**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**

**Université Abderrehmane Mira Béjaïa**

**Faculté de Technologie**

**Département Génie Electrique**

**Filière : Electronique**

**Spécialité : Instrumentation**

PROJET DE FIN D'ETUDE

Pour l'obtention du diplôme de

***Master en Instrumentation***

***Thème***

# *Etude et Réalisation d'une Carte de Variateur de Vitesse d'un Moteur à CC par PIC 16F877*

Présenté par :

**ANKI Bassem**

**CHENNIT Lila**

Dirigé par :

**M^r^ HANFOUG Salah**

Membres du jury :

**M^r^ CHARIKH Ahmed**

**M^me^ GHENNAM Souheila**

Présenté le 11/10/2020

Année universitaire 2019/2020

# *Remerciements*

# *Remerciements*

*Nous remercions notre encadreur M^r^ S. HANFOUG pour avoir accepter de diriger notre travail.*

*Nous remercions également les membres du jury M^r^ A. CHARIKH et M^me^ S. GHENNAM d'avoir accepter de juger notre travail.*

# *Résumé*

## *Résumé*

Varier la vitesse d'un moteur est primordiale dans divers domaines. Ce travail présente l'étude et la réalisation d'un variateur de vitesse d'un moteur à courant continu dans les deux sens de rotation.

Une partie de ce travail est consacrée à des rappels sur les moteurs à courant continu, les hacheurs (pont en H) et le microcontrôleur PIC 16F877. Des notions de la méthode de modulation de largeur d'impulsion sont également présentées.

L'étude du variateur de vitesse est faite à l'aide du logiciel de simulation Proteus ISIS en expliquant en détails le fonctionnement de chaque partie du circuit.

Ce travail est couronné par la mise en pratique de l'étude réalisée à travers la conception de la carte électronique du variateur de vitesse.


***Mots clé :*** Variateurs de vitesse, PIC16F877, microcontrôleurs, Moteurs à courant continu, MCC, modulation de largeur d'impulsion, MLI.


## *Abstract*

Varying the speed of an engine is primordial in various fields. This work presents the study and realization of a DC (direct current) motor speed controller in both directions of rotation.

Part of this work is dedicated to reminders about DC motors, choppers (H-bridge) and PIC 16F877 microcontroller. Notions of pulse width modulation are also presented.

The study of speed controller is done using Proteus ISIS simulator software by explainning in detail how each part of the circuit works.

This work is crowned by an implementation of the study carried out through the design of an electronic card of the speed controller.


***Key words :*** Speed controllers, speed variators, PIC16F877, microcontrollers, DC Motors, pulse width modulation, PWM


## *ملخّص*

يعد تغيير سرعة المحرك أمرًا ضروريًا في مختلف المجالات. يقدم هذا العمل دراسة وإنجاز مغيّر السرعة لمحرك التيار المستمر في كلا اتجاهي الدوران.

تم تخصيص جزء من هذا العمل للتذكير بمحركات التيار المستمر، محولات التيار المستمر (جسر H) والميكروكونترولر PIC 16F877. كما يتم عرض مفاهيم طريقة تعديل عرض النبضة.

تتم دراسة مغيّر السرعة باستخدام برنامج محاكاة Proteus ISIS من خلال شرح مفصّل لطريقة تشغيل كل جزء من الدارة.

توج هذا العمل بتنفيذ الدراسة التي أجريت من خلال تصميم البطاقة الإلكترونية لمغيّر السرعة.


