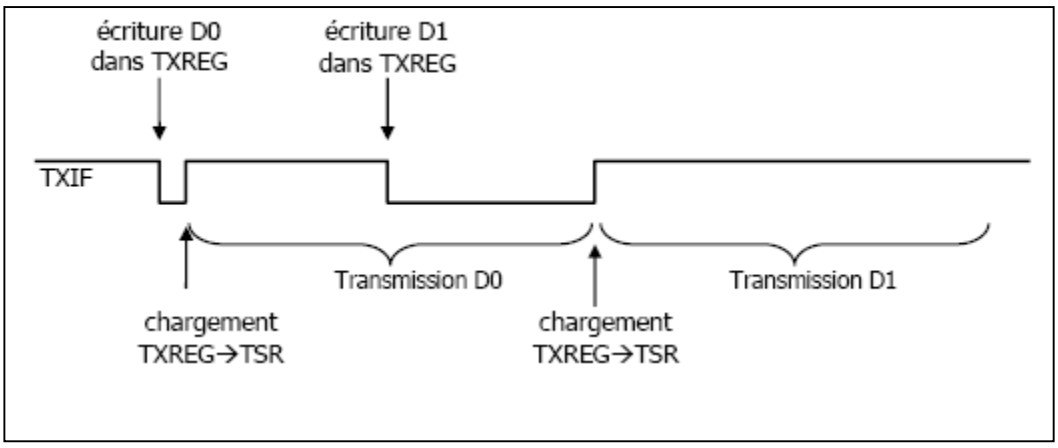


Figure 2.9 : illustration de la transmission de deux octet successifs D0 et D1.

- A la mise sous tension, tous les drapeaux sont à zéro y compris TXIF. Dès qu'on valide la transmission en positionnant le bit TXEN, le drapeau TXIF passe automatiquement à 1, Sauf si on écrit d'abord un octet dans TXREG et on valide ensuite le bit TXEN, dans ce cas, dès la validation de TXEN, le contenu de TXREG est transféré dans TSR qui commence sa transmission et le drapeau TXIF passe à 1.
- Comme on vient de le voir, le drapeau TXIF est géré automatiquement, on ne peut pas le modifier directement par programme.

d.3) Le port en réception :

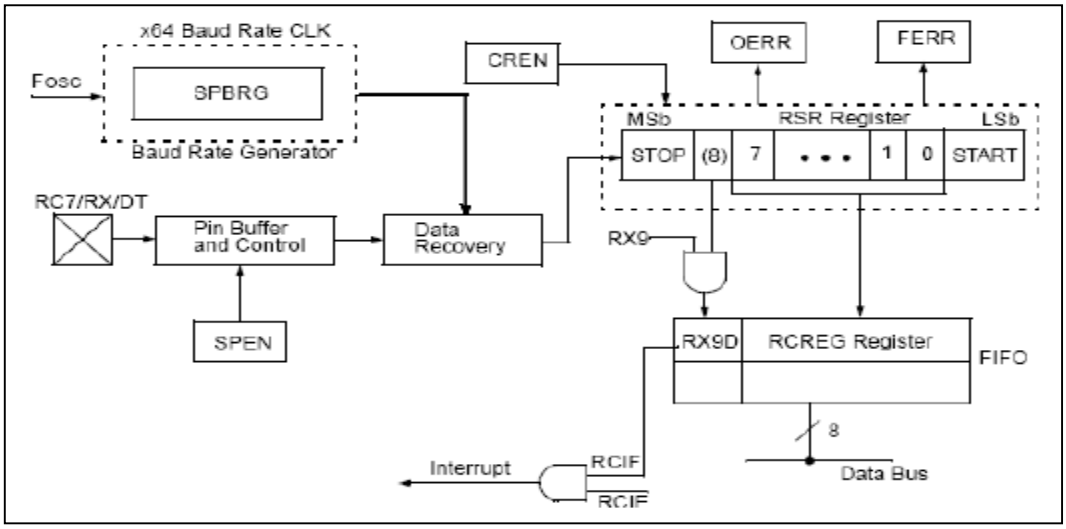


Figure 2.10 : Module de transmission/réception du PIC 16F877 en mode de réception.

Le contrôle de la réception se fait par le registre RCSTA

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

SPEN : Validation du port série (1 = validé, 0 = Inhibé).

RX9 : Validation du mode 9 bits (1 → mode 9 bits, 0 →mode 8 bits).

SREN : validation de réception d'u seul octet (non utilisé en mode asynchrone).

CREN : Validation du mode réception continue (1 →validé, 0 →inhibé).

ADDEN : validation du mode détection d'adresse en mode 9 bits, utilisé en mode multiprocesseurs (1 → validé, 0 →Inhibé).

FERR : Erreur de synchronisation, lecture seule.

OERR : Erreur débordement du buffer de réception, lecture seule.

RX9D : En mode 9 bits, le 9ème bit est reçu ici

d.3.1) Déroulement de la réception :

-La réception est validée par le bit CREN.

-La réception d'un octet démarre à la réception du START bit qui commence toujours par une transition 1 → 0.

- A la réception du stop bit le contenu du registre à décalage de réception RSR est recopié dans le registre tampon de réception RCREG. Le drapeau RCIF (PIR1.5) est positionné, l'interruption associée est déclenchée si elle est validée. Le drapeau RCIF est remis à zéro automatiquement au moment de la lecture dans RCREG.

- Le registre RCREG est un registre double (FIFO à 2 positions). On peut donc avoir 2 octets en attente dans ce registre et être en train de recevoir un 3ème dans le registre à décalage RSR. A la fin de la réception du 3ème octet, si RCREG est toujours plein, alors le dernier octet reçu est perdu et le bit OERR (Overrun ERRor bit) est positionné ce qui provoque l'arrêt des transferts du registre à décalage RSR vers le buffer RCREG. Pour reprendre la réception il faut réinitialiser le module de réception en mettant à 0 puis à 1 le bit CREN (□).

- Le drapeau PIR1.RCIF ne passe à 0 que quand la pile RCREG est vide

- Si on reçoit un 0 à la position du STOP bit qui doit être toujours à 1, alors le bit FERR (Framing ERRor) est positionné. Ce bit ainsi que le 9ème bit (si utilisé) sont bufférisés, Ils doivent être lu dans le registre RCSTA avant la lecture du registre RCREG.

d.4) La vitesse de communication:

La vitesse de communication est déterminée par le générateur de rythme BRG (Baud Rate Generator) qui est dédié au port série. La vitesse de communication est définie à l'aide du registre de control SPBRG et du bit BRGH, qui quadruple la vitesse quand il est positionné. Il préférable d'utiliser le mode haute vitesse (BRGH=1) car permet d'obtenir une meilleur précision sur la fréquence.

$$vitesse = \frac{4^{BRGH} \times Fosc}{64 \times (SPBRG + 1)} baud.$$

Le bit BRGH=0 : basse vitesse.
Le bit BRGH=1 : haute vitesse.

Les valeurs enregistrées dans le registre SPBRG sont de 0X00 à 0XFF, ce qui donne 256x2 vitesses de transfert différentes.

II.3.2) interface série RS232:

Une liaison série est une ligne où les bits d'information (1 ou 0) arrivent successivement, soit à intervalles réguliers (transmission synchrone), soit à des intervalles aléatoires, en groupe (transmission asynchrone), La liaison RS232 est une liaison série asynchrone.

a)Principe de communication série :

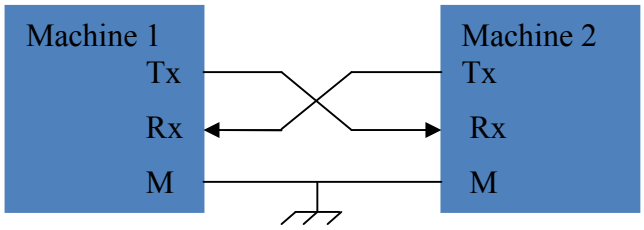


Figure 2.11 : communication série entre deux machines.

L'octet à transmettre est envoyé bit par bit (poids faible en premier) par l'émetteur sur la ligne Txt, vers le récepteur (ligne Rx) qui le reconstitue

La vitesse de transmission de l'émetteur doit être identique à la vitesse d'acquisition du récepteur. Ces vitesses sont exprimées en BAUDS (1 baud correspond à 1 bit / seconde, dans notre cas). Il existe différentes vitesses normalisées: 9600, 4800, 2400, 1200... bauds

La communication peut se faire dans les deux sens (duplex), soit émission d'abord, puis réception ensuite (half-duplex), soit émission et réception simultanées (full-duplex)

La transmission étant du type asynchrone (pas d'horloge commune entre l'émetteur et le récepteur), des bits supplémentaires sont indispensables au fonctionnement: bit de début de mot (start), bit(s) de fin de mot (stop)

D'autre part, l'utilisation éventuelle d'un bit de parité, permet la détection d'erreurs dans la transmission.

a.1) Communication série entre PC et PIC :

La communication série entre un PC et un microcontrôleur est assuré par les pins de transmission (Tx) et réception(Rx) dont disposent le port série du PC et le microcontrôleur. Le port série du PC fonctionne en +12/-12V et le microcontrôleur en +5/0V. Il faut donc convertir les signaux du microcontrôleur vers le PC et inversement. Ainsi nous utilisons un MAX 232 comme circuit intégré entouré de quatre condensateurs de même valeur.

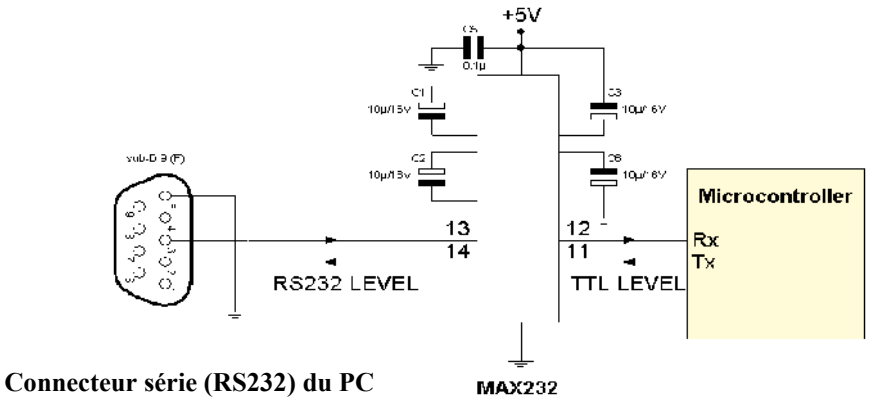


Figure 2.12 : interface PC-PIC.

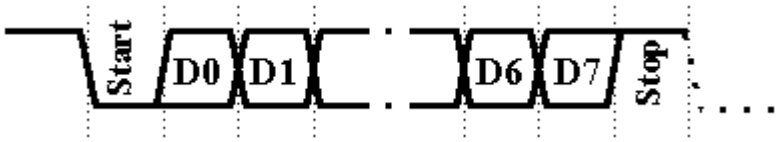
a.1.1) Signaux électriques du Port série (RS232) :

La liaison série RS232 exploite, pour la transmission, des signaux en +/- 12v :

- + 12 V: niveau logique "0"
- - 12 V : niveau logique "1"

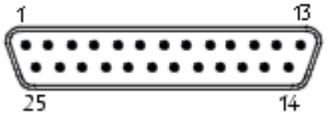
a.1.2) Les données transmises par le port série :

La transmission des données peut se faire suivant plusieurs formats (7 ou 8 bits) avec ou sans contrôle de parité (celle-ci) pouvant être gérée comme paire ou impaire); une trame commence par 1 bit de *start* ("0" logique) et se termine par 1 ou 2 bits de *stop* ("1" logique), le figure suivante présentent l'allure du format (RS232 8, n, 1 : 8 bits de données, pas de parité, 1 bit de stop) :

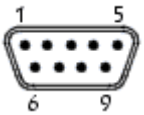


a.1.3) Brochage du connecteur RS232 :

Pour le PC, nous utiliserons le port série existant sous deux types de connecteurs mâles: DB 25 (25 broches) (a) et DB 9 (9 broches) (b), Notre carte de commande utilise un connecteur DB9 :



(a)



(b)

Le tableau ci-dessous présente les fonctions des broches d'un connecteur DB9 :

	Connecteur DB9	
TXD	3	Transmission de données
RXD	2	Réception de données
RQS ou RTS	7	Demande d'émission
CTS	8	Prêt à émettre
DSR	6	Données prêtes
SG	5	Masse logique
DTR	4	Terminal prêt
CD	1	Détection de porteuse
RI	9	-

Tableau 2.1 : Brochage d'un connecteur DB9.

En général, dans les applications de communication entre PC et PIC, que le pin (3) de transmission de données et le pin (2) de réception de données et le pin (5) de la masse sont utilisées.

a.1.4) Circuit intégré Max232 :

Lorsque, on désire interfacer un matériel quelconque vers le Pc, on utilise généralement un circuit bien connu, le **MAX232** à 16 broches, qui réclame 4 condensateurs chimiques de 1 à 10 µF extérieurs pour un fonctionnement correct.

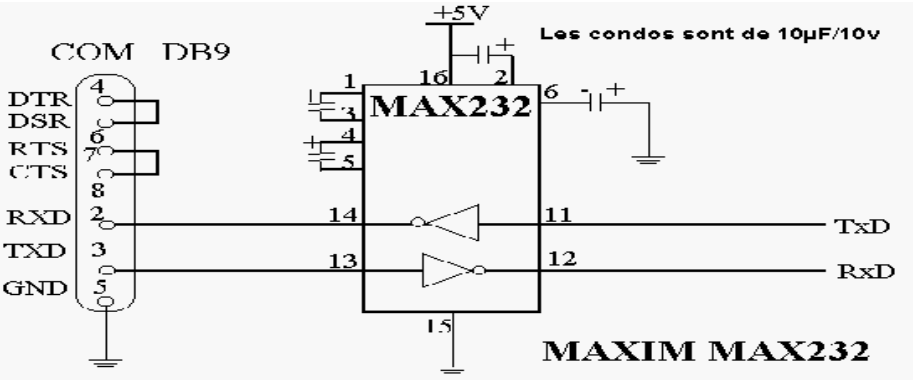


Figure 2.13 : branchage du circuit intégré MAX232.

II.3.3) les potentiomètres :

Deux potentiomètres POT1 et POT2 de 20KΩ alimentés par une tension de 5v et relier successivement aux entrées analogiques A0 et A1 du microcontrôleur sont utilisés, l'un pour varier la largeur d'impulsion et l'autre pour varier la fréquence du signal (PWM/PFM). L'utilisation d'une entrée analogique ou plusieurs et toujours accompagnée par deux tension de référence Vref+ (RA3) et Vref-(RA2) pour fixer la dynamique du convertisseur A/N du microcontrôleur. Ces potentiomètres nous offrent l'alternative analogique de contrôler la fréquence et/ou le rapport cyclique du signal désiré.

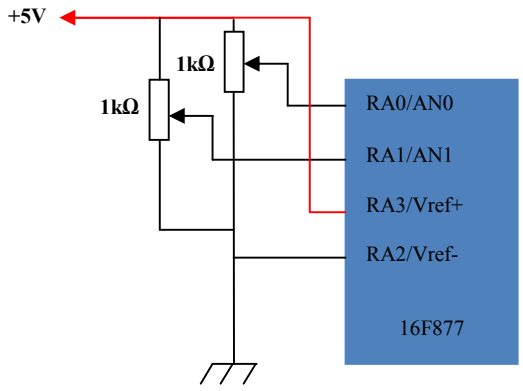


Figure 2.14 : branchage des potentiomètres au PIC 16F877.

Ces potentiomètres servent à varier la tension à l'entrée analogique du PIC de 0 à 5volts avant d'être convertie par le CAN du PIC à une valeur numérique correspondante, cette dernière qui sera utilisée par les algorithmes implémentés dedans.

II.3.4) L'entrée numérique:

Notre carte de commande comprend une entrée numérique, deux entre eux sont utilisés pour augmenter et diminuer la largeur d'impulsion et les autres pour varier la fréquence du signal (PWM/PFM). Ces potentiomètres nous offrent l'alternative numérique de contrôler la fréquence et/ou le rapport cyclique du signal désiré.

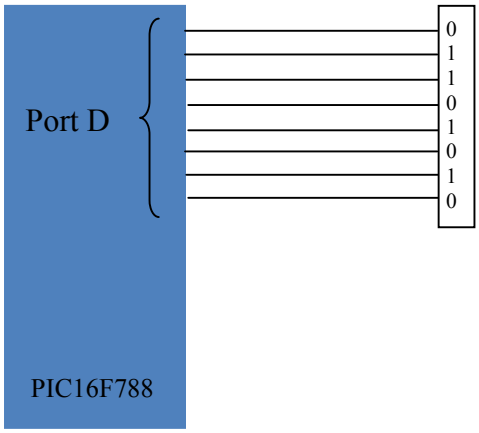
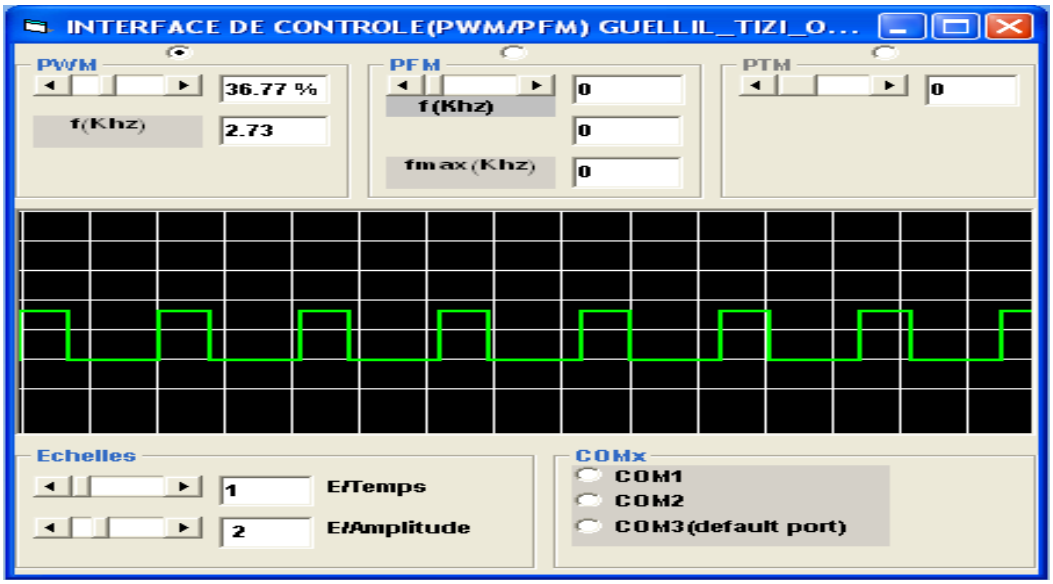


Figure 2.15 : branchage des boutons poussoirs au PIC 16F877.

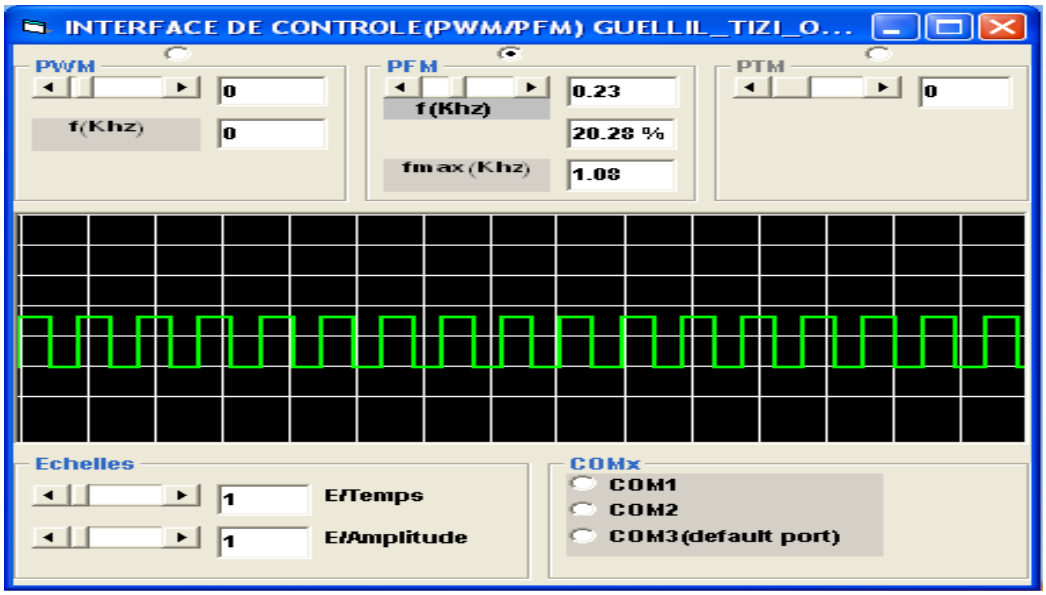
Cette entrée analogique est de 8 bits, donc elle nous permet de choisir une valeur de code 0x00 à 0xFF, la valeur choisie sera utilisée par les algorithmes implémentés dans le microcontrôleur.

II.3.5) Interface graphique de contrôle:

L'interface graphique est pour contrôler la fréquence du signal PFM ou le rapport cyclique du signal PWM par PC via le RS232 en envoyant de caractère du PC au PIC qui va les lire comme des entiers de 0 à 255 (code ASCII) qui seront récupérés par le PIC pour effectuer les calculs est générer le signal PWM ou PFM correspondant .
Le PIC lui même envoi des caractères au PC qui seront récupérer par l'interface pour faire des calculs et afficher soit la fréquence du signal PWM ou la fréquence maximale du signal PFM.



(a)



(b)

Figure 2.16 : Interface graphique de contrôle par PC du signal PWM(a) et PFM(b).