**الكلمات المفتاحية :** مغيّر السرعة، ميكروكونترولر، PIC 16F877، محركات التيار المستمر، تعديل عرض النبضة.

# *Sommaire*

## Chapitre III : Etude et Réalisation du circuit du variateur de vitesse

# *Liste des Figures et Tableaux*

# *Liste des figures*

## *Liste des tableaux*

# *Liste des abréviations*

# *Liste des abréviations*

| | |
|---|---|
| ALU | Unité Arithmétique et Logique (*en anglais : Arithmetic and Logic Unit).* |
| A/N | Analogique/Numérique (*en anglais : Analog/Digital).* |
| CC | Courant Continu |
| CCP | Capture, Comparaison, MLI (*en anglais : Capture, Compare, PWM).* |
| CISC | Processeur à Jeu d'Instruction Etendu (*en anglais : Complex Instruction Set Computer).* |
| CPU | Unité Central de Traitement (*en anglais : Central Processing Unit)*. |
| CRT | Tube à Rayons Cathodiques (*en anglais : Cathode-Ray Tube)*. |
| EEPROM | Mémoire à Lecture Seule Effaçable Electriquement et programmable *(en anglais : Electrically- Erasable Programmable Read-Only Memory).* |
| E/S | Entrée/Sortie. |
| f.e.m | Force électromotrice. |
| GPR | Registre à Usage Général (*en anglais : General-Purpose Register*). |
| ISR | Routine de Service d'Interruption *(en anglais : Interrupt Service Routine).* |
| I2C | Circuit Inter-Intégré *(en anglais : Inter-Integrated Circuit).* |
| MCC | Moteur à Courant Continu |
| MSSP | Port Série Synchrone Maître *(en anglais : Master Synchronous Serial Port*). |
| N/A | Numérique/Analogique *(en anglais : Digital/Analog).* |
| PC | Compteur de Programme *(en anglais : Program Counter)*. |
| PCLATH | Compteur de Programme à verrouillage haut (*en anglais :* Program Counter Latch High). |
| PIC | Contrôleur d'interface périphérique *(en anglais : Peripherical Interface Controller).* |
| PSP | Port Esclave Parallèle *(en anglais :* Parallel Slave Port*).* |
| PWM | Modulation de Largeur d'impulsion *(en anglais : Pulse Width* |

*Modulation).*

RAM Mémoire à Accès Aléatoire *(en anglais : Random Access Memory).*

RISC Processeur à Jeu d'Instructions Réduit *(en anglais : Reduced Instruction Set Computer).*

ROM Mémoire à Lecture Seule *(en anglais : Read Only Memory).*

SCI Interface de Communication Série *(en anglais : Serial Communications* Interface*).*

SFR Registres de Fonctions Spéciales *(en anglais : Special Function Registers).*

SPI Interface Périphérique Série *(en anglais : Serial Peripheral Interface)*.

USART Module Emetteur-Récepteur Synchrone Asynchrone Universel *(en anglais : Universal Synchronous and Asynchronous Receiver Transmitter).*

WDT Timer Chien de Garde *(en anglais : Watch Dog Timer).*

# *Introduction Générale*

# *Introduction Générale*

La vitesse représente un paramètre clé dans plusieurs secteurs comme l'industrie ou le domaine spatial. Faire varier la vitesse à notre guise et avoir un contrôle total de la vitesse par un simple geste, de manière instantanée et sans pertes est le but ultime à atteindre. D'où l'importance de cette étude.

Il existe plusieurs types de variateurs de vitesse. Ils se distinguent par leurs topologies, leurs méthodes de commande ainsi que le type de charge à contrôler.

Pour pouvoir varier la vitesse de manière optimale on doit veiller à choisir la bonne méthode de commande. Cette dernière doit être perfectionnée pour minimiser les pertes et maximiser le rendement.

L'objectif de ce travail est l'étude et la conception d'un circuit d'un variateur de vitesse avec la méthode de modulation de largeur d'impulsion en utilisant le PIC 16F877 afin de contrôler la vitesse et le sens de rotation d'un moteur à courant continu.

Le travail sera partagé en trois chapitres. Le premier chapitre traite les généralités sur les moteurs à courant continu (description, caractéristiques, types, ...). Il décrit également le principe d'un hacheur ainsi que la commande d'un moteur à courant continu par un hacheur quatre quadrants.

Le deuxième chapitre expose l'architecture, les caractéristiques et les fonctionnalités de base du microcontrôleur PIC 16F877 dont la méthode de modulation de largeur d'impulsion.

Le troisième et dernier chapitre présente l'étude par le logiciel Proteus ISIS du circuit du variateur de vitesse et la description des différentes parties du circuit. Il présente également les différentes étapes de réalisation ainsi que le résultat final de la carte du variateur de vitesse.