Cette interface graphique est conçue en Visuel Basic, elle nous offert le moyen de gérer la carte de commande par PC.

II.3.6) Les interrupteurs :

Les quatre interrupteurs que dispose la carte de commande sont utilisés comme des entrées logiques 0 et 1 pour des fins de sélection :

- L'interrupteur (P1) : leur mise à la masse nous permet de varier la fréquence du signal PWM ou la fréquence maximale du signal PFM par le bais du potentiomètre POT2.
- L'interrupteur (P2) : pour choisir le signal à générer (PWM ou PFM).
- Deux interrupteurs (S1, S2) : pour choisir le moyen de control des paramètres du signal (fréquence ou rapport cyclique), soit par l'entrée analogique, par l'entrée numérique ou par PC.

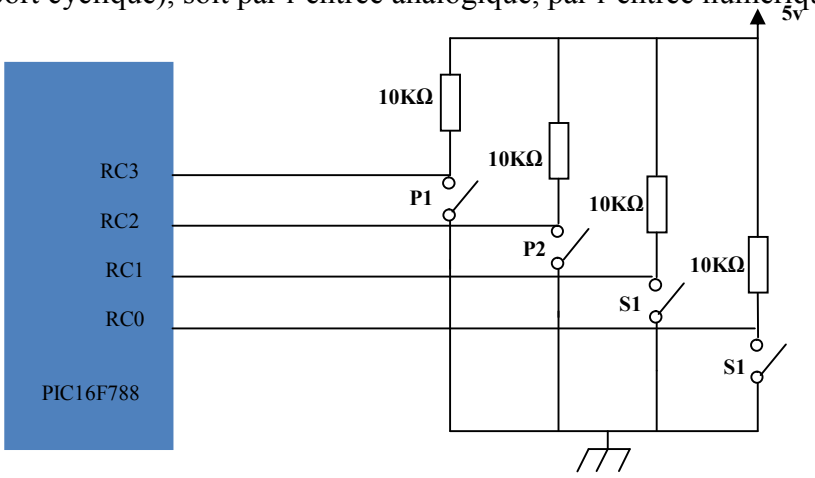


Figure 2.16 : La connexion des interrupteurs au PIC 16F877.

Ces interrupteurs sont utilisés comme suit :

Si : P1P2=00→ choisir la fréquence maximale du signal PFM de 1KHZ à 20 KHZ par POT2.

Si : P1P2=01→ choisir la fréquence du signal PWM de 1KHZ à 20 KHZ par POT2.

Si : P1P2=10→ générer le signal PFM pour Fmax choisie.

Si : P1P2=11→ générer le signal PWM pour F choisie.

Si : S1S2=00→ Utiliser l'entrée analogique.

Si : S1S2=01→ Utiliser l'entrée numérique.

Si : S1S2=10→ Utiliser le PC.

Cette diversité d'entrées rend la carte exploitable de plusieurs manières selon le besoin d'utilisateur.

La diversité des signaux générer permet d'utiliser cette carte dans de nombreuses applications industrielles.

II.3.7) Afficheur LCD :

a)Description d'un afficheur LCD 2x16 :

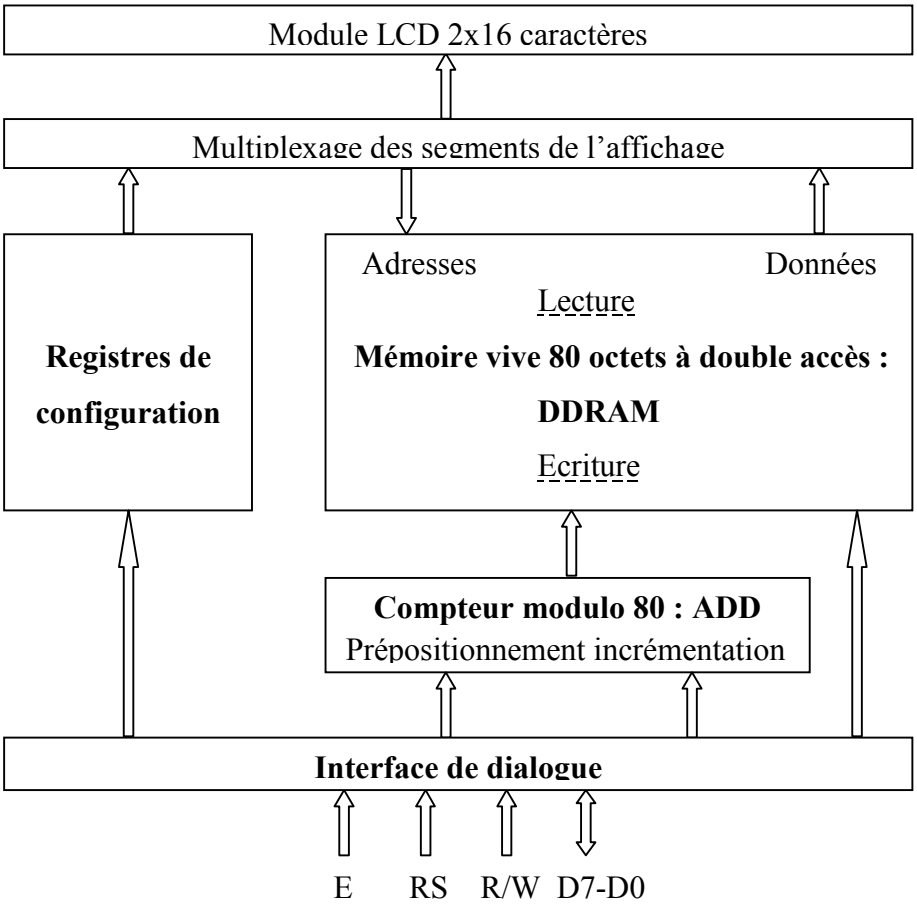
Au même titre que le port ‘’Centronics ‘’ d'un PC, le connecteur et la gestion des LCD dits intelligents sont devenus des standards.

En fait, la société Hitachi a développé il ya plusieurs années une famille de circuits intégrés de control et de pilotage d'afficheur constitué de caractère en matrice de points(5x8 ou 5x10).

Ces composants ont eu un grand succès, aujourd'hui tous les fabricants de LCD proposent des modules de taille diverses (1x8 caractères, 4x40 caractères ou plus) prêt à l'emploi et facile à mettre en ouvre.

Pour l'utilisateur, un module LCD de ce type se comporte comme une RAM de 80 octets à accès séquentiel, un module LCD dont chaque case mémorise le code ASCII du caractère affiché. Ainsi, pour présenter la lettre 'C' sur la 6ème position de la 1ère ligne, il suffit d'afficher la case mémoire d'adresse 5 avec l'octet 43h (code ASCII de 'C').

b) Schéma synoptique de l'afficheur LCD :



Les fonctions de gestion du curseur n'ont pas été représentées car celui-ci se met automatiquement dans la position correspondante à l'état du compteur ADD, L'utilisateur a toutefois la possibilité de changer sa forme (souligné, plein, clignotant).

L'utilisateur de l'afficheur doit s'acquitter de 2 ensembles de tâches :

-Affecter les registres de configuration en fonction du format (nombre de ligne par exemple).

-Affecter la mémoire DDRAM avec le message à afficher.

Pour ce faire, on doit disposer des signaux de contrôle E, RS, R/W et d'un bus de données de 8 bits bidirectionnels :

- **E :** enable**.**
- Cycle d'écriture : l'état du bus de données est pris en compte à l'état haut de ce signal.
- Cycle de lecture : la donnée est fournie pendant l'état haut de ce signal.
- **R/W :** indique le sens du transfert :
- ''0'' : écriture vers l'afficheur.
- ''1'' : lecture depuis l'afficheur.
- **RS :** Register select, l'état de ce signal indique la destination des données :
- ''0 '' : registre de contrôle.
- ''1'' : mémoire DDRAM.
- **D7-D0 :** bus de données bidirectionnel.

d) Affectation de la mémoire DDRAM :

C'est l'opération la plus simple. En effet, il faut produire un cycle d'écriture avec RS= 1, et une donnée valide (D7-D0) correspondant au code ASCII du caractère.

La donnée fournie est mémorisée à l'adresse ADD courante et le caractère est affiché à la position actuelle du curseur. Le compteur ADD est incrémenté automatiquement à la fin de cette opération (ce qui fait aussi avancer le curseur d'un cran) pour l'écriture du caractère suivant.

e) registres de configuration :

Il s'agit d'affecter les indicateurs de fonctionnement suivants :

DL=1 : interface en mode '' 8bits''.

0 : interface en mode ''4bits''.

N= 0 : 1 ligne.

1 : 2 lignes

F= 0 : caractères en 5x8 points.

1 : caractère en 5x10 points.

D= 0 : afficheur ''éteint '', mais le contenu de DDRAM est maintenu.

1 : afficheur ''allumé''.

C= 0 : curseur caché.

1 : curseur apparent (barre ''souligné'').

B= 0 : caractère du curseur affiché normalement.

1 : caractère du curseur clignotant (positif/négatif).

I/D=1 : le compteur d'adresse ADD s'incrémente à chaque caractère entré.

0 : le compteur d'adresse ADD se décrémente à chaque caractère entré.

S= 0 : le texte déjà affiché reste figé et le curseur se déplace à chaque caractère entré..

1 : le curseur reste figé et le texte déjà se déplace à chaque caractère entré.

On note qu'au reset, chaque indicateur est affecté par le premier état décrit.

e) Affectations des registres de configuration et instruction :

L'affectation des indicateurs de fonctionnement est réalisée par le biais d'instructions. Des instructions complémentaires permettent de réaliser certaines opérations.

Ecrire une instruction consiste à produire un cycle d'écriture avec RS=0. Les bits de poids fort de la donnée écrite sont utilisés pour identifier l'instruction et les poids faibles contiennent les paramètres.

f) Affectation des indicateurs de fonctionnement :

Instructions	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Function set	0	0	**0**	**0**	**1**	**DL**	**N**	**F**	*	*
Display on/off control	0	0	**0**	**0**	**0**	**0**	**1**	**D**	**C**	**B**
Entry mode set	0	0	**0**	**0**	**0**	**0**	**0**	**1**	**I/D**	**S**

Tableau 2.2 : Affectation des indicateurs de fonctionnement.

Instructions	Durée	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Display clear	1.52ms	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
Cursor home	1.52ms	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	*

Cursor display shift	40µs	0	0	**0**	**0**	**0**	**1**	**S/C**	**R/L**	*	*
CG RAM adress	40µs	0	0	**0**	**1**	**ACC**					
DD RAM address set	40µs	0	0	**1**	**ADD**						

Tableau 2.3 : Instruction de commande.

-Les états des bits de données représentés en gras sont utilisés pour identifier les instructions.

-la valeur ‘’*’’ indique un état indifférent.

-Display clear :

-rempli la mémoire DDRAM avec le caractère ‘’ ‘’ (Code ASCII).

-Raz du compteur ADD.

-Curseur sur la 1ère position.

-Display home :

-Raz du compteur ADD.

-Curseur sur la 1ère position.

-Contenu DDRAM inchangé.

-Cursor display shift :

-S/C, R/L=00 : le curseur se déplace d'une position à gauche.

-0 1 : Le curseur se déplace d'une position à droite.

-1 0 : Le texte et le curseur se déplacent d'une position à gauche.

-1 1 : Le texte et le curseur se déplacent d'une position.

-CG RAM address set : prépositionnement du compteur ACC d'adressage de la mémoire CGRAM avec le paramètre.

-DD RAM address set : prépositionnement du compteur ACC d'adressage de la mémoire CGRAM avec le paramètre. Le curseur est placé dans la position correspondante au caractère.

g) Brochage du connecteur :

Les connecteurs peuvent être du type SIL (Single IN Line) ou DIL (Dual IN Line) à 16 broches :

Broche N°	Nom	Niveau	Fonction
1	Vss	0V	Alimentation
2	Vcc	2.7V à 5.5V	Alimentation
3	V0	-5V à 5V	Contraste
4	RS	H/L	0=instruction/1=caractère
5	R/W	H/L	0=écriture/1=lecture
6	E	H→L	Enable (front montant)
7	D0	H/L	Donnée bidirectionnelle LSB
8	D1	H/L	Donnée bidirectionnelle
9	D2	H/L	Donnée bidirectionnelle
10	D3	H/L	Donnée bidirectionnelle
11	D4	H/L	Donnée bidirectionnelle
12	D5	H/L	Donnée bidirectionnelle
13	D6	H/L	Donnée bidirectionnelle
14	D7	H/L	Donnée bidirectionnelle MSB
15	A		Anode- LED de rétro éclairage
16	K		cathode- LED de rétro éclairage

Tableau 2.3 : Brochage du connecteur de l'afficheur LCD 2x16 caractères.

II.3.8) L'étage d'isolation galvanique :

Isolation galvanique parfait entre deux circuits est assuré par les photo coupleurs Ils transmettent, grâce à des photons, des informations logiques ou analogiques entre l'entrée et la sortie de ces composants.

a) Principe :

Cet isolement permet de coupler des circuits ayant une grande différence de potentiel, sans risque d'interférence par une boucle de terre, due au câblage. Ils permettent l'emploi, dans une large plage de fréquences, entre le courant continu et la haute fréquence, grâce à leurs caractéristiques et à leurs faibles dimensions ; ils sont de loin supérieurs aux transformateurs et aux convertisseurs. L'isolation entre l'entrée et la sortie est comprise entre 2500 volts et 7500

volts. Quand à la capacité et la résistance d'isolement elle est respectivement de 0.2 à 2 pF et de $10^{10}\Omega$ à$10^{13}\Omega$.

b) Description :

Un photocoupleur est constitué par l'assemblage de deux cristaux semi-conducteurs :

- L'un en entrée est un émetteur de lumière.
- L'autre en sortie est un photorécepteur.

Ces deux composants sont couplés dans un même boitier étanche à la lumière ambiante.
Le tableau ci-dessous décrit les différents dispositifs utilisés en entrée et sortie.

Entrée	Sortie
Diode	Diode
Diode	Phototransistor
Diode	Darlington
Diode	Triac

Tableau 2.4 : Dispositif utilisé en entée et sortie d'un photocoupleur.

Le dispositif d'entrée est toujours une diode. Attention, celle-ci est réalisée en AsGa, sa tension de seuil est donc légèrement plus élevée que pour les diodes de redressement au silicium (1V environ).

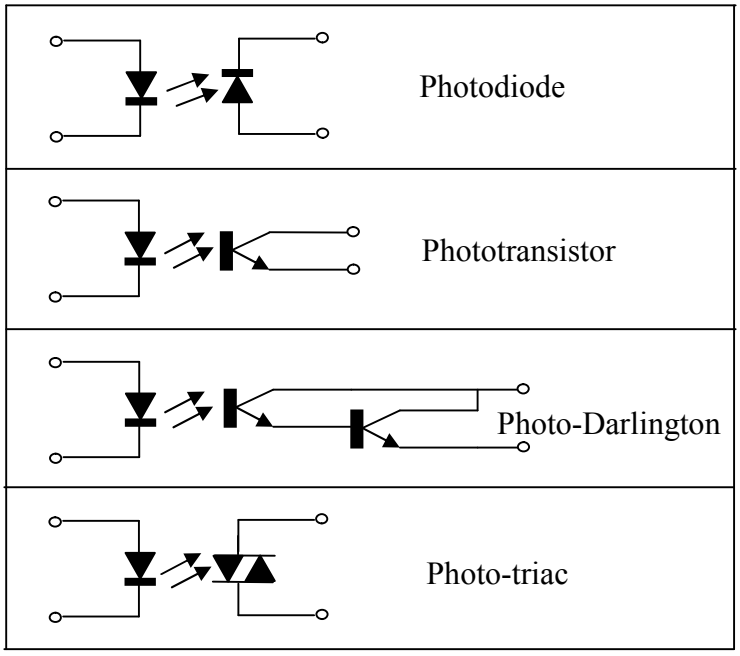


Figure 2.18 : les différents types de photocoupleurs.

Le dispositif de sortie dépend de l'application visée (puissance pour les triacs et photocoupleurs, linéarité pour les phototransistors (transmission de signaux analogiques) ou rapidité pour les diodes (transmission de signaux logiques. Pour notre carte on va utiliser un circuit appelé PC847 à base de phototransistor.

c) Le circuit intégré PC847 :

Cette isolation optique est réalisée par quatre photocoupleurs (un pour chaque sortie). Un photocoupleur ce compose d'une diode et d'un phototransistor. Lorsque la die est alimentée, elle génère un rayon lumineux qui vient frapper la base du photocoupleur qui passe alors de l'état bloqué à l'état passant.

Pour rendre le montage plus compact nous avons utilisé un circuit intégré comportant 4 phototransistors (figure 2.19). Ce circuit dispose d'une tension d'isolation de 5KV, nous voilà donc rassuré. Le circuit PC847 utilisée ici.

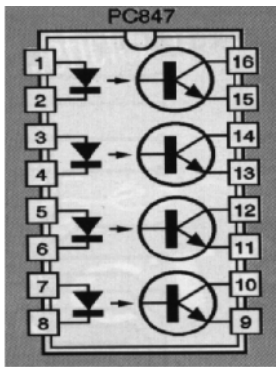


Figure 1.19 : CI PC847.

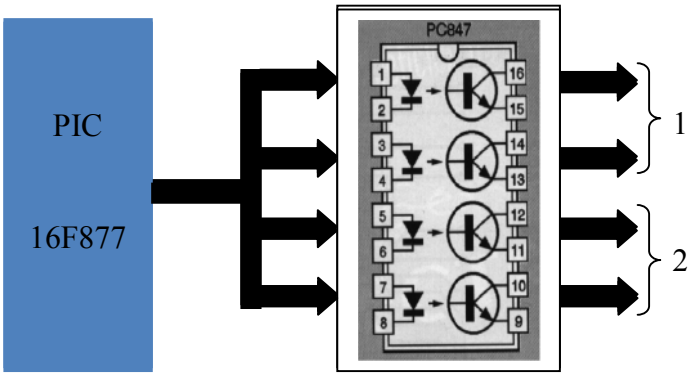


Figure 1.20 : La connexion du CI PC 847 au PIC.

Les quatre diodes du circuit intégré PC847 sont liées au PIC via la broche de sortie du signal, ce qui nous donnera le même signal ou sont inverse sur le collecteur ou l'émetteur du phototransistor.

1 : deux sorties non-inversées (Le phototransistor est monté en collecteur commun).

2 : deux sorties inversées (Le phototransistor est monté en émetteur commun).

d) Réponse temporelle et fréquentielle du circuit d'isolation galvanique :

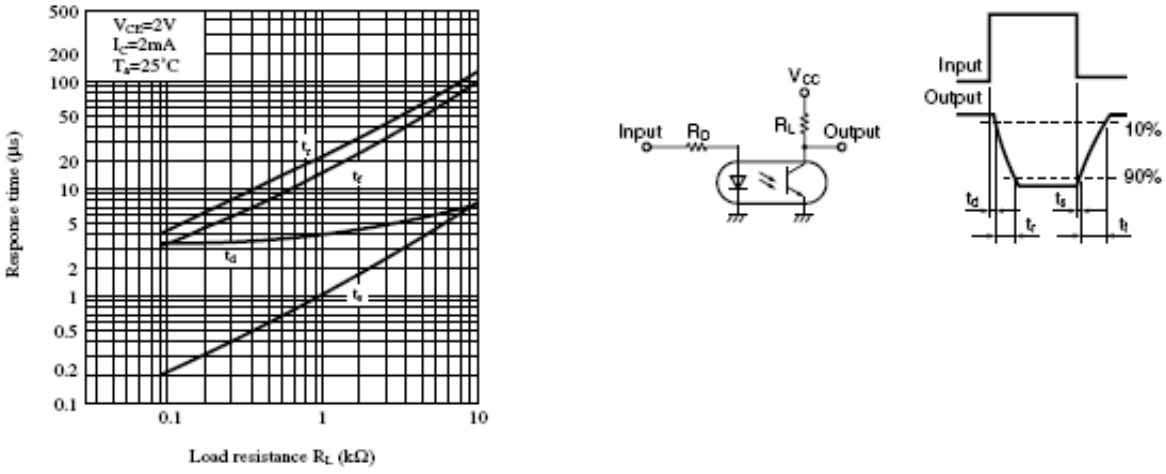


Figure 1.21 : La réponse temporelle du photocoupleur en fonction de la charge R_L .

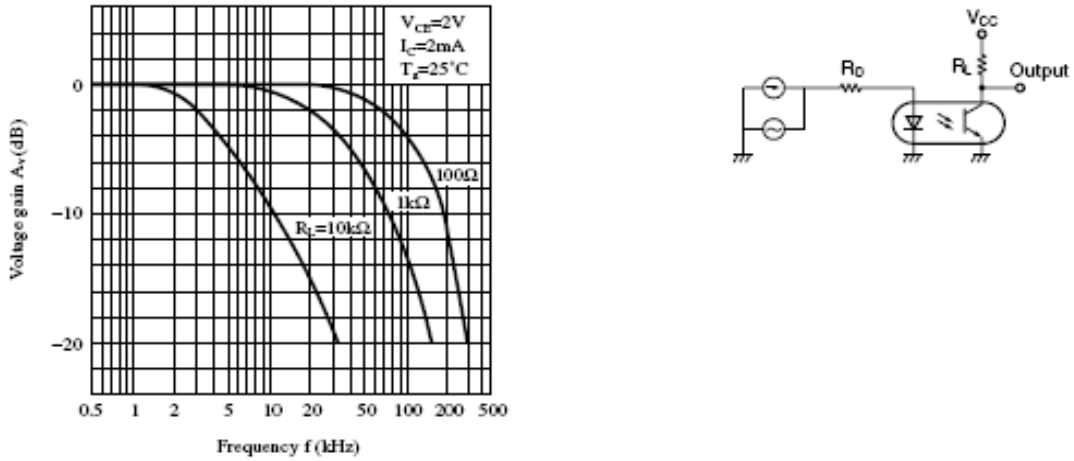


Figure 1.22 : la réponse fréquentielle photocoupleur en fonction de la charge R_L .

De la figure 1.22, on remarque que la largeur de la bande passante du photocoupleur est inversement proportionnelle à la charge R_L.

Tant qu'un signal PWM/PFM est riche en harmonies, le choix de la charge R_L doit faire l'objet d'une étude qui prend en considération la fréquence de ce signal, le choix d'une résistance de faible valeur est de grande importance pour élargir la bande passante du photocoupleur et par conséquent garder la forme du signal moins distordue.

L'exemple suivant montre comment la charge R_L peut influe sur la forme le signal :

-On utilise deux résistances de 1K et 10K pour un signal d'horloge de 5KH.

-L'oscilloscope nous permet de visualiser les différents signaux.

- ✓ Le signal en couleur jaune, c'est le signal d'entrée.
- ✓ Le signal en couleur bleu, c'est le signal de sortie du photocoupleur de charge de 10K.
- ✓ Le signal en couleur rouge, c'est le signal de sortie du photocoupleur de charge de 1K.

Remarque :

Le signal en couleur bleu et fortement distordu car un grand nombre des harmonies de haute fréquence ont été atténuées.

Le signal en couleur rouge et faiblement distordu car un nombre moins important des harmonies de haute fréquence ont été atténuées.

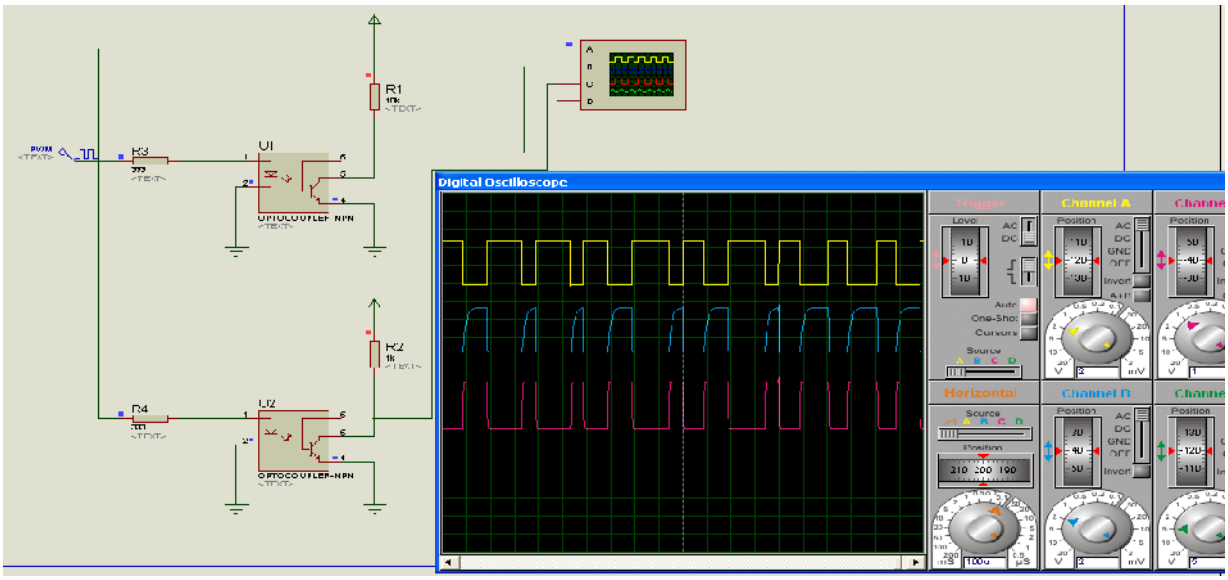


Figure 1.23 : exemple illustratif de la réponse fréquentielle photocoupleur en fonction de la charge R_L .

II.3.9) L'étage d'amplification :

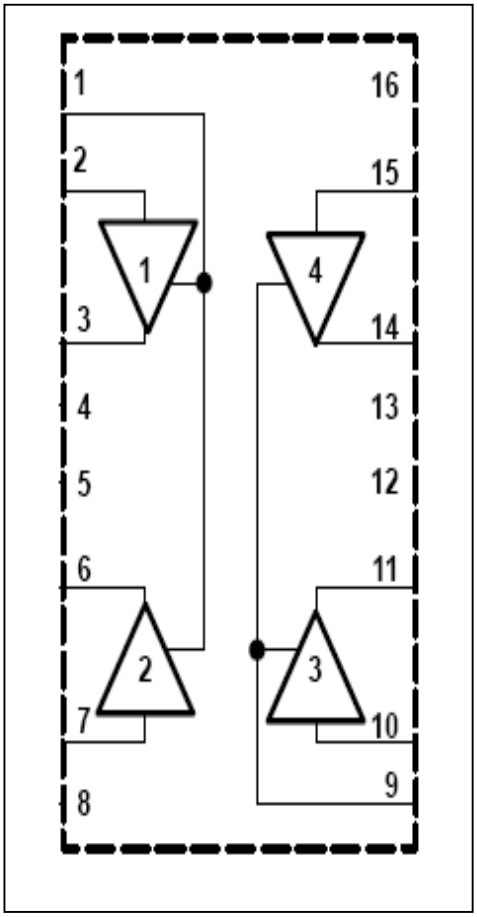
le signal qui sort de l'étage d'isolation galvanique ne sera pas être utilisé directement dans le pilotage de différents dispositifs électriques, mais il subira un première amplification locale (au niveau de la carte de commande) par le biais d'un amplificateur de puissance qui lui donnera les performances qui lui permettent de répondre aux exigences de courant et de tension nécessaires aux pilotage de nombreux systèmes électriques.

a) L'amplificateur de puissance L289D :

Le circuit de puissance L293D contient quatre amplificateurs digitaux munis de diodes de protection contre les surtensions dues aux charges inductives, il peut délivrer un courant qui arrive jusqu'à 600mA et une tension qui varie de Vcc à 36V.

a.1) Brochage du circuit L293D :

Broche N°	Fonction
1	Validation de l'amplificateur 1et 2 (VCC1)
2	L'entrée de l'amplificateur 1
3	La sortie de l'amplificateur 1
4	La masse
5	La masse
6	La sortie de l'amplificateur2
7	L'entrée de l'amplificateur 2
8	Alimentation VCC2 (VCC1, 36V)
9	Validation de l'amplificateur 3et 4 (VCC1)
10	L'entrée de l'amplificateur 3
11	La sortie de l'amplificateur 3
12	La masse
13	La masse
14	La sortie de l'amplificateur 4
15	L'entrée de l'amplificateur 4
16	Alimentation VCC1 (4.5V, 7V)



Recommandation et remarque:

Le signal d'entrée de l'amplificateur est un signal logique d'état bas VL, et état haut VH ,on doit respecter la marges de tension suivante :

VH : 2.3 V → VCC1.

Le signal de sortie aura l'image du signal d'entée avec un état haut :

VH=VCC2

III.1.Introduction :

La programmation d'applications à base de microcontrôleurs PIC en langage C suscite de nombreuses questions, tant de la part des débutants que de celle des programmateurs chevronnés. En effet, si les bases du langage restent conformes au langage ANSI C bien connu, la gestion des ressources internes du microcontrôleur impose de faire appel à de nombreuses fonctions nouvelles, spécifiques de ce type de circuit.
En outre, si vous voulez pouvoir mettre au point votre application avec un maximum de chance de succès, le retour à un environnement de développement spécial, allant de l'éditeur de programme au simulateur en passant par le compilateur bien sûr, est nécessaire.

III.2.Les compilateurs C pour PIC :

II existe aujourd'hui de nombreux compilateurs C pour PIC et le choix de l'un d'entre eux presque aussi important que celui du microcontrôleur lui-même lors du développement d'une application. En effet même si les différents compilateurs C pour PIC actuellement disponibles sur le marché sont pour la plupart de grande qualité, leur confort d'utilisation varie d'un compilateur à un autre, ce qui peut avoir une influence non négligeable sur les délais de développement d'une application.
Les compilateurs C pour PIC les plus connus sont :
1- Les compilateurs C de CCS.
2- Les compilateurs C de Hi-Tech software.
3- Les compilateurs CC5X et CC8E.
4- Les compilateurs FED et WIZ-C.
5- Les compilateurs C18 de Microchip.
Il existe d'autres tels ceux de Byte Craft ou bien encore d'IAR.

III.2.1 Les compilateurs C de CCS :

Le compilateur C de CCS appelé PCW fait partie des moyens de développement de notre projet, vue sa flexibilité et richesse en outils. On s'amuse de cité quelques avantages que ce compilateur nous offert :
-Librairies de fonctions compatibles de tous les PIC support pour ce qui est des interfaces

RS232, I2C, des entrées/sorties parallèles et de la gestion précise de délais.

- Intégration avec l'environnement de développement MPLAB permettant d'utiliser le simulateur contenu dans ce dernier pour la mise au point des programmes écrits en C.
- Gestion optimisée des appels de fonctions autorisant des imbrications plus importantes que ce que permet théoriquement la taille de la pile des PIC.
- Accès à toutes les ressources matérielles internes des PIC au moyen de fonctions en C très simples d'emploi.
- Supports des types entiers sur 1, 8,16 et 32 bits et des types flottants sur 32 bits.
- Support de l'insertion de code assembleur en n'importe quel point du listing source en C. Ce code assembleur peut en outre faire appel aux variables déclarés dans le programme en C.
- Les constantes utilisées par le programme sont sauvegardées dans la mémoire programme.
- L'éditeur de liens détermine automatiquement ou manuellement si un bloc de code répétitif doit être appelé comme une fonction ou doit être répété autant de fois que nécessaire, selon que l'on souhaite privilégier la vitesse d'exécution ou la compacité du code.
- Les codes source des pilotes de très nombreux circuits externes standards sont fournis et sont prêts à être intégrer dans nos applications, par exemple : interface pour clavier en matrice, EEPROM série, circuit « un fil » de Dallas, capteurs de température, potentiomètre numérique, etc.
- Support complet des interruptions assurant la détection de la source et la gestion intégrale des bits d'indications d'état et de validation les concernant.
- Possibilité d'impression, sur le listing source, du code assembleur généré par la compilation du programme.
- Fenêtre de visualisation de la cartographie mémoire utilisée, des arbres d'appels de fonction et des statistiques d'utilisation de la mémoire du microcontrôleur choisi.
- Information de configuration du circuit utilisé incluses dans le code source afin de gérer correctement la programmation des bits de configuration.

Il y a en effet, cinq versions de compilateur C de CCS (PCB, PCM, PCH, PCW, PCWH), les versions les plus intéressantes sont évidemment les versions PCW pour ceux qui souhaitent pas développer d'application à base de la famille 18xxxx et PCWH pour ceux qui veulent pouvoir travailler avec toutes les familles de PIC. En effet, ces deux versions intègrent en outre un environnement complet, dont on peut voir la fenêtre principale à titre d'exemple figure III.1.

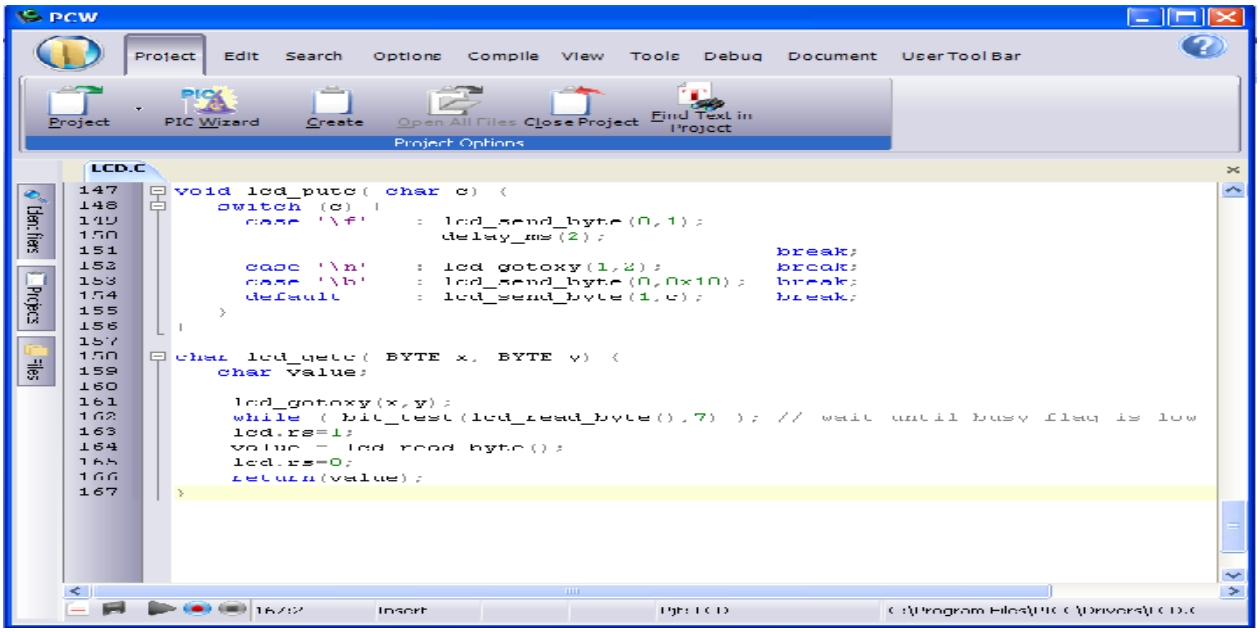


Figure III.1 : fenêtre principale de l'environnement de développement offert par le compilateur CCS.

III.2.2. L'intégration dans MPLAB :

L'environnement MPLAB de Microchip est nécessaire afin de rendre plus confortables D'emploi des compilateurs qui ne travaillent qu'en mode ligne de commande, on est en droit de se demander ce qu'elle apporte dans le cas d'un environnement tel celui de CCS qui comporte déjà sous forme graphique tous les outils nécessaires à la compilation.

En fait, et dans ce cas elle permet d'accéder au simulateur intégré dans MPLAB et donc de simuler le programme écrit en C de la même façon que de qu'il est possible de faire lorsqu'on travaille en langage machine.

Même s'il est possible de forcer le simulateur « à la main » sans réaliser cette intégration, il serait ridicule de ne pas la réaliser car le passage d'un environnement à l'autre est automatique et transparent.

Donc il est recommandé d'installer MPLAB avant d'installer le compilateur C ; cela facilitera ensuite l'intégration de ce dernier dans l'environnement MPLAB, dont on peut voir la fenêtre principale à titre d'exemple figure III.2.

Cet environnement contient plusieurs moyens de communication avec l'environnement extérieur (les différents programmateurs, Matlab/simulink, Proteus,…).

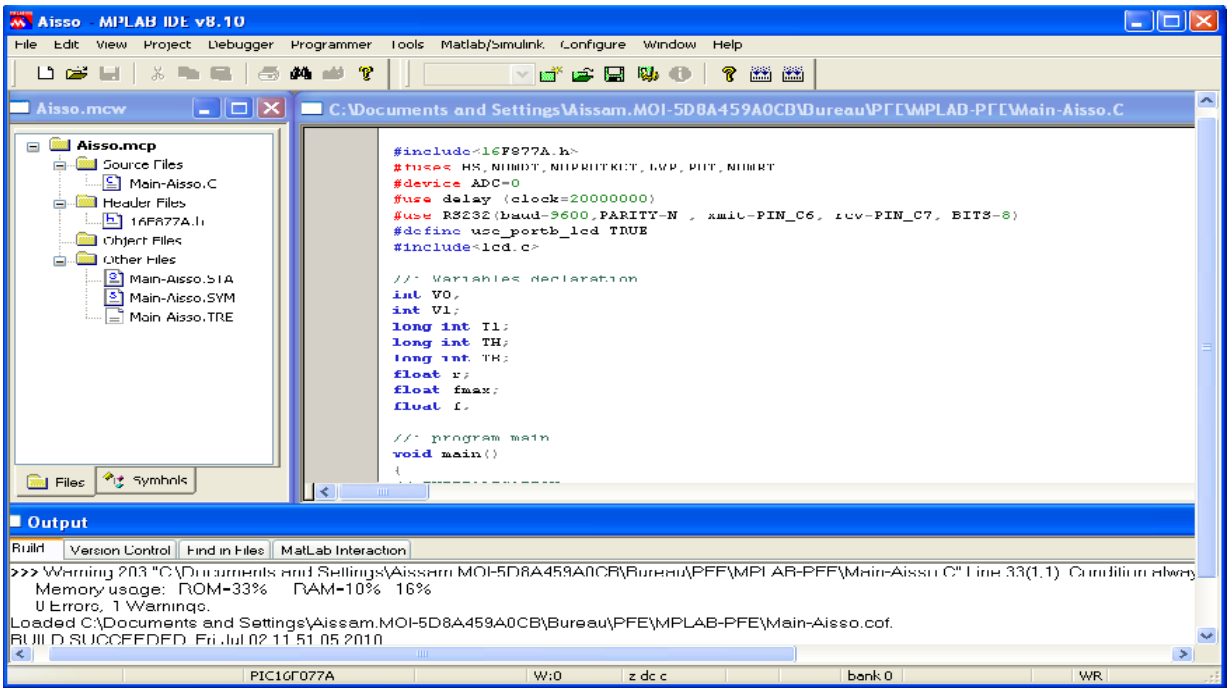


Figure III.2 : fenêtre principale de l'environnement MPLAB.

Donc, le compilateur C de CCS intégré dans l'environnement MPLAB nous offert un moyens idéal pour la programmation des PIC et avec une grande optimalité, sans avoir besoin à d'autre logiciels complémentaires.

III.3. Les principales directives fonctions spécifiques au langage C des PIC (nécessaires au programme de commande):

III.3.1 Les directives :

Lors de la compilation d'un programme écrit en C on à coutume de dire que le compilateur traduit le langage machine. C'est certes exact dans son ensemble, mais c'est un peu réducteur quant aux différentes étapes qui se produisent réellement, en fait, avant même cette traduction, un premier programme intervient :

Le préprocesseur. Ce dernier interprète un certain nombre de directives qui lui sont propres en fournit en conséquent des informations au compilateur proprement dit ; compilateur dont le travail ne commence donc qu'après l'intervention de ce préprocesseur.

Il existe deux types de directives : les directives d'informations de compilation et les directives de définitions matérielles.

a) les directives d'informations de compilation :

Un certain nombre de directives du préprocesseur peuvent être qualifiées d'informations de compilation. Elles permettent en effet d'informer le compilateur sur l'affectation de noms ou d'étiquettes à des positions mémoires ou des registres particuliers. Voici les plus utiles d'entre eux pour notre projet.

#Define

Cette directive permet de substituer du texte, au sens large du terme, par un identifiant choisi par vos soins. Elle s'utilise sous l'une des deux formes suivantes :

#define id texte

Elle substituera l'identifiant par le texte qui le suit toutes les fois où il apparaîtra dans le programme.

#Include

Elle s'utilise de la façon suivante :

#include<nom de fichier> ou *#include 'nom de fichier'*

Dans les deux cas, le fichier dont le nom est spécifié est inclus dans le listing source contenant cette directive. La différence entre les deux syntaxes n'a d'influence que sur l'ordre dans lequel le fichier à inclure est recherché dans les répertoires de la machine utilisée, dans le premier cas le répertoire contenant le listing source est parcouru en dernier alors que, dans le deuxième cas il est parcouru en premier.

b) les directive de définition du matérielles :

Contrairement aux précédentes, ces directives sont liées au processeur utilisé. Elles permettent d'en définir le type, les fusibles de configuration utilisés, les interruptions à valider, etc.
Voici les plus utiles d'entre eux pour notre projet.

#Device

Comme son nom le laisse supposer, cette directive indique au compilateur quel est le type de processeur utilisé ainsi qu'un certain nombre d'information telles que :

-la taille de pointeurs (*=5, *=8, *=16 selon que l'on souhaite (5, 8 ou 16 bits) ;

-le nombre de bits du convertisseur analogique/digital interne (ADC=x) ;

-la validation du mode de mise au point (« debug ») en circuit (ICD=TRUE).

Cette directive s'utilise de la façon suivante :

#device processeur options

Ainsi par exemple :

*#device PIC16F877 *=16 ADC=10*

Indique que l'on utilise un 16F877 avec des pointeurs sur 16 bits et que la résolution du convertisseur analogique/digital est fixée à 10 bits

#Fuses

Cette directive indique comment programmer les bits ou fusibles de configuration du PIC. La syntaxe et la suivante :

#fuses liste et état des fusibles

Liste et état des fusibles est une suite d'informations séparées par des virgules, précisant les fusibles à programmer ou l'état de la fonction à valider ou non à choisir parmi :

-*LP, XT, HS, RC* pour le type d'horloge.

-*WDT* ou *NOWDT* pour valider ou non le timer chien de garde.

- *PROTECT* ou *NOPROTECT* pour protéger ou non la mémoire de programme.

-*PUT* ou *NOPUT* pour valider ou non le timer à la mise sous tension.

-*BROWNOUT* ou *NOBROWNOUT* pour valider ou non la détection de baise anormale de tension d'alimentation.

Ainsi par exemple :

#fuses XT, NOWDT, PUT, BROWNOUT

#Use delay

Cette fonction permet au compilateur de connaitre la fréquence d'horloge exacte du PIC.

Elle s'utilise sous la forme suivante :

#use delay (clock=vitesse)

Où *vitesse* est la fréquence d'horloge exprimée en hertz.