Ce travail sera clôturé par une conclusion générale et une présentation de quelques perspectives pour améliorer le projet réalisé.

# *Chapitre I*

# *Moteurs à Courant Continu et Leurs Commandes*

## I.1 Introduction

Les moteurs à courant continu sont des convertisseurs électromécaniques bidirectionnels. Ils sont présents dans plusieurs domaines comme les systèmes de traction, l'électroménager et d'autres applications industrielles.

Afin de commander le moteur avec précision et rapidité, on fait appel à des convertisseurs statiques (interrupteurs électroniques) tels que les hacheurs.

Dans ce chapitre, nous allons décrire le fonctionnement de base d'un moteur à courant continu et ses caractéristiques. Puis, on présentera le principe de commande de vitesse d'un moteur à courant continu par un hacheur quatre quadrants.

## I.2 Description d'un moteur à courant continu

Un moteur à CC comprend trois parties principales :

### I.2.1 Induit

L'induit est porté par le rotor qui est la partie mobile du moteur. Il est parcouru par un champ crée par l'induit.

L'induit est composé d'un ensemble de bobines identiques réparties uniformément autour d'un noyau cylindrique. Il est monté sur un arbre et tourne entre les pôles de l'inducteur. L'induit constitue donc un ensemble de conducteurs qui coupent le flux magnétique. Les bobines sont disposées de telle façon que leurs deux côtés coupent respectivement le flux provenant d'un pôle nord et d'un pôle sud de l'inducteur **[1]**.

### I.2.2 Inducteur

L'inducteur est situé sur la partie fixe du moteur qui est le stator. Lorsque les bobines de l'inducteur sont alimentées en courant continu, l'inducteur produit un flux magnétique dans la machine (électro-aimant). Cette aimantation peut parfois être créée par des aimants permanents (au lieu d'un bobinage). Dans ce dernier cas, le flux dans la machine est considéré comme étant constant.

L'entrefer est l'espace entre les deux parties du circuit magnétique du moteur, le stator et le rotor.

### I.2.3 Collecteur et balais

Le collecteur est monté sur l'arbre de la machine, mais isolé de celui-ci. Les deux fils sortant de chaque bobine de l'induit sont successivement et symétriquement soudés aux lames du collecteur **[1]**.

Les balais frottent sur le collecteur en rotation assurant ainsi la connexion du rotor à un organe externe fixe.

L'ensemble collecteur/balais garantit ainsi la circulation du courant dans l'induit. Par contre, en fonctionnement générateur, l'ensemble permet de récolter une force électromotrice (f.e.m.).

## I.3 Types de fonctionnement d'un moteur à CC

Le moteur à CC est réversible (bidirectionnel). Il peut fonctionner soit en moteur soit en générateur.

### I.3.1 Fonctionnement en moteur

Si un courant est imposé par une alimentation, on aura un fonctionnement en moteur. Un couple mécanique qui fait tourner le rotor sera alors crée.

L'induit reçoit une puissance électromagnétique positive ($P_{em} > 0$).

### I.3.2 Fonctionnement en générateur

Si un courant est induit par la rotation du rotor (rotation forcée), le moteur aura un fonctionnement en générateur.

Dans ce cas, la puissance électromagnétique est négative ($P_{em} < 0$).

## I.4 Avantages et inconvénients

Les avantages des moteurs à cc sont :

- Fonctionnement simple et plus linéaire.
- Contrôle continu et presque instantané de la vitesse.
- Faible coût.

Les inconvénients des moteurs à cc sont :

- Problème d'usure (dû au frottement du collecteur).
- Incompatibilité avec le réseau de distribution. Le moteur doit être alimenté par un convertisseur alternatif continu.
- Arcs électriques.

## I.5 Caractéristiques d'un moteur à CC

Les moteurs à cc sont caractérisés par :

### I.5.1 Force électromotrice

La f.e.m est la tension (E) produite par le rotor lors de sa rotation dans le flux magnétique produit par l'inducteur.