#Use rs232

Cette directive indique au compilateur de faire usage de l'interface série asynchrone, appelée RS232 par un raccourci un peu restrictif, et précise ses différents modes d'utilisation au moyen de la syntaxe suivante :

#use rs232 (options)

Où options est constitue par un ou plusieurs mots clés dont on présente les plus importants pour notre travail dans le tableau suivant :

Options de #use rs232	**Signification**
baud=n	Définit la vitesse de transmission en bits par seconde
Xmit=pin	Définit la patte d'émission des données
Rcv=pin	Définit la patte de réception des données
Parity=n	Définit la parité (N=aucune, E=paire, O=impaire)
Bits=n	Définit le nombre de bits de donnée (5 à 9)

III.3.2.Les fonctions :

Les directives du préprocesseur ne suffisent pas pour qu'un compilateur C, quel qu'il soit, puisse prendre en compte les particularités d'un microcontrôleur. En effet, la gestion performante de ses entées/sorties et de ses ressource internes impose d'ajouter au langage C « de base » un certain nombre de fonctions spécifiques.

C'est même de la richesse de ces ajouts que dépend aujourd'hui la facilité d'emploi d'un compilateur car cela décharge le développement d'un certain nombre de tâches de programmation fastidieuses. Voici donc, classées par ordre alphabétique, les principales fonctions spécifiques du compilateur C de CCS qu'on a besoin pour développer notre programme de commande de l'unité à réaliser.

Delay_US()

Comme son nom l'indique cette fonction réalise une boucle d'attente dont la durée est spécifiée en microsecondes entre 1 et 65 535. Elle s'utilise de la façon suivante :

Delay_Us(nombre)

Où le nombre est une constante comprise entre 1 et 255 ou entre 1 et 65 535 qui indique la durée d'attente en microsecondes.

Cette fonction ne fait appel à aucun timer du PIC. Ceux-ci restent donc totalement libres pour l'application. Par contre, comme le délai généré par cette fonction ne repose que sur des

durées d'exécution d'une suite d'instruction bien précise, la survenue d'une interruption pendant l'exécution de cette fonction allonge allonge le délai spécifié de manière imprévisible.

Getc(), getch, getchar, fget()

Ces fonctions sont classiques en langage C mais nécessitent une petite précision dans le cas des compilateurs C. les trois première formes attendent un caractère depuis la liaison série RS232 du PIC ne comportant pas d'identifiant de flux, telle qu'elle a été définie par la directive # use rs232.

La syntaxe est la suivante :

Valeur= getc () ou getch() ou getchar() dans le premier cas et

Valeur= fgetc(flux) dans le deuxième cas.

Où *valeur* est un variable contenant le caractère reçu et où *flux* est une constante spécifie le flux utilisé

Input()

Cette fonction indique l'état logique de la ligne de port spécifiée dans une variable de type bits qu'elle rend égale à 0 ou à 1, Contrairement à INPUT, elle respecte le mode de traitement des ports parallèles imposé au préalable par la directive #use xxx_IO et peut être donc amenée à forcer la ligne correspondant en entée avant de lire son état. Elle s'utilise de la façon suivante :

Input_x ()

Cette fonction lit d'un seul coup l'état des huit lignes du port spécifié. Elle respecte le mode de traitement des ports parallèles imposé au préalable par la directive #use xxx_IO et peut donc être amené à forcer le port correspondant en entrée avant de lire son état. Elle s'utilise de la façon suivante :

Valeur= input_x()

Où x est la lettre de désignation du port choisi, comprise entre A et G selon le PIC utilisé.

Ainsi, par exemple :

Valeur= input_b

Lit les données présentes sur les 8 lignes du port B.

Output_high()

Cette fonction met au niveau logique haut la ligne de port spécifiée. Elle respecte le mode de traitement des ports parallèles imposé par la directive #use xxx_IO et peut donc être amenée à forcer la ligne correspondant en sortie avant d'y écrire la valeur désirée. Elle s'utilise de la façon suivant :

Output_hign(ligne de port)

Où *ligne de port* correspondant à la définition d'une ligne de port d'entée/sortie telle qu'elle est faite dans le fichier .h du PIC utilisé. Ainsi par exemple :

Output_high(PIN_C7)

Met au niveau logique haut le bit de poids fort du port C.

Output_low()

Cette fonction met au niveau logique bas la ligne de port spécifiée. Elle respecte le mode de traitement des ports parallèles imposé par la directive #use xxx_IO et peut donc être amenée à forcer la ligne correspondant en sortie avant d'y écrire la valeur désirée. Elle s'utilise de la façon suivant :

Output_low(ligne de port)

Où *ligne de port* correspondant à la définition d'une ligne de port d'entée/sortie telle qu'elle est faite dans le fichier .h du PIC utilisé. Ainsi par exemple :

Output_low(PIN_C0)

Met au niveau logique haut le bit de poids faible du port A.

Putc(), putchar(), fputc()

Ces fonctions sont classiques en langage C mais nécessitent une petite précision dans le cas des Compilateur C pour PIC. Les deux premières formes émettent un caractère sur la liaison série RS232 du PIC ne comportant pas d'identifiant de flux, telle qu'elle a été définie par la directive #use rs232.

La forme *fputc ()* quant à elle émet un caractère mais via le flux qui y est spécifié. Ce flux fait alors référence à une liaison série RS232 du PIC telle que définie au moyen de la directive #use rs232, mais spécifiant cette fois-ci le même flux.

La syntaxe est la suivante :

Putc(donnée) ou putchar(donnée) dans le premier cas et

Fputc(donnée, flux) dans le deuxième cas.

Où *donnée* est une variable contenant le caractère à émettre et où *flux* est une constante spécifie le flux

Read_adc()

Cette fonction lit le résultat fourni par le convertisseur analogique/numérique contenu dans le PIC sélectionné. Selon le type de PIC et le mode de fonctionnement du convertisseur, le résultat est un entier codé sur 8 bits ou 16 bits. Elle s'utilise de la façon suivante :

Valeur =read_adc(mode)

Où *valeur* contient le résultat de la lecture et où *mode* est un paramètre optionnel qui peut prendre une des trois valeurs suivantes :

-*Adc_start_and_read* qui est la valeur par défaut et qui pour effet de démarrer une conversion puis d'en fournir le résultat effectif.

-*Adc_start_only* qui n'a pour effet que de démarrer la conversion mais sans attendre son résultat.

-*Adc_read_only* qui lit le résultat de la conversion précédente.

Set_adc_channel()

Cette fonction permet de spécifier quel canal d'entrée utiliser lors de l'appel de la fonction *read_adc()* qui fait suite. Il est nécessaire de patienter quelques instants entre l'appel de cette fonction et celui de *read_adc()*. Une durée de 10µs est usuellement suffisante. La syntaxe de cette fonction est la suivante :

Set_adc_channel(numéro)

Où *numéro* est le numéro de canal d'entrée du convertisseur analogique/digital tel qu'indiqué dans la fiche technique du PIC utilisé.

Setup_x()

Nous abordons maintenant toute une série de fonctions qui servent en fait à paramétrer diverses ressources internes du PIC en positionnement correctement les bits des registres de contrôle de ces derniers. Lorsque l'on programme en assembleur, on fait cela « à la main » en utilisant la

fiche technique du circuit pour voir quels sont les bits à positionner pour obtenir tel ou tel comportement.

Avec un compilateur C, il est possible de procéder de la même façon mais, très souvent, et c'est le cas pour le compilateur de CCS, des fonctions permettent de réaliser ces paramétrage de façon plus conventionnelle.

Par exemple, si vous voulez utiliser la fonction *Setup_adc()*, les différents valeurs du paramètre à utiliser sont :

Adc_off, adc_clock_internal, ou encore *adc_clock_div_32.*

Les appellations données à ces différents paramètres correspondent en fait aux divers modes de fonctionnement ou de configuration de la ressource concernée tels qu'ils sont décrits dans la fiche technique éditée par Microchip.

III.4.Programme de commande :

Dans cette patrie on détaillera le fonctionnement du programme de commande en donnant son organigramme et quelques explications.

Rappel :

Sur notre carte de commande il existe quarte interrupteurs P1, P2, S1, S2 qui représentent le clavier de la carte, on rappelle le rôle de chaque interrupteur pour mieux comprendre l'organigramme principale du programme de commande.

Si : P1P2=00→ choisir la fréquence maximale du signal PFM de 1KHZ à 20 KHZ par POT2.

Si : P1P2=01→ choisir la fréquence du signal PWM de 1KHZ à 20 KHZ par POT2.

Si : P1P2=10→ générer le signal PFM pour Fmax choisie.

Si : P1P2=11→ générer le signal PWM pour F choisie.

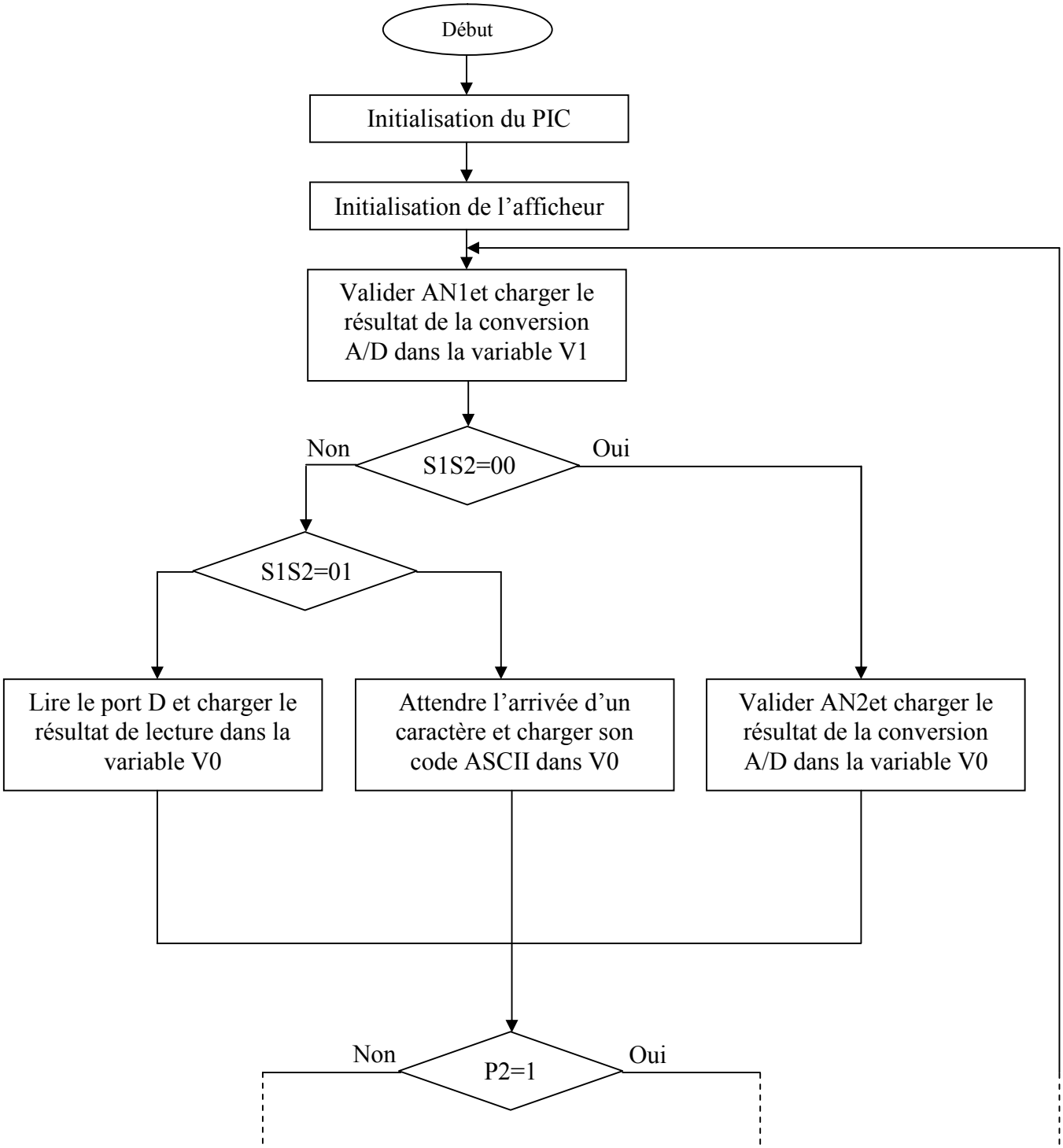
Si : S1S2=00→ Utiliser l'entrée analogique.

Si : S1S2=01→ Utiliser l'entrée numérique.

Si : S1S2=10/11→ Utiliser le PC.

Sur cette base on va construire notre programme de commande, en commence par mettre le schéma de son organigramme.

III.4.1 Organigramme principal :



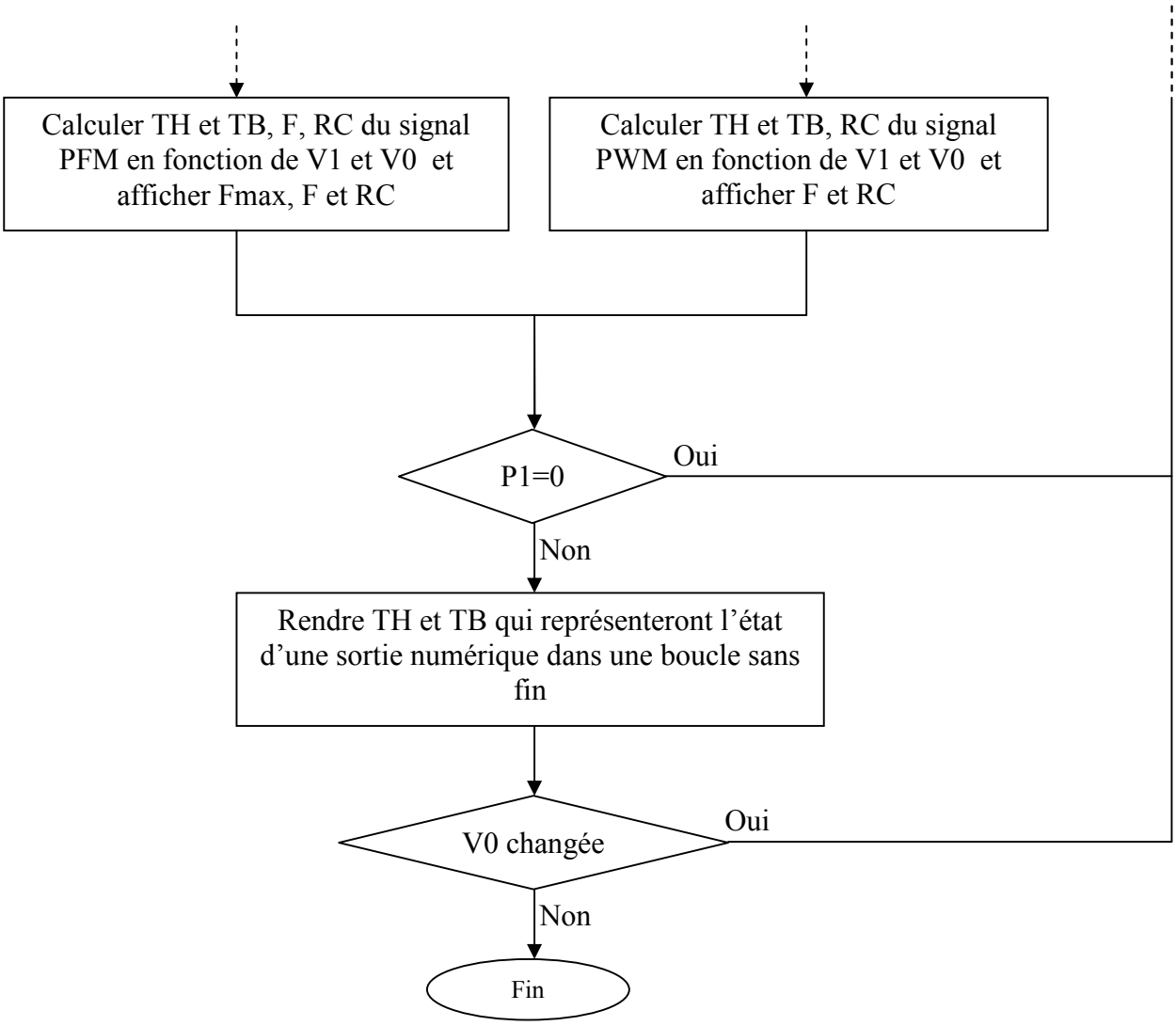


Figure III.3 : Organigramme principal

III.4.2.Interprétation du Programme de commande :

A la mise sous tension le microcontrôleur procède à l'initialisation des registres et l'afficheur, cette opération est déclenchée par les instructions de l'entête suivantes :

```
#include<16F877A.h>
#fuses HS, NOWDT, NOPROTECT, LVP, PUT, NOWRT
#device ADC=8
#use delay (clock=20000000)
#use RS232 (baud=9600, PARITY=N, xmit=PIN_C6, rcv=PIN_C7, BITS=8)
#define use_portb_lcd TRUE
#include<lcd.c>
```

Où on inclut les fichiers et les configurations nécessaires au fonctionnement du programme :

-fichier définissant du PIC à utiliser.
- la configuration de l'état des fusibles du PIC.
-La configuration du convertisseur ADC.
-La définition de la fréquence d'horloge.
- La configuration du port RS232.
-La configuration du port B comme port destiné à l'affichage.
-Le fichier pilote de l'afficheur LCD.

Puis il s'oriente à la sélection de l'entrée analogique AN1 la lecture du résultat de la conversion AD de l'entrée AN1 qui vient du potentiomètre POT2 après la configuration de la fréquence d'échantillonnage, par les instructions suivantes :

```
set_adc_channel(1);
 delay_ms(2);
 V1=read_adc();// V0 est toujours entre 0 et 255
```

Au même temps on n'oublie pas d'envoyer V1 par le PIC à travers le port série du PC car l'interface graphique à besoin à V1 pour faire des calculs et afficher les différents paramètres du signal, ça se fait par l'instruction suivante :

```
Putchar(V1);
```

Après il cherche l'entrée validée (analogique/numérique/rs232).

Si l'entée analogique est validée, c'est par l'instruction suivante que la lecture du résultat de conversion AD de l'entrée analogique AN0 se fait :

set_adc_channel(0);

delay_ms(2);

V0=read_adc();// V0 est toujours entre 0 et 255

Si l'entée numérique est validée, c'est par l'instruction suivante que la lecture du port D se fait :

V0=input_D();// V0 est toujours entre 0 et 255

Si l'entée série rs232 est validée, c'est par l'instruction suivante que la lecture du caractère survenant se fait:

V0=getchar();// V0 est toujours entre 0 et 255

Après le PIC utilise ces deux valeurs V0 et V1 pour calculer TH, TB, Fmax, F, et RC du signal PFM ou TH, TB, F, et RC du signal PWM selon le choix.

Fmax, F, RC du signal PFM ou F, RC du signal PWM sont pout être affichés sur l'afficheur LCD 2x16, au biais de l'instruction suivante :

Printf()

Les deux résultats de calcul TH et TB sont utilisé successivement par les deux instructions

Suivantes qui eux même utilisées par une boucle infinie.

output_high(pin_x);

output_low(pin_x)

pin_x: c'est la ligne où on trouve notre signal PWM ou PFM.

La sortie de la boucle infinie ne se fait que par une condition de test qui compare l'ancienne valeur de V0 qui se mesure chaque fois une période du signal à été produit avec l'ancienne valeur , s'il y aune inégalité le programme revient à un point du programme où il peut refaire tout les calculs et Ainsi afficher la nouveaux paramètre du signal.

Remarque :

La condition de test toujours s'ajoute à l'état bas du signal, donc on doit vieillir à la soustraire de l'état bas du signal sinon ça va donner des faux mesures surtout quand on parle des valeurs extrêmes de quelque paramètre du signal.

IV.1.Schéma électrique du montage :

A l'aide du logiciel ISIS Proteus, logiciel de conception et simulation des circuits électronique, j'ai conçu le schéma électrique du montage suivant :

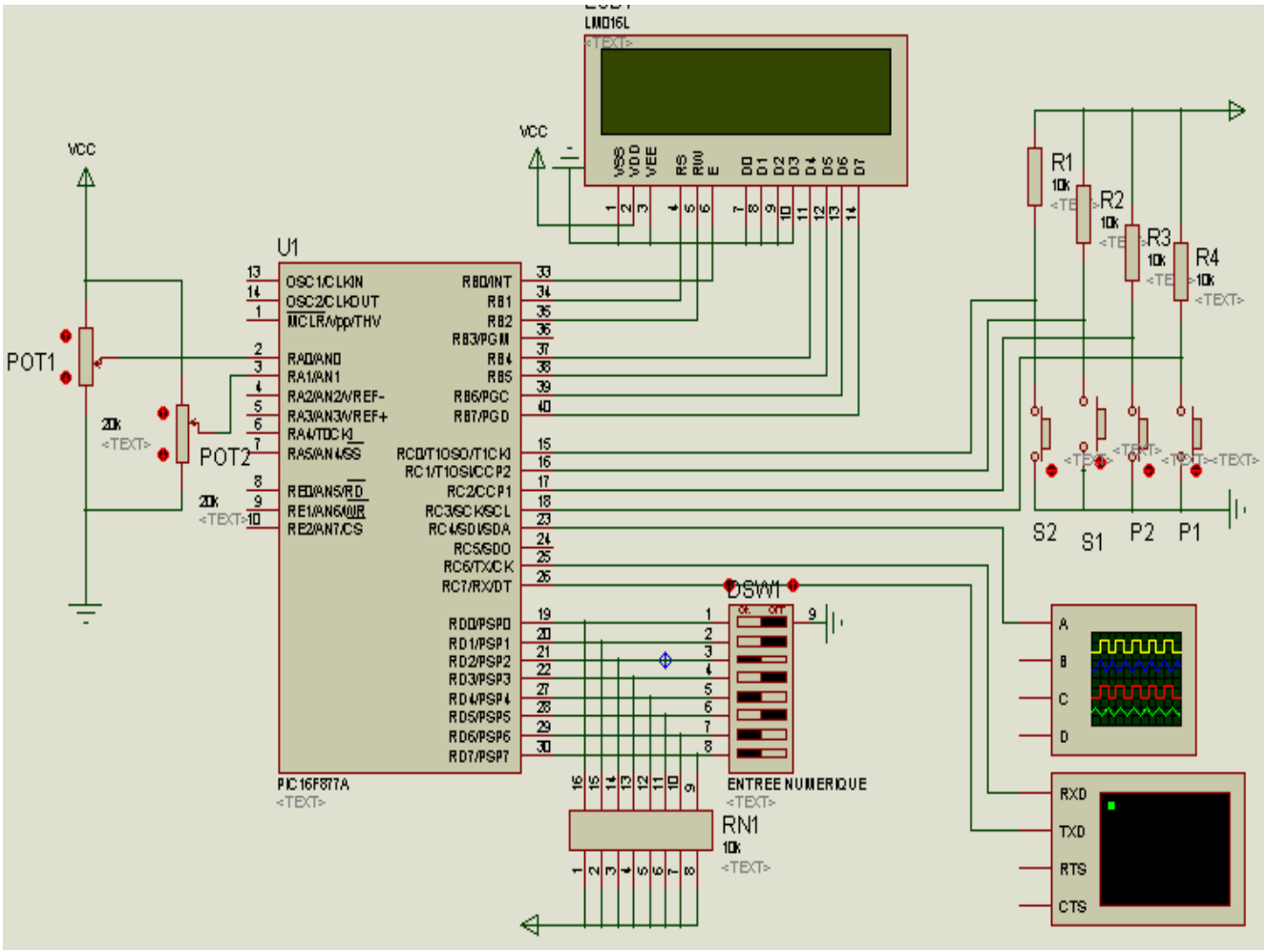


Figure IV.1: schéma électrique du montage.

Ce circuit ne contient que les éléments nécessaires pour faire réussir la simulation car beaucoup d'entrée surtout du PIC son configurés dans le fichier.hex qui contient le programme à exécuter par le PIC, donc on se contente du minimum pour ne pas complique le schéma et le rendre lisible, les éléments qui figurent dans le schéma son tous liés au PIC qui se présente comme la pièce maitresse du circuit, et pour mieux clarifier notre schéma et faciliter la compréhension de ce qui viendra par la suite.

- L'entrée analogique (manuelle et externe) représentée par le potentiomètre POT1 est connectée à la broche AN0.
- Le potentiomètre de choix des fréquences POT2 est connecté à la broche AN1.
- L'entée numérique est connectée au port D.
- L'interrupteur P1 de validation du signal est lié à la broche C3.
- L'interrupteur P2 de choix du signal est liés à la broche C2.
- Les interrupteurs S1et de sélection de l'entée sont liés successivement aux broches C1 et C0.
- L'afficheur est connecté au port B.
- La pin de transmission du terminal virtuel est connecté à la broche C7 de réception du PIC.
- La pin de réception du terminal virtuel est connecté à la broche C6 de transmission du PIC.
- la broche C4 est La sortie du signal qui connecté à l'oscilloscope pour des fins de simulation.

IV.2.La simulation de différents mode de fonctionnent:

Dans cette partie le fonctionnement de l'unité de commande sera mis en évidence en utilisant comme moyen le logiciel de simulation ISIS Proteus qui reflète le fonctionnement réel du circuit. Par des images de simulation qu'on essaiera de faire la démonstration

IV.2.1.Le choix des fréquences:

a) Pour le signal PWM:

Pour choisir al fréquence *F* du signal PWM on doit mettre :

- P1=0 et P2=0, après on choisi la fréquence par POT2

La sortie du signal sera mise au niveau bas (voir Figure IV.2).

b) Pour le signal PWM:

Pour choisir al fréquence *Fmax* du signal PFM on doit mettre :

- P1=0 et P2=1, après on choisi la fréquence par POT2.

La sortie du signal sera mise au niveau bas (voir Figure IV.3).

Dans les images dans la page suivante, il faut bien voir que le curseur sur POT1 et POT2 (c'est l'entrée analogique qu'est sélectionnée) sont dans les mêmes positions et que seule P2 qui a changé d'état.

Remarque: le niveau logique haut est représenté dans les images de simulation par des carreaux rouges et le niveau bas représenté par des carreaux bleus.

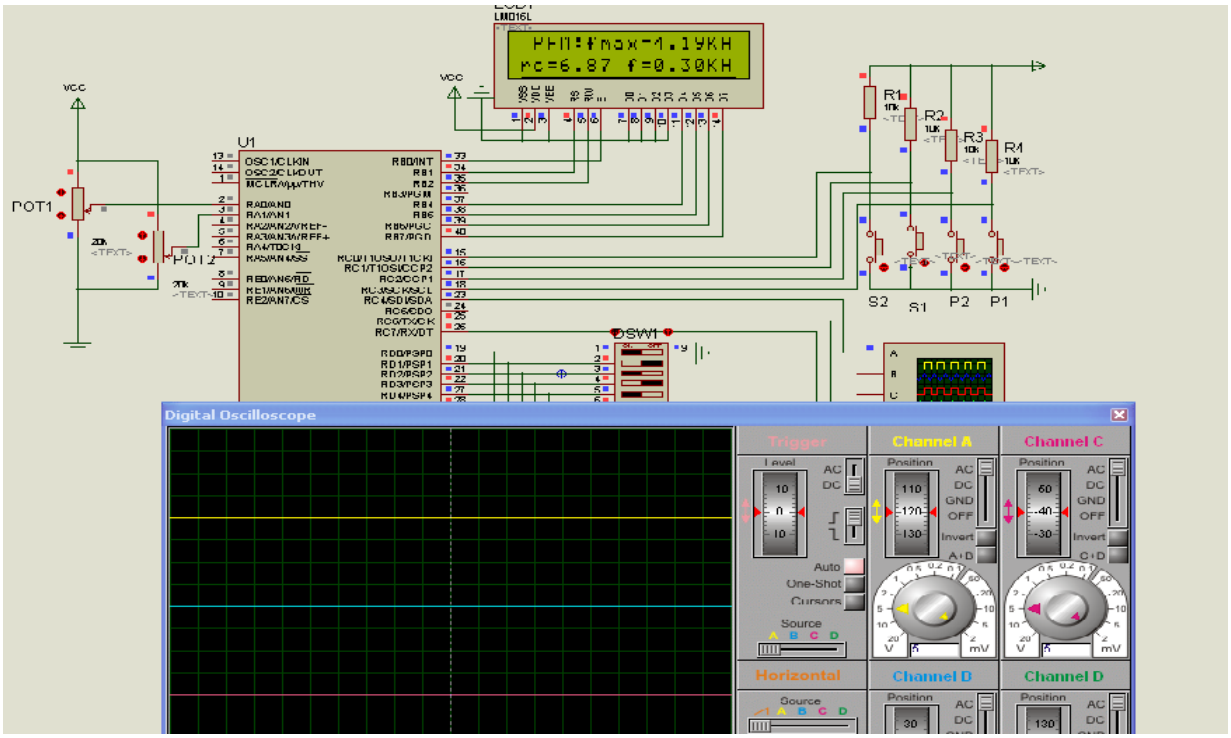


Figure IV.2: choix de Fmax du signal PFM

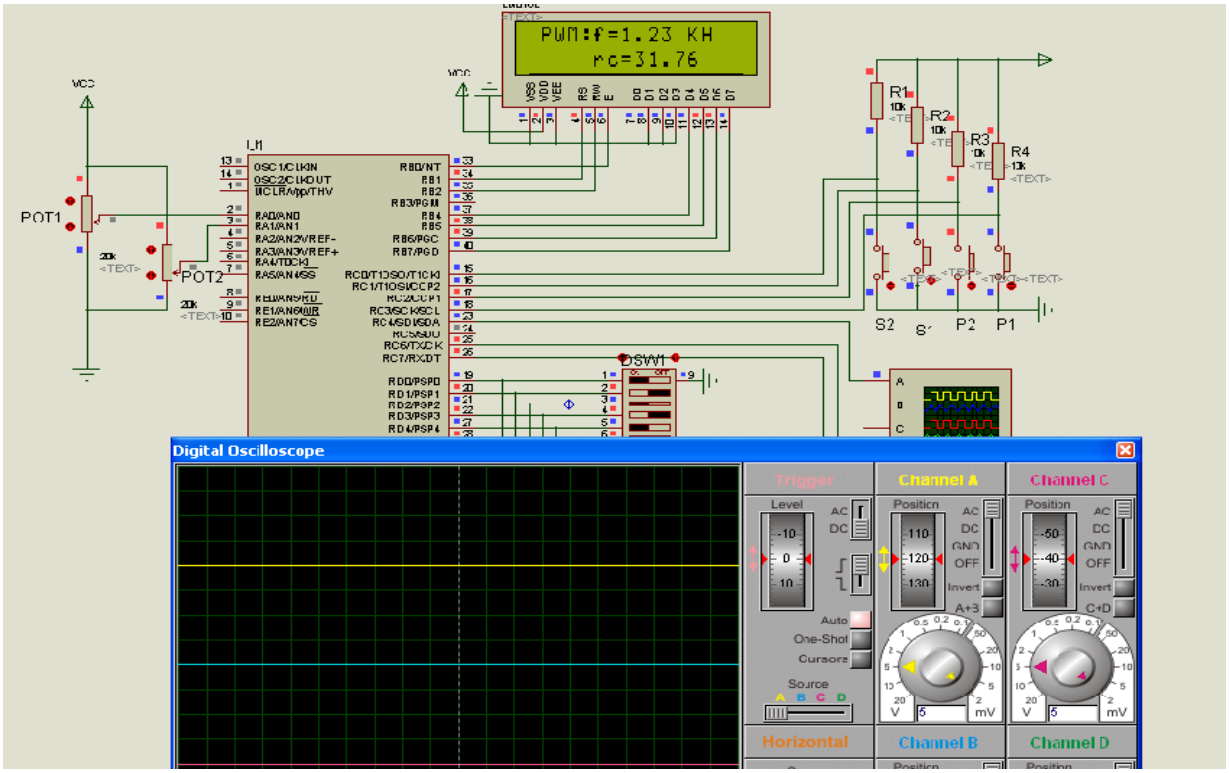


Figure IV.3 : choix de F du signal PWM.

Maintenant on varie change la position du curseur du POT2 pour voir comment Fmax et F changent.

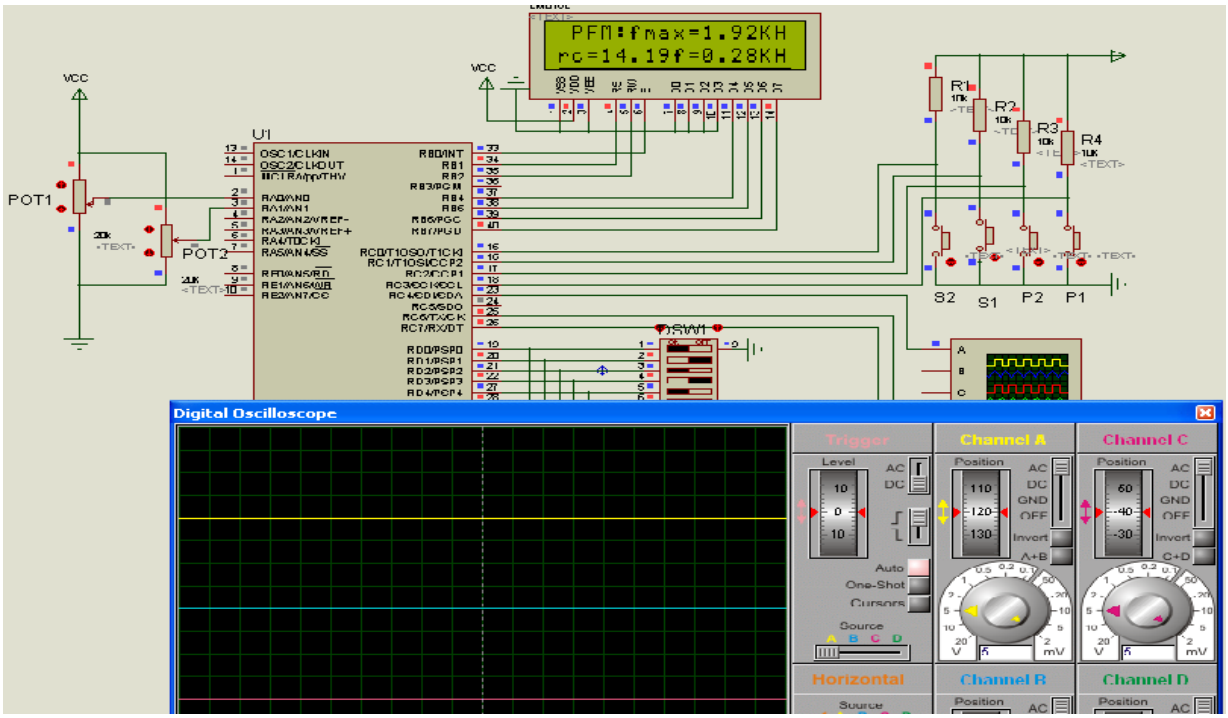


Figure IV.4 : choix d'une autre Fmax du signal PFM

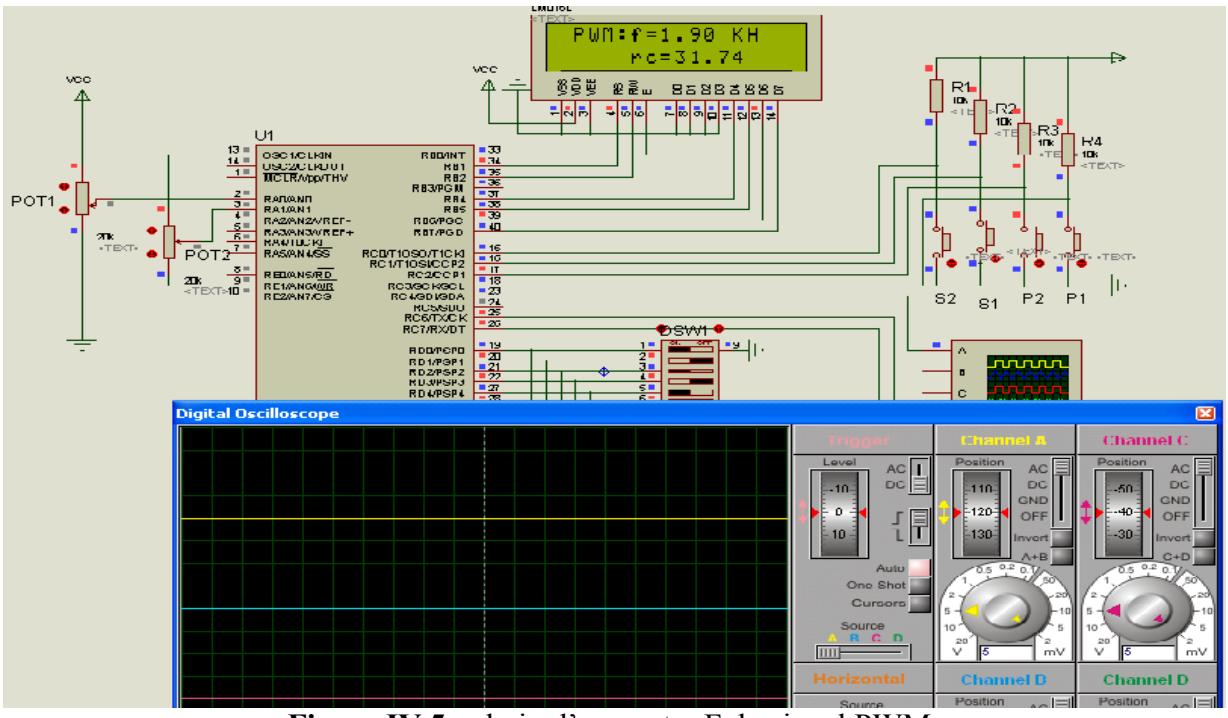


Figure IV.5 : choix d'une autre F du signal PWM

IV.2.2.La génération du signal:

Pour générer le signal il suffit de mettre : P1=1.

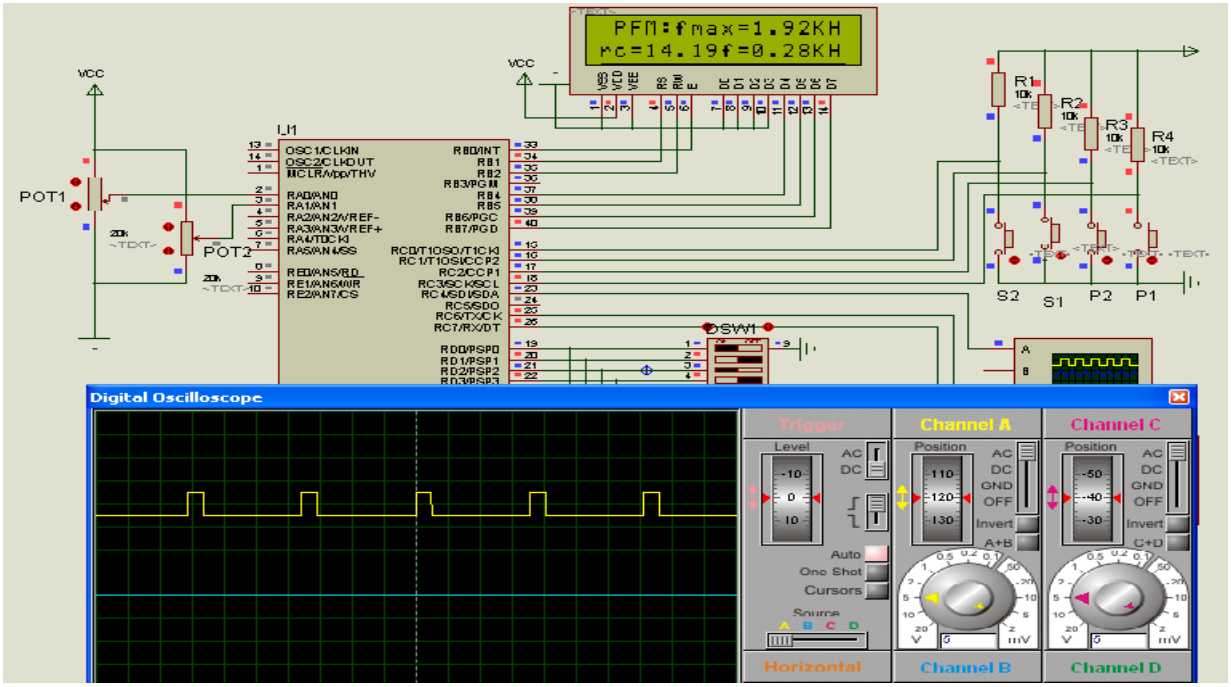


Figure IV.6 : validation du signal PFM

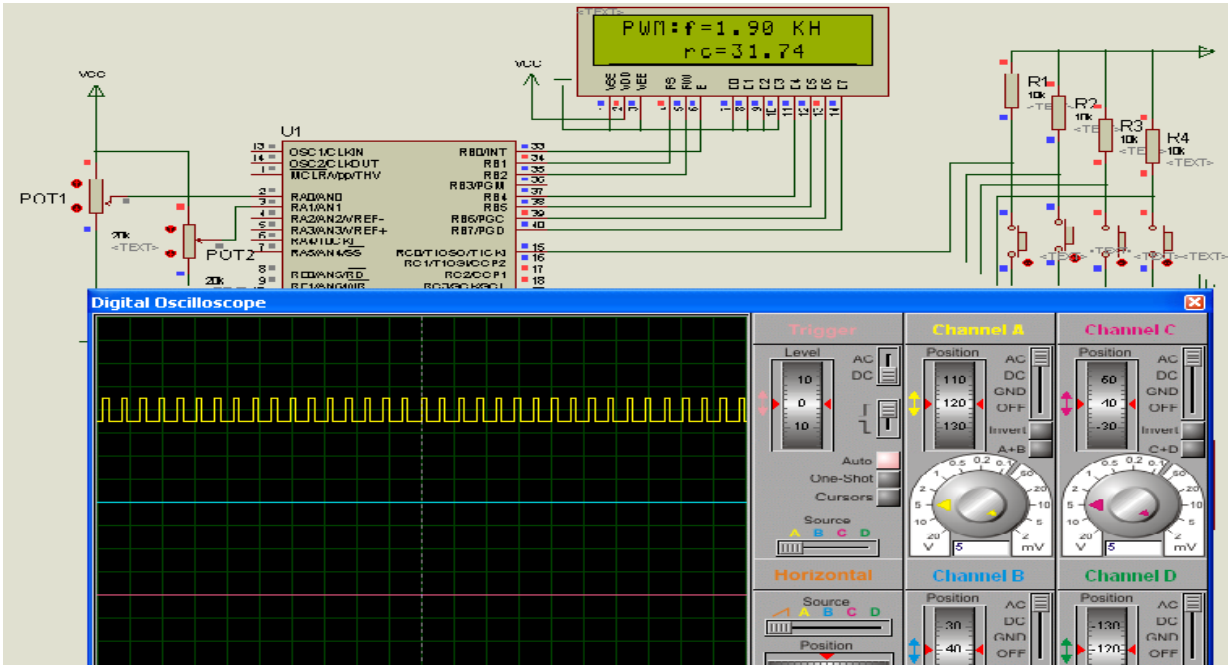


Figure IV.7 : validation du signal PWM.

IV.2.3.Le choix de l'entrée :

Le choix de l'entrée s'effectue de la même façon pour le deux signaux PWM ou PFM, tel que chaque position des interrupteur S1 et S2 est la validation d'une entrée.

-S1=0 et S2=0, validation de l'entrée analogique.

-S1=0 et S2=1, validation de l'entrée numérique.

- S1=1 et S2=0 ou 1, validation de l'entrée série RS232.

En gardant la même fréquence Fmax du signal PFM et F du signal PWM, on utilise toutes les entrée existants pour varie le rapport cyclique du signal PWM ou la fréquence du signal PWM.

a) L'entée analogique :

Il suffit de mettre S1=0, S2=0, P1=1, P2=0, et en variant POT1 pour avoir un rapport cyclique du signal PFM qui varie de 5% à 95%, qui correspond à une fréquence qui varie de Fmax à 0.1Kh.

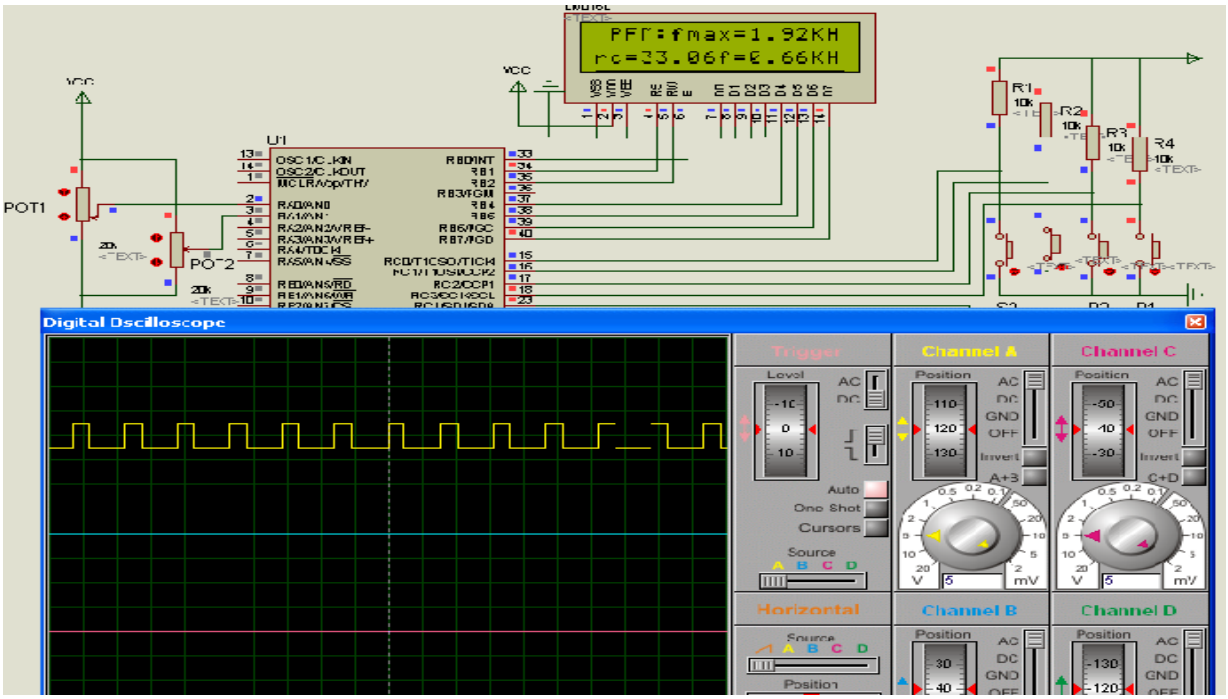


Figure IV.8 : l'utilisation de l'entrée analogique pour varie F (RC) du signal PFM.

Il suffit de mettre S1=0, S2=0, P1=1, P2=1, et en variant POT1 pour avoir un rapport cyclique du signal PWM qui varie de 5% à 95%.

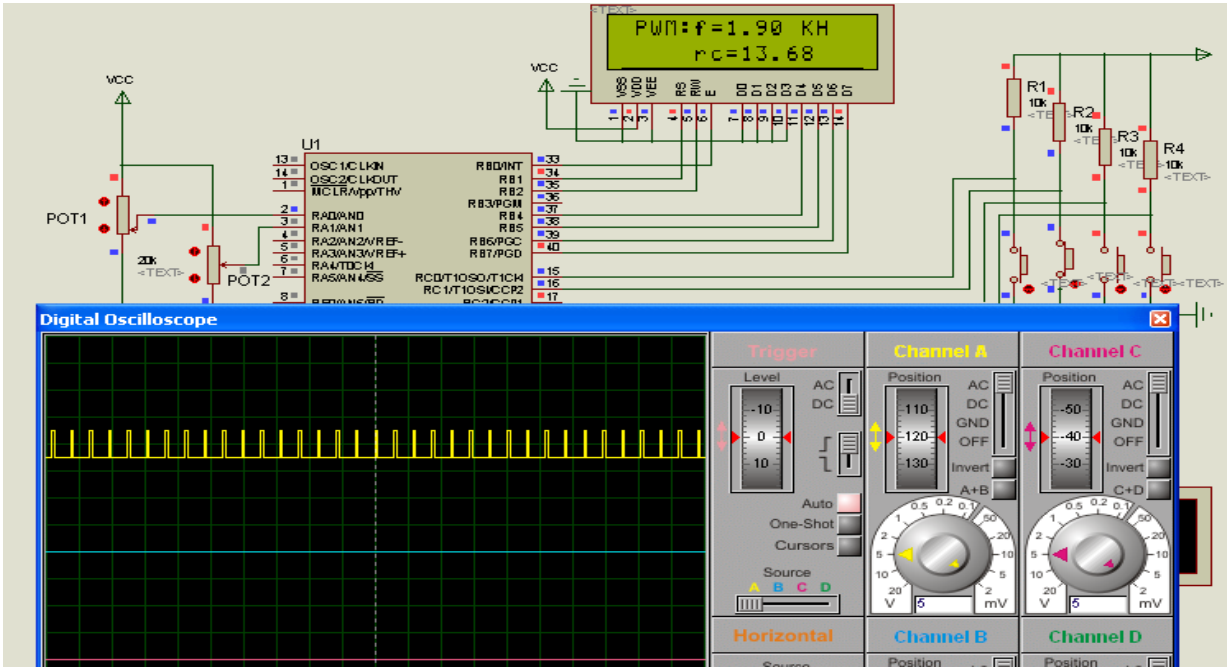


Figure IV.9 : l'utilisation de l'entrée analogique pour varie RC du signal PWM.

b) L'entrée numérique :

Il suffit de mettre S1=0, S2=1, P1=1, P2=0, et en variant l'entrée numérique (SW1 sur l'image) pour avoir un rapport cyclique du signal PM qui varie de 5% à 95%, qui correspond à une fréquence qui varie de Fmax à 0.1Kh.

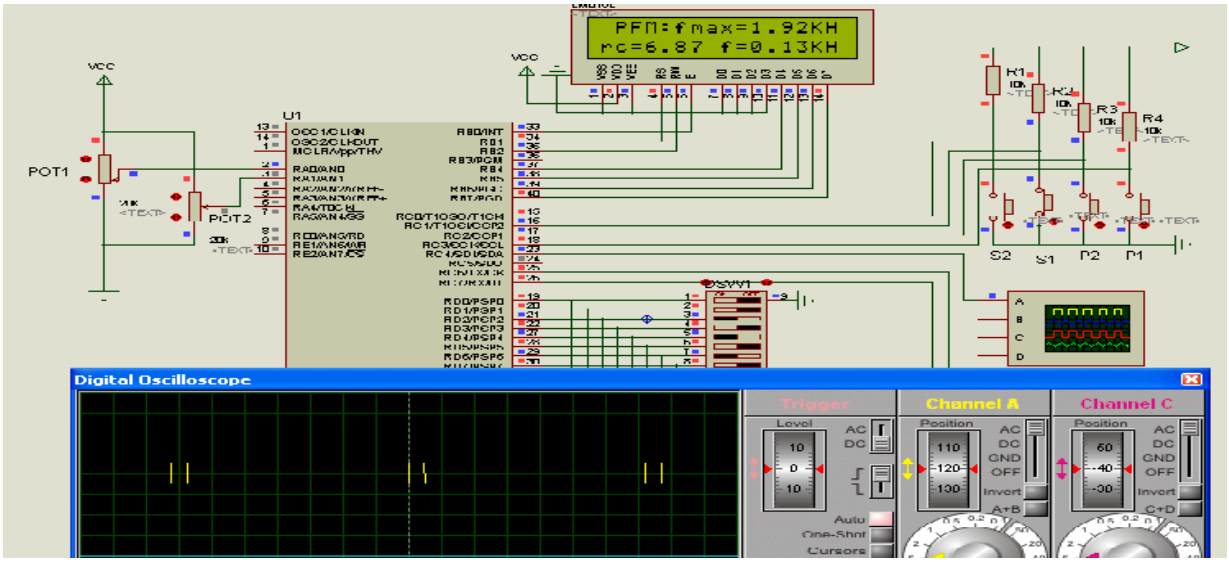


Figure IV.10: l'utilisation de l'entrée numérique pour varie F (RC) du signal PFM.

Il suffit de mettre S1=0, S2=0, P1=1, P2=1, et en variant POT1 pour avoir un rapport cyclique du signal PWM qui varie de 5% à 95%.

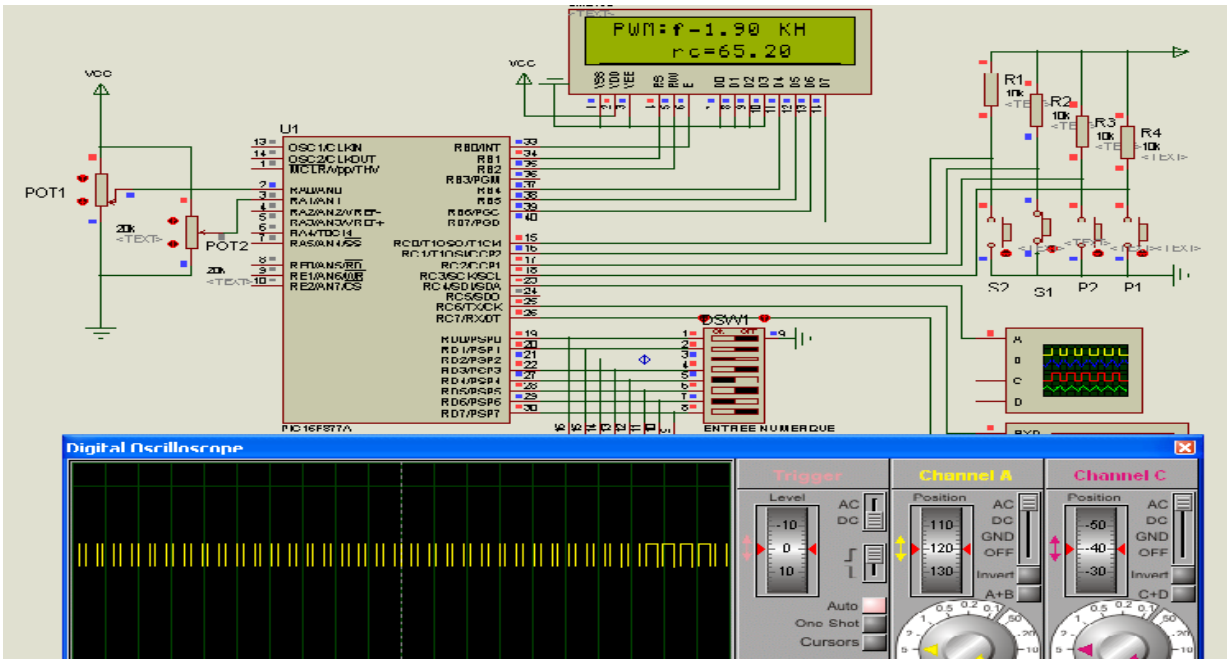


Figure IV.11 : l'utilisation de l'entrée numérique pour varie RC du signal PWM.

c) L'entrée série RS232 :

Il suffit de mettre S1=1, S2=1, P1=1, P2=0, et en appuyant sur n'importe quelle touche du clavier du PC, le terminal virtuel envoie un caractère sur l'entée série du PIC, ce caractère correspond à un rapport cyclique. Donc on peut avoir un rapport cyclique du signal PFM qui varie de 5% à 95%, qui correspond à une fréquence qui varie de Fmax à 0.1Kh.

Le PIC lui-même envoie un caractère qu'on peut voir sur la fenêtre du terminal, ce caractère qui est du à la conversion AD de l'entrée AN1 et qui est l'image de la fréquence Fmax.

Dans le cas pratique il est pour être récupérer par l'interface de contrôle afin d'y afficher les différents paramètres du signal et qui sont les même affichés sur l'afficheur LCD.

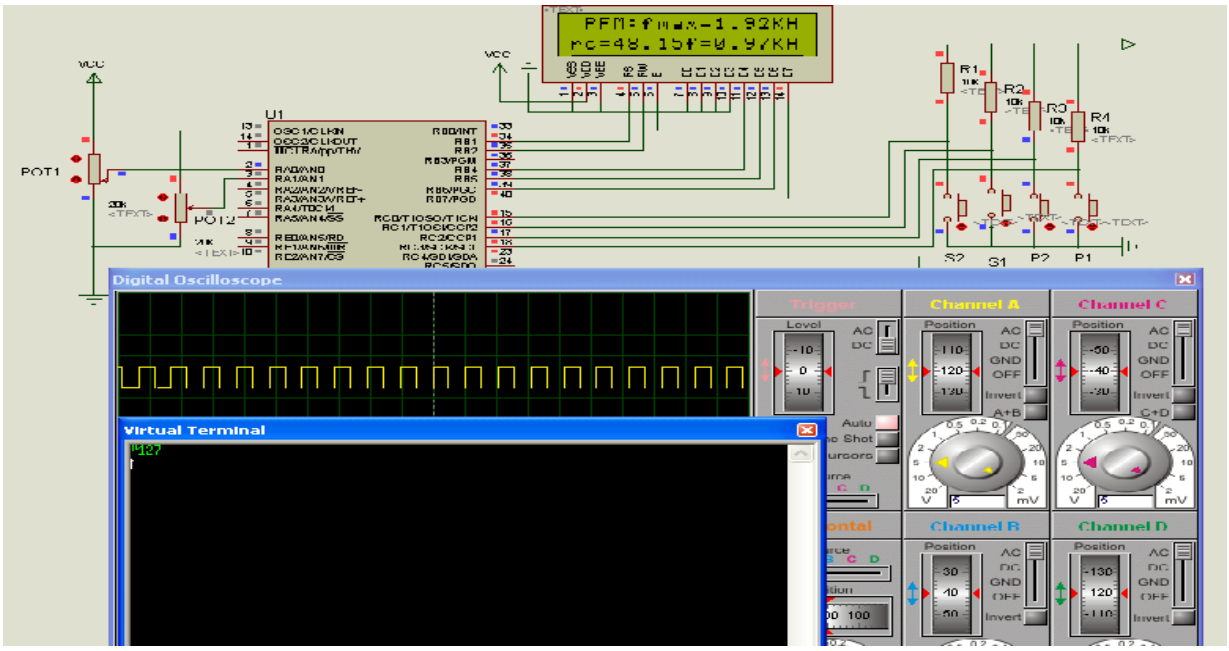


Figure IV.11: l'utilisation de l'entrée série pour varie F (RC) du signal PFM.

Il suffit de mettre S1=1, S2=1, P1=1, P2=1, et en appuyant sur n'importe quelle touche du clavier du PC, le terminal virtuel envoie un caractère sur l'entée série du PIC, ce caractère correspond à un rapport cyclique. Donc on peut avoir un rapport cyclique du signal PWM qui varie de 5% à 95%.

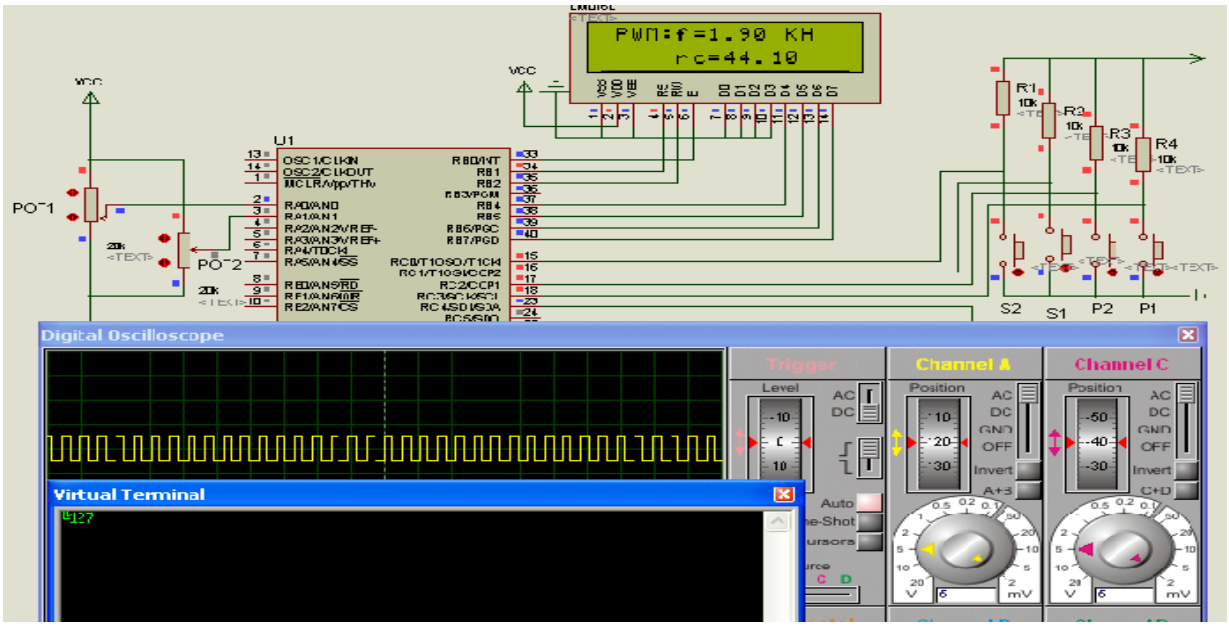


Figure IV.11: l'utilisation de l'entrée série pour varie F (RC) du signal PFM.

IV.3.Réalisation pratique :

La réalisation pratique diffère dans sa philosophie de la simulation de ce qu'elle demande comme esprit réaliste et pratique où on ne doit pas se baigner dans le monde virtuel de la simulation qui peut nous tromper lors de la conception, car la plus part des composant disponible dans la bibliothèque du simulateur ISIS Proteus ne rassemble pas dans leurs repartions de broches à celles des composants réel, ce dernier est conçu pour faciliter la tâche de simuler le fonctionnement d'un circuit électrique, où on trouve des configurations, des message d'erreur, des indicateurs… etc.

La conception d'un circuit électrique sur le simulateur obéit à se qui et donné par les images des composants de sa bibliothèque, et la réalisation pratique se base sur les datasheets des composants donnés par le constructeur et les détails qui viennent avec, l'oublie d'un seul contact mettra en péril le fonctionnement tout ou une partie de notre circuit.

IV.3.1.La mise en œuvre :

De l'idée au circuit-imprimé fini, les étapes de fabrication sont les suivantes:

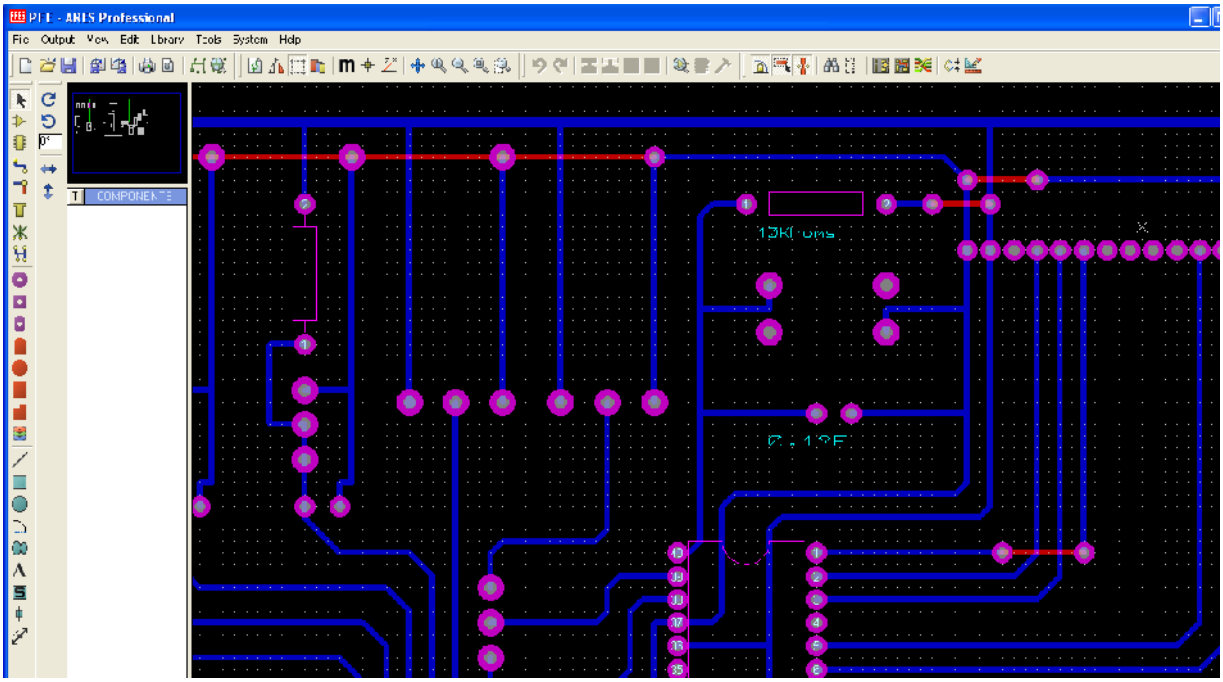
1. Dessin du schéma.
2. Dessin du circuit-imprimé.
3. Impression du masque.
4. Développement.
5. Finition et perçage .
6. Montage des composants.

1. Dessin du schéma :

La première étape est de mettre au priori le schéma. Bien sûr on peut se contenter d'un gribouillage sur papier mais le schéma reste important car, en le dessinant, on s'en imprègne, ce qui est le meilleur moyen pour imaginer ensuite la disposition composants sur le circuit-imprimé. Autre avantage, disposer ensuite d'un schéma propre et publiable car, tant qu'à développer, autant en faire profiter les copains en publiant le fruit de son travail. (Voir la figure VI.1).

2. Dessin du circuit-imprimé :

Il faut tout d'abord le dessiner. On pourrait le faire de plusieurs façons mais la plus efficace est d'utiliser un logiciel, comme dans notre cas j'ai utilisé ARES Proteus.



Il permet de réaliser des layouts de circuit à simple et à double-face et on peut dessiner 2 couches au recto et deux couches au verso: cuivre et composants. Il comporte une bibliothèque de composants traditionnels et une pour CMS (Composants à Montage de Surface ou SMD en anglais) mais là aussi on peut développer ses propres macros. Pour de la HF, il permet de dessiner automatiquement le plan de masse autour des pistes, c'est vraiment très pratique.

3. Impression du masque :

La précision de l'impression était le plus grand souci mais il s'est avéré sans fondement à l'usage. Il suffit d'imprimer à l'échelle 1:1 à l'aide d'une imprimante laser sur du film transparent .Ce genre film transparent existe aussi pour les imprimantes à jet d'encre.

En suivant le chemin suivant : *output>print…*

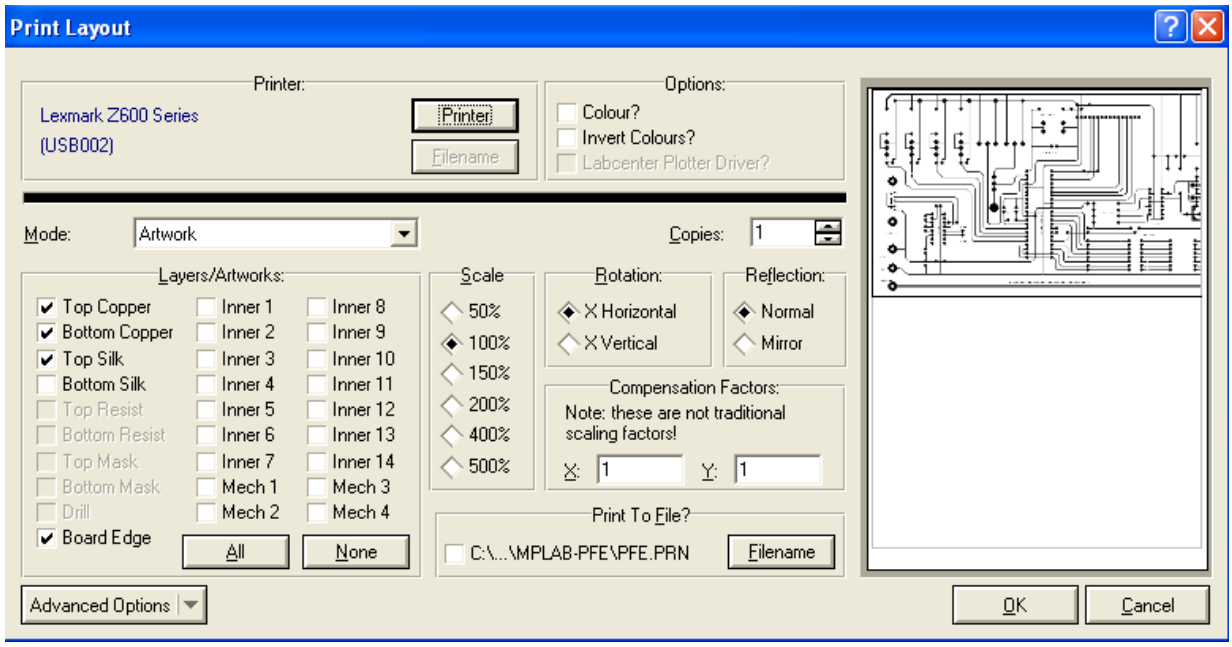


Figure IV.13 : la fenêtre d'options d'impression des circuits imprimés d'ARES

4. Développement :

Le substrat pour circuits-imprimés se compose d'une plaque d'isolant (époxy, fibre de verre, téflon) d'une épaisseur variable (de 0.1 à 1.5 mm) recouvert sur une (simple face) ou deux faces (double-face) d'une couche de cuivre de 35 microns en général. Le choix se fera en fonction de la fréquence, des pertes acceptées (filtres imprimés ou pas) et de la puissance.

Il faut transférer le dessin du circuit-imprimé à la couche photosensible du substrat, La première opération est de retirer le papier noir qui protège la couche photosensible de la lumière afin de mettre cette dernière à nu. Ensuite, il faut apposer le dessin imprimé sur le papier transparent à la plaque de circuit-imprimé, en exposant le côté verre au rayonnement d'un néon ultra-violet, la couche photosensible sera atteinte par le rayonnement à travers les parties transparentes du papier. Par contre, les parties noires ne laisseront rien passer et la couche photosensible restera vierge.

Comme pour une photo, il faut maintenant développer cette "photo" et la durcir afin qu'elle résiste à la gravure chimique et pour cela tremper la plaque dans une solution de soude caustique à 7 pour mille.

En 2-3 minutes dans ce bain à température ambiante, les parties exposées au rayonnement UV deviendront bleu foncé, se dissoudront dans la solution de soude caustique, mettant le cuivre à

nu. Ce sont ces parties-là qui seront ensuite attaquées chimiquement afin de les faire disparaître. Bien rincer à l'eau et sécher avec un sèche-cheveux.

L'étape suivant est la gravure chimique, qui est l'opération la plus longue, plus de 30 minutes. Il suffit d'immerger la plaque pré sensibilisée dans une solution, en principe de perchlorure de fer, produit brun très salissant .Il faudra le chauffer légèrement, disons 30-40 degrés, à l'aide d'un sèche-cheveux par exemple, ou d'une plaque chauffante. La plus grande rapidité d'attaque sera obtenue en déposant délicatement la plaque à attaquer, qui doit être très sèche, sur la surface du liquide, cuivre côté liquide. Le cuivre se dissoudra et descendra au fond du bac. Ce qui provoquerait des îlots non attaqués du plus mauvais effet! Ou alors il faut agiter le bain en permanence, à l'aide d'une pompe soufflant de l'air par exemple (pompe pour aquarium), c'est ce que font les graveurs professionnels.

5. Finition et perçage :

Tout d'abord enlever les reste de laque photosensible à l'aide d'un chiffon et d'acétone. Ensuite, donner un coup de lime sur les tranches du circuit afin de les rendre propres et nettes.
Finalement, poncer les couches de cuivre à l'aide d'une brique abrasive Scotch, afin de rendre ce dernier brillant et bien soudable.
Le perçage des trous s'effectue à l'aide d'une perceuse rapide (5-10'000 tours/min de préférence mais pas obligatoirement) et de forets en métal dur. Ce matériau est obligatoire car la fibre de verre use très vite une mèche en acier normal.

6. Montage des composants :

Le montage des composants vient comme la dernière opération à faire, où chaque composant doit être placé dans son emplacement, en utilisant un fer à souder et l'étain pour souder les pates des composants avec les pistes de cuivre.

Photo de la carte de commande connectée au PC.

Photo de la carte de commande connectée au PC.

IV.4. Applications:

La commande par PWM ou PFM est largement utilisée dans le domaine de l'électronique de puissance, vue son aspect économique et aptitude à la manipulation.

IV.4.1.Applications sur les hacheurs :

Les hacheurs sont les convertisseurs statiques continu-continu permettant de fabriquer une source de tension continue variable à partir d'une source de tension continue fixe.

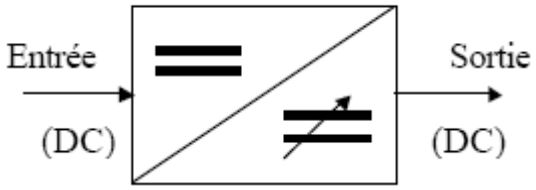


Figure IV.14 : Schéma de principe du hacheur.

Il est évident que le procédé le plus simple pour transformer une tension continue de valeur fixe en une tension continue réglable est le montage en potentiomètre diviseur de tension.

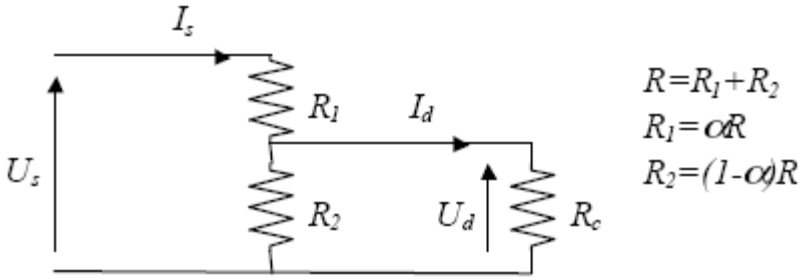


Figure V.15 : Montage potentiométrique.

Le réglage de *a* permet de faire varier la tension disponible aux bornes de la charge *Ud* :

$$U_d = \frac{R_2}{R_1 + R_2} U_s = (1-\alpha) U_s \qquad \text{à vide } (R_c = \infty)$$

- ✓ Pour : $\alpha = 0$, on a : $U_d = U_s$.
- ✓ Pour : $\alpha = 1$, on a : $U_d = 0$.

Inconvénient : l'inconvénient de ce montage est son rendement médiocre, ce qui s'avère critique pour des applications faisant intervenir des puissances non négligeables.

Le rendement s'écrit : $\eta = \frac{P_d}{P_s} = \frac{R_c . R_2^2}{(R_c + R_2)(R_c + R_1 + R_2)}$.

Ainsi les montages potentiométrique sont utilisés uniquement en électronique de faible puissance. En électronique de puissance, on fera systématiquement appel à des hacheurs.

IV.4.1.1. Principe du hacheur (alimentation à découpage):

On distingue plusieurs types de hacheurs, les deux types de base étant le montage série et le montage parallèle.

Le principe consiste à interrompre périodiquement l'alimentation de la charge par la source. L'interrupteur commandable (transistor, IGBT, triac,…) *I* hache la tension d'alimentation *U*. Après filtrage, on obtient une tension U_d constante.

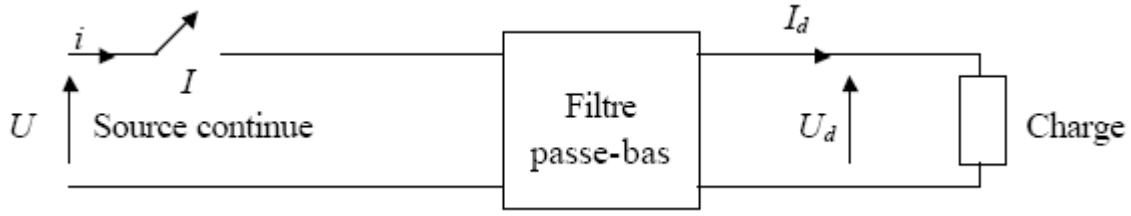


Figure V.16 : Principe du hacheur série.

IV.4.2.2. Hacheur série et parallèle:

Comme on l'a dit en introduction, les hacheurs sont des convertisseurs statiques qui sont alimentés par des sources de tension continue et produisent aux bornes d'une charge une tension unidirectionnelle de valeur moyenne réglable. On peut imaginer un grand nombre de dispositifs électroniques réalisant cette fonction.

On se contentera ici d'indiquer les types de montages les plus utilisés ainsi que quelques applications. Ces montages utiliseront des interrupteurs unidirectionnels statiques.

IV.4.1.2.1.hacheur série :

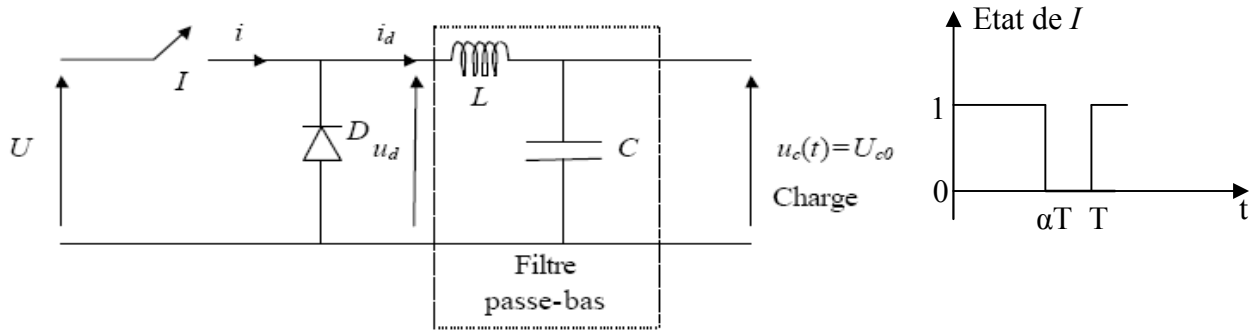


Figure V.17 : Hacheur série avec filtre passe-bas en sortie.

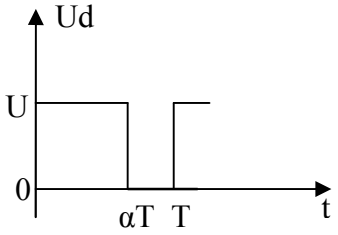


Figure V.18 : La tension Ud(t).

Le montage de la figure V.4, est de loin le plus utilisée en pratique car elle permet un filtrage aisé de la tension $u_d()$ par un filtre passe-bas qui permet d'éliminer les harmoniques élevés de $u_d()$.

La modélisation du système hacheur série a donnée la relation liant la tension de sortie à celle d'entrée U :

$$U_{c0} = \alpha U$$

α : est le rapport cyclique du signal de commande appliqué sur l'interrupteur commandable *I.*

Le contrôle du rapport cyclique α permet de faire varier la tension de sortie qui est la valeur moyenne de $u_d()$ après que le filtrage des harmonies.

¤Technique de commande :

Pour avoir la tension désirée on applique le signal PWM généré par l'unité de commande sur la base de l'IGBT, ce qui donne un signal Ud à l'entrée du filtre passe-bas de la même image que le signal de commande et pour avoir la tension U_{c0}, il suffit de multiplier la valeur du rapport cyclique affichée sur l'afficheur LCD par la tension d'entrée du hacheur.

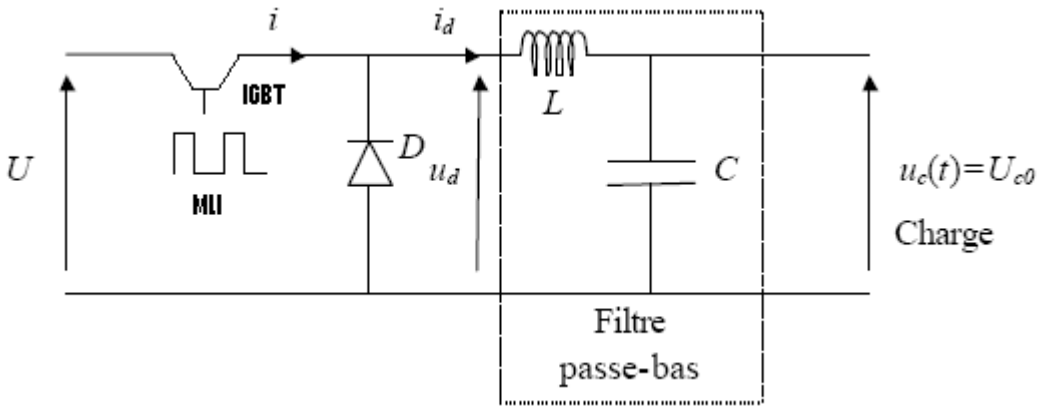


Figure V.19 :.Technique de commande par MLI d'un hacheur série.

IV.4.1.2.2.hacheur parallèle :

Le hacheur parallèle est aussi appelé hacheur survolteur. Ce montage permet de fournir une tension moyenne U_{d0} à partir d'une source de tension continue $U < U_{d0}$.

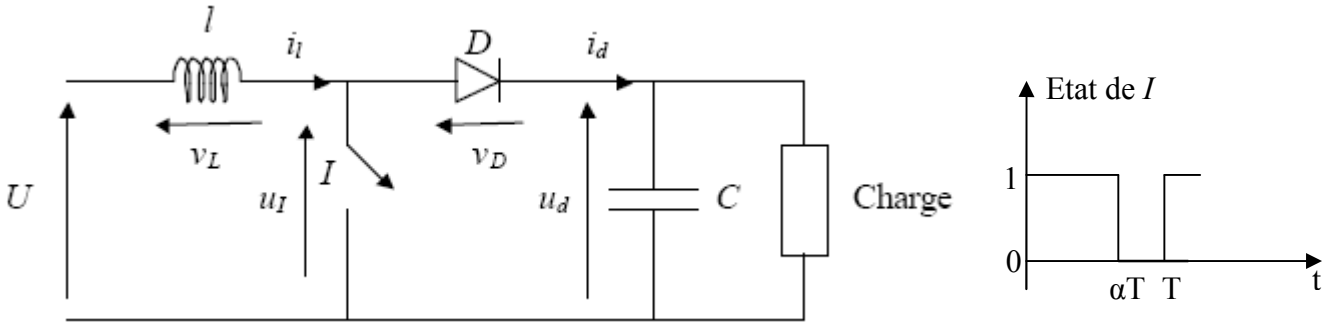


Figure V.20 : Hacheur parallèle.

Le montage de la figure V.7, est de loin le plus utilisée en pratique car elle permet un filtrage aisé de la tension $u_d()$ par un filtre passe-bas qui permet d'éliminer les harmoniques élevés de $u_d()$.

La modélisation du système hacheur parallèle a donnée la relation liant la tension de sortie à celle d'entrée U :

$$U_d = \frac{U}{1-\alpha}$$

α : est le rapport cyclique du signal de commande appliqué sur l'interrupteur commandable *I*.

Le contrôle du rapport cyclique α permet de faire varier la tension de sortie qui est la valeur moyenne de $u_d()$ après que le filtrage des harmonies.

¤Technique de commande :

Pour avoir la tension désirée on applique le signal PWM inversé généré par l'unité de commande sur la base de l'IGBT, et diviser la tension d'alimentation par le rapport cyclique affichée sur l'afficheur LCD.

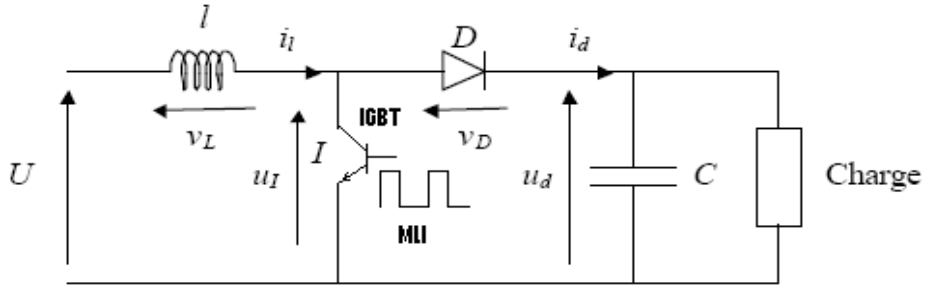


Figure V.21 :.Technique de commande par MLI d'un hacheur série.

IV.4.2.Applications sur les moteurs à courant continu :

IV.4.2.1.équation de fonctionnement :

Le fonctionnement général d'une machine à courant continu (MCC) accouplé à une charge mécanique peut être traduit par les quatre équations :

¤ L'équation mécanique :

$$J\frac{d\Omega}{dt} = C_E - C_R \text{ ……..(1)}$$

Elle relie la vitesse $\Omega(rd)\ s$ et le couple électromagnétique C_E(Nm) produit par la machine, et J et le moment d'inertie de la partie tournante. C_E Représente l'ensemble des couples extérieurs appliqués (moteurs ou résistants), frottements compris.

¤ L'équation électrique :

$$u = E + Ri + L\frac{di}{dt} \text{ ……(2)}$$

Met en relation la tension aux bornes u, le courant absorbé *i* par la machine de f.e.m, E. de résistance interne R et d'inductance L. l'expression (2) écrite ici en convention récepteur, conduit au modèle électrique de la figure suivante :

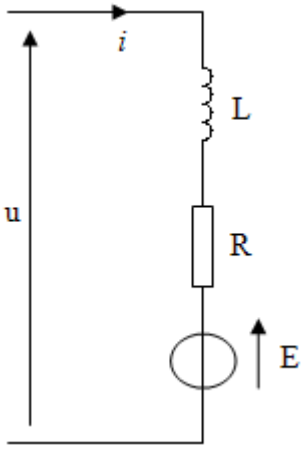


Figure IV.22 : modèle électrique d'une MCC.

Les relations (1) et (2) ne sont pas indépendantes puisqu'on leur adjoint les deux équations électromécaniques :

$C_E = ki$ ……...(3) et $E = k\Omega$ ……..(4)

La grandeur k qui intervient dans les relations (3) et (4) est directement proportionnelle au *flux d'entrefer* de la machine.

Dans le cas du *régime établi*, en dehors des phases d'accélération et de freinage.

La vitesse Ω et constante ($\frac{d\Omega}{dt} = 0$), le courant *i* est périodique (il s'agit généralement) d'un courant continu ondulé), alors ($\frac{L\,di}{dt} = 0$), les équations de fonctionnement se résume à :

$C_E = C_R \ldots(5)$ $\quad \bar{u} = E + R\bar{i} \ldots(6)$ $\quad C_E = ki \ldots(7)$ $\quad E = k\Omega \ldots(8)$

Les équations (5) et (7) montrent que le *courant moyen* absorbé en régime permanent est *imposé* par la charge mécanique de la machine.

Les quatre relations précédentes peuvent être facilement combinées pour donner *l'équation électromécanique* de la MCC *en régime permanent* reliant vitesse et couple :

$$\Omega = \frac{\bar{u}}{k} - \frac{R}{k^2}C \qquad \ldots(9)$$

C est la valeur commune aux couples électromagnétiques C_E et C_R en régime permanent.
L'expression (9) se traduit par une droite de pente négative représentée dans la figure suivante :

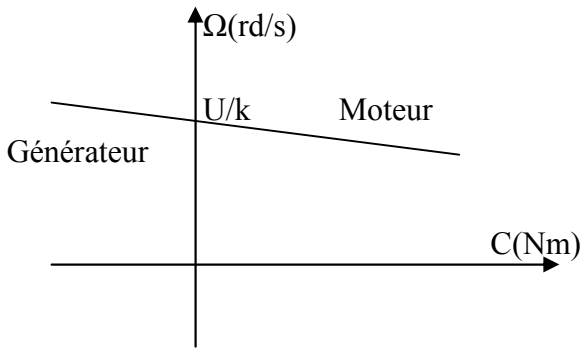


Figure IV.23 : Caractéristique électromécanique d'une MCC.

De l'équation (9) on remarque qu'on peut contrôler la vitesse d'une MCC par la tension $\bar{u}$.

On réalité y a pas mieux qu'utiliser un montage hacheur pour contrôler la tension $\bar{u}$ et par conséquent la vitesse Ω, il y a plusieurs types de hacheurs pour le contrôle de la vitesse d'une MCC qui s'utilisent selon le besoin, on en cite :

¤ Hacheur série.

¤ Hacheur parallèle.

¤ Hacheur réversible en courant (deux quadrants).

¤ Hacheur réversible en courant et tension (quatre quadrants).

IV.4.2.2 hacheur série :

La commande d'une MCC par un hacheur série consiste à mettre la MCC comme charge du hacheur, le montage est montré dans la figure suivante :

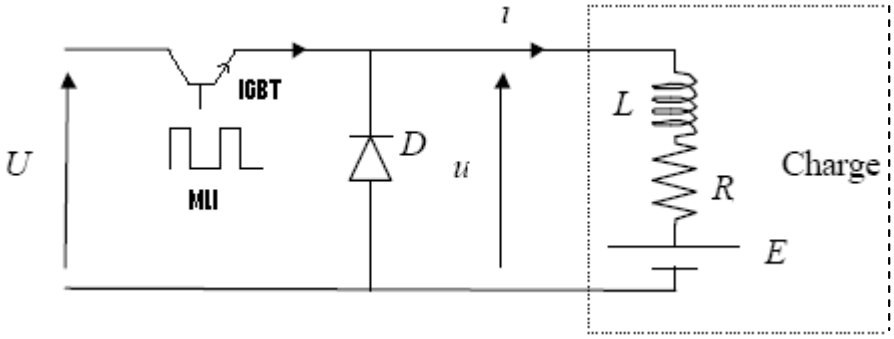


Figure IV.24 : montage hacheur série pour la commande d'une MCC.

C'est le modèle de la machine à courant continu le plus classique. Comportant une résistance R une f.e.m E=kΩ constante en régime établi. L'inductance totale bobine de lissage comprise, notée L, est d'abord considérée comme suffisant pour assurer la continuité de la conduction.

¤ Point de fonctionnement du moteur :

De la relation (9) on a : $\Omega = \frac{\bar{u}}{k} - \frac{R}{k^2} C$

Tel que : $\bar{u}$ est la tension moyenne que le hacheur série délivre à la charge et qui est donnée par la relation suivante pour un hacheur série: $\bar{u} = \alpha U$ ……..(10)

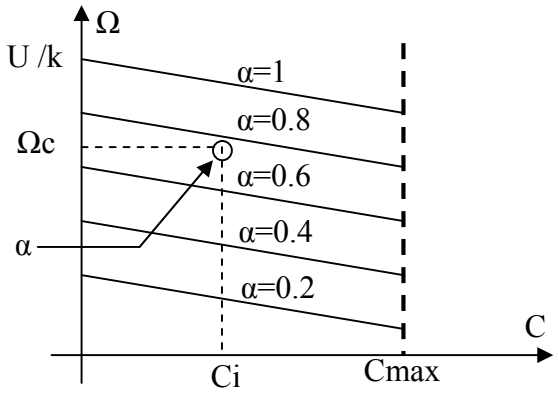
De (9) et (10), on obtient l'équation linéaire de fonctionnement en régime établi.

$$\Omega = \frac{\alpha U}{k} - \frac{R}{k^2} C$$

Les caractéristiques de Ω(C) du moteur forment une famille de droites représentées (Figure V.12) dans le quadrant I, ces caractéristiques montrent que la vitesse de rotation dépend à la fois du rapport cyclique α et du couple résistant Cr appliquée.

¤ Technique de commande :

A partie des allures représentants les caractéristiques de la MCC, on peut choisir la vitesse de rotation du moteur pour un couple déterminé qui correspondent à un rapport cyclique bien détermine qu'on tire de l'allure des caractéristiques Ω(C).



Ωc : Vitesse choisi.

Ci : Couple imposé.

α: rapport cyclique à appliquer.

Figure IV.25 : Caractéristique du mateur (conduction continue).

IV.4.2.3 hacheur parallèle :

Dual du hacheur série, le hacheur parallèle est l'interface entre un dipôle de courant fonctionnant en générateur et un dipôle de tension fonctionnant en récepteur.

En pratique :

- le dipôle est un réseau à courant continu U, qui peut absorber de l'énergie.
- le dipôle de courant peut être une MCC fonctionnant en génératrice, il s'agit souvent d'un équipement en *phase de freinage* (freinage en récupération).

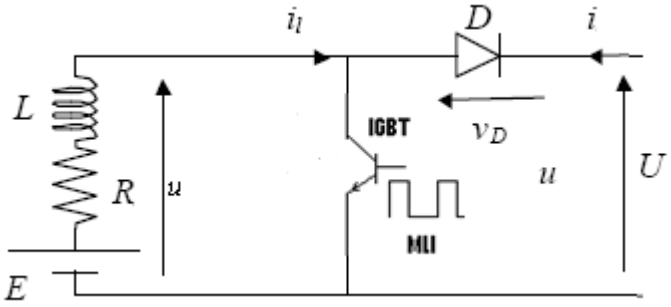


Figure IV.26 : montage hacheur parallèle pour récupération d'énergie MCC.

¤ Point de fonctionnement de la génératrice :

La relation (9) en mode génératrice de la MCC devient:

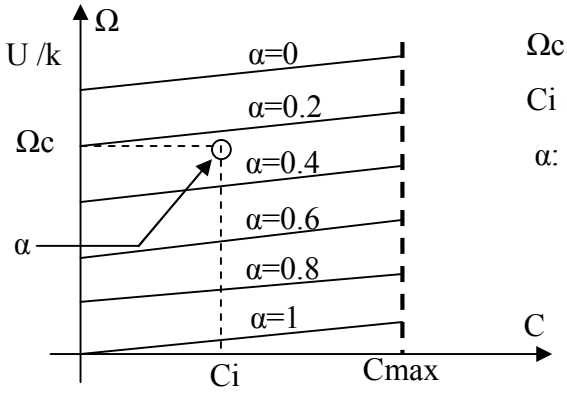
$$\Omega = \frac{\bar{u}}{k} + \frac{R}{k^2} C \quad \text{.....(11)}$$

Tel que : $\bar{u}$ est la tension moyenne au borne de la génératrice, elle donnée par la relation suivante : $\bar{u} = (1-\alpha)U$ .....(12)

De (11) et (12), on obtient l'équation linéaire de fonctionnement en régime établi.

$$\Omega = (1-\alpha)\frac{U}{k} + \frac{R}{k^2}C$$

Les caractéristiques de Ω*(C)* du moteur forment une famille de droites représentées (Figure V.14) dans le quadrant I, ces caractéristiques montrent que la vitesse de rotation dépend à la fois du rapport cyclique α et du couple résistant *C*.



Ωc : Vitesse choisi.

Ci : Couple imposé.

α: rapport cyclique à appliquer.

Figure IV.27 : Caractéristique de la MCC en fonctionnement génératrice.

¤ **Technique de commande:**

A partie des allures représentants les caractéristiques de la génératrice, on peut choisir la vitesse de rotation du moteur pour un couple déterminé qui correspondent à un rapport cyclique bien détermine qu'on tire de l'allure des caractéristiques Ω(C).

IV.4.2.4 Hacheur réversible en courant (fonctionnement « deux quadrants ») :

Le convertisseur représenté figure V.15 est constitué d'une structure de hacheur série juxtaposée à une structure de hacheur parallèle. Le réseau continu est une source de tension réversible en courant. La machine à courant continu sera traitée en convention récepteur

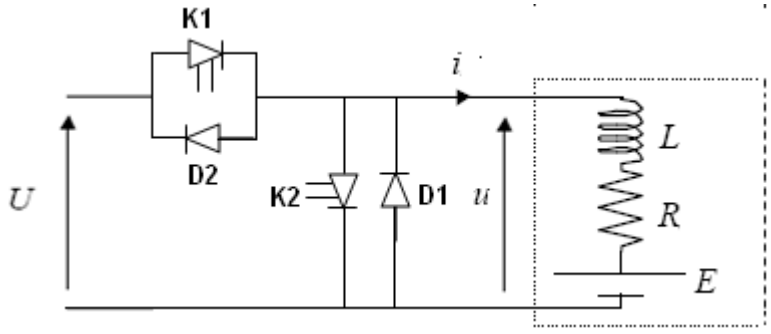


Figure IV.28 : Hacheur réversible en courant.

- En mode normal, l'interrupteur *K1* est commandé périodiquement avec un rapport cyclique α1, le temps où *K2* est incommandable et *D2* est court-circuitée si *K1=1* et bloquée si *D1=1*, donc ils ne participent pas au montage. Le courant *i(t)* à chaque instant. Avec la même hypothèse, la tension moyenne aux bornes du moteur est positive et vaut: $\bar{u} = \alpha_1 U$.

  En faisant intervenir les grandeurs mécaniques liées au moteur :

  $$\Omega = \alpha_1 \frac{U}{k} - \frac{R}{k^2} C \quad \text{.....(13).}$$

  Avec Ω>0 et C>0.

- En mode freinage, c'est *K2* qui est commandé périodiquement avec le rapport cyclique α2, K1 et D1 ne participent pas au fonctionnement. Le courant *i(t)* est négatif et la tension moyenne aux bornes de la génératrice, positive est : $\bar{u} = (1-\alpha_2)U$ .on écrit pour la génératrice :

  $$\Omega = (1-\alpha_2)\frac{U}{k} - \frac{R}{k^2} C \quad \text{.....(14).}$$

  Avec Ω>0 et C<0.

Les équations (13) et (14) sont traduites figures V.16 par deux familles de droites de même pente, respectivement dans les quadrants I(C>0) et II(C<0).

Le système ainsi conçu ne possède qu'un seul sens de marche, mais permet le freinage en récupération.

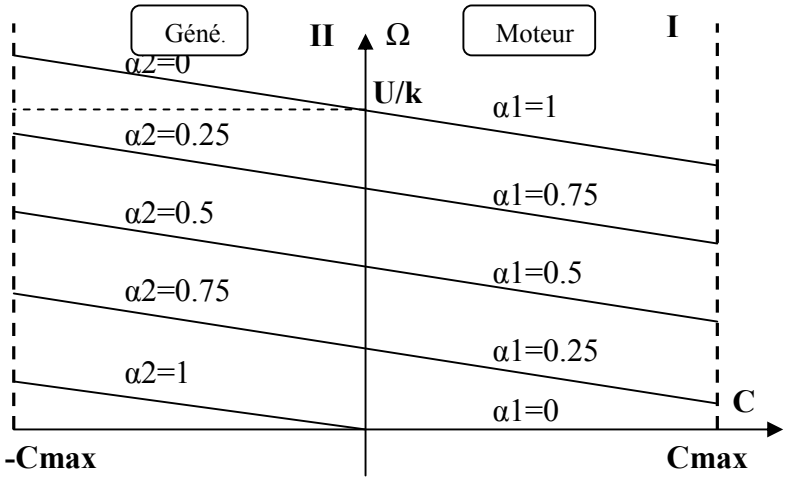


Figure IV.29 : Caractéristique de fonctionnement dans les quadrants I et II.

Pour assurer la continuité du courant dans le moteur quelque soit le couple et ne pas tomber dans le problème de discontinuité, il y a une technique de commande appelée la commande complémentaire, son modèle mathématique est le suivant : $\alpha_1 + \alpha_2 = 1$

C'est-à-dire : Commander *K1* pendant le durée : t : 0 à $\alpha_1 T$.

Commander *K2* pendant le durée : t : 0 à $\alpha_2 T$.

Ça permet de récupérer l'énergie par la source sans avoir de discontinuité de fonctionnement de la MCC.

¤ **Technique de commande:**

La commande complémentaire peut être assurée par notre carte de commande en reliant la sortie non-inversée à l'interrupteur *K1* et la sortie inversée à *K2*.

IV.4.3.Applications à la régulation des processus:

Notre carte de commande peut être utilisée dans des systèmes de régulation qui acceptent ce type de commande par les signaux PWM/PFM, en basant la conception de notre système sur le principe de fonctionnement de la carte de commande, le principe de régulation par MLI est représenté dans la figure suivante :

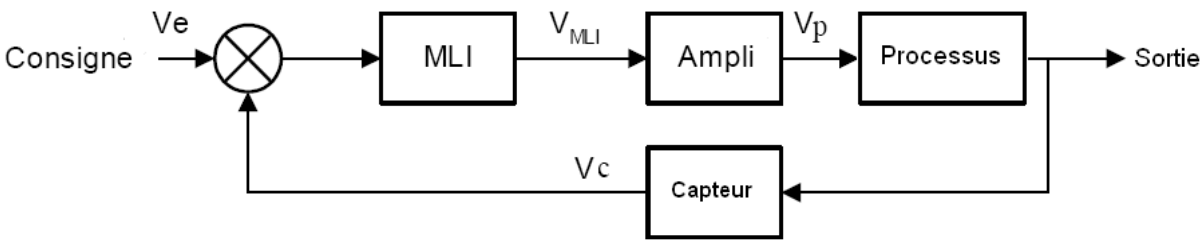


Figure IV.30 : Principe de régulation par PWM/PFM.

Pour mieux clarifier l'utilité de notre carte de commande dans ce type de régulation, on montre par des schémas comment on peut exploiter les différentes entrées de la carte pour la régulation d'un système.

IV.4.3.1 Utilisation de la carte de commande pour la régulation analogique :

Dans ce cas on utilise l'entrée analogique de la carte de commande qui reçoit une tension 0 à 5V qui représente l'image du signal capté par le capteur de la chaine de retour mais adapté au principe de génération d'un signal PWM ou PFM.

Donc après le capteur il faut toujours un circuit de conditionnement et d'adaptation analogique.

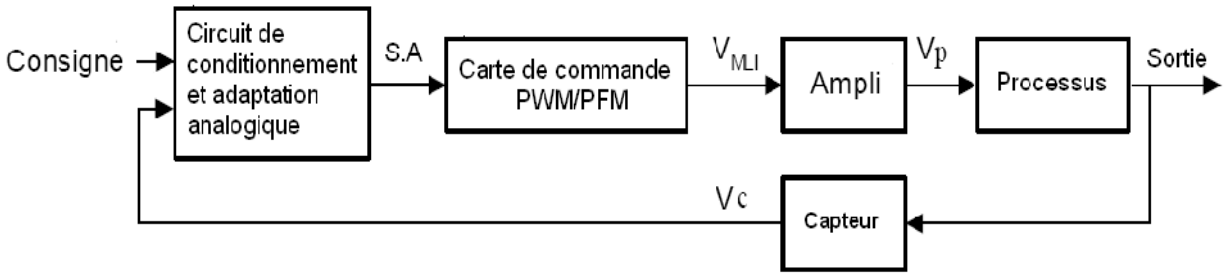


Figure IV.31 : Principe de régulation analogique par PWM/PFM.

IV.4.3.2 Utilisation de la carte de commande pour la régulation numérique :

Dans ce cas on utilise l'entrée numérique de la carte de commande qui reçoit une combinaison de 8 bits 0x00 à 0xFF qui représente l'image du signal capté par le capteur de la chaine de retour mais adapté au principe de génération d'un signal PWM ou PFM.

Donc après le capteur il faut toujours un circuit de conditionnement et d'adaptation analogique et/ ou numérique.

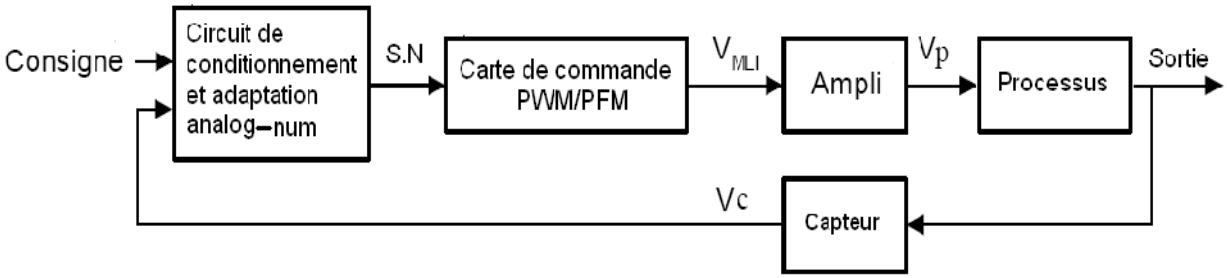


Figure IV.32 : Principe de régulation numérique par PWM/PFM.

IV.4.3.3.Utilisation de la carte de commande pour la régulation par PC :

Dans le cas où on envisage des algorithmes de trainement de signal compliqués on peut se doter d'un PC à lequel on envoi l'information collectée au niveau de la sortie du système, qu'il utilise comme argument d'entrée pour exécute l'algorithme qu'on implémenter dedans, après il envoi l'argument de sortie à la carte de commande via le port série pour qu'elle génère le signal PWM/PFM qu'il faut pour la commande du processus.

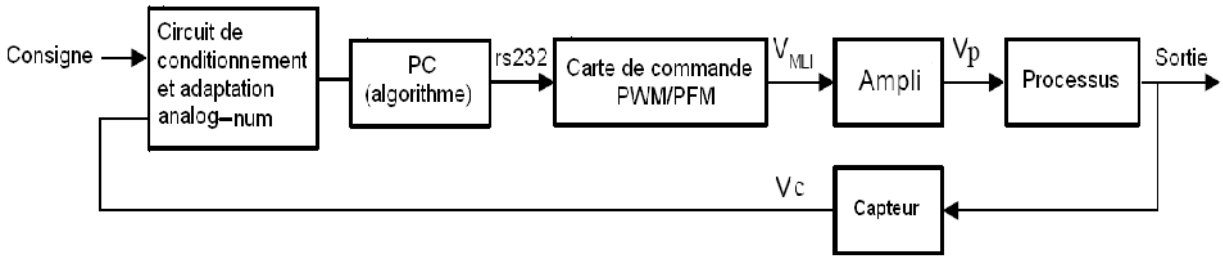


Figure IV.33: Principe de régulation numérique (par PC) par PWM/PFM.

La régulation par cette carte de commande est valable pour des processus de temps de réponse relativement long vue la lente de traitement de donnée par le PIC 16F877, elle devient plus sophistique si on utilise des processeurs plus rapide comme les DSP, ça permettra d'élargir le domaine d'utilisation de nôtre carte de commande.

Conclusion Générale

Ce projet m'a permis d'approfondir mes connaissances dans le domaine de la programmation des PICs, la conception des circuits et la conception des logiciels, et aussi il m'a permis d'avoir la manière dont on peut faire réussir un projet dés qu'il soit idée jusqu'à la mise en œuvre.

Le travail que j'ai effectué est de réaliser une carte de commande universelle qui nous génère un signal PWM ou PFM sur une plage de fréquence qui couvre le domaine fréquentiel de fonctionnement de plusieurs dispositifs fonctionnant à base de ces signaux, elle nous offert aussi plusieurs manière d'accéder aux paramètres de ces signaux. Ce projet a aussi sa partie soft qui se présente sous forme d'un petit logiciel qui permet la communication entre le PC et le PIC.

Cette carte de commande peut être utilisée pour la commande manuelle directe des dispositifs électriques, mais elle a aussi l'aptitude d'être utilisée pour la régulation de certain processus qui ne demande pas beaucoup de performance en matière de rapidité (les systèmes relativement lents).

En fin j'espère que ce projet sera l'objet d'une amélioration et valorisation par les étudiants des promotions futures.

Bibliographie

[1]. Programming 8-bit Microcontrolers in C.	Martin P. Bates.
[2]. An introduction to Programming the Microchip PIC in CCS C.	Nigel Grander.
[3]. Programmation en C des PIC.	Christian Tavernier.
[4]. Visual Basic 6.	Aït Ali.
[5]. Visual Basic.	Byron S. Gottfried.
[6]. Signals and Systems.	Hwei P. Hsu.
[7]. Machine électrique théorie et mise en œuvre.	Philipe Barret.
[8]. Les convertisseurs de l'électronique de puissance.	Francis Labrique& Gay Segier& Robert Baussière.
[9]. Electronique de Puissance	Jacques Laroche.