Elle est donnée par :

$$E = \frac{PZN\emptyset}{a.60} = K\emptyset\Omega \quad [V] \qquad \textbf{(I.1)}$$

Où :

- P : nombre de paires de pôles.
- Z : nombre de conducteurs.
- a : nombre de paires de voies d'enroulement.
- N : vitesse de rotation [tr/min].
- Ø : flux magnétique par pôle en Webers [Wb].
- Ω : vitesse de rotation [rad/s].
- K : coefficient de la machine [V.s/rad.Wb].

Le terme $K = \frac{PZ}{a2\pi}$ [V.s/rad.Wb] est constant, et $\Omega = \frac{2\pi N}{60}$ .

### I.5.2 Couple électromagnétique

Le couple ($T_{em}$) est la mesure de la force servant à produire une rotation, il est crée par les champs magnétiques dans l'induit et l'inducteur.

Il est donné par :

$$T_{em} = K\emptyset I \quad [N.m] \qquad \textbf{(I.2)}$$

Où :

- K : coefficient de la machine [V.s/rad.Wb].
- Ø : Flux magnétique par pôle en Webers [Wb].
- I : Courant dans l'induit [A].

Si le flux est constant, on aura :

$$T_{em} = K'I \qquad [N.m] \tag{I.3}$$

Le terme $K' = K\emptyset$ est constant.

### I.5.3 Puissance électromagnétique

La puissance électromagnétique ($P_{em}$) est le produit de la f.e.m (E) avec le courant circulant dans l'induit (I).

Elle est donnée par :

$$P_{em} = EI \quad [W] \tag{I.4}$$

D'après le principe de conversion d'énergie, cette puissance est égale à la puissance développée par le couple électromagnétique :

$$P_{em} = T_{em}\Omega = EI \quad [W] \tag{I.5}$$

### I.5.4 Vitesse de rotation

Elle est donnée par : $$\Omega = \frac{U-RI}{K\emptyset} \tag{I.6}$$

Où :

- Ω : vitesse de rotation [rad/s].
- U et I : tension [V] et courant [A] dans l'induit respectivement.
- R : résistance de l'induit [Ω].
- K : coefficient de la machine [V.s/rad.Wb].
- Ø : flux magnétique par pôle en Webers [Wb].

D'après cette relation, la variation de vitesse d'un moteur à CC peut se faire de deux manières :

- **Par variation de flux**

La variation du courant inducteur (courant d'excitation) engendre une variation proportionnelle du flux magnétique.

Quand le flux diminue, la vitesse du moteur augmente. Lorsque le flux est proche de zéro, la vitesse augmente de manière excessive et prend des valeurs inacceptables jusqu'à l'emballement du moteur et sa destruction. Pour cette raison, on ne doit jamais annuler le courant d'excitation quand le moteur est en fonctionnement.

Pour arrêter le moteur, on doit d'abord annuler la tension d'alimentation (U) puis enlever l'excitation.

- **Par variation de la tension d'alimentation**

Si le flux magnétique est constant, ce qui signifie que l'inducteur est soit à aimant permanant ou à courant d'excitation constant, la vitesse de rotation du moteur (**Ω**) sera proportionnelle à la tension d'alimentation (U).

### I.5.5 Rendement

Il est définit comme étant le rapport de la puissance utile ($P_u$) sur la puissance absorbée ($P_a$) :

$$\eta = \frac{P_u}{P_a} \qquad \textbf{(I.7)}$$

## I.6 Démarrage d'un moteur à CC

Au démarrage, si on applique directement la tension d'alimentation au moteur, un courant de démarrage ($I_d$) de 5 à 20 fois le courant nominal ($I_n$) sera débité. Or, le courant admissible pour le démarrage est de 1.5 à 2 fois le courant nominal.

Pour remédier à ce problème, il faut limiter le courant au démarrage soit par un rhéostat de démarrage, ou en augmentant graduellement la tension d'alimentation.

## I.7 Freinage d'un moteur à CC

Le temps de freinage d'un moteur à CC dépend de l'énergie cinétique emmagasinée par celui-ci. Afin d'assurer une décélération rapide, on fait recours à deux méthodes **[1]** :

### I.7.1 Freinage dynamique

Cette méthode consiste à raccorder une résistance extérieure (R) à l'induit. Lorsqu'on coupe l'alimentation du moteur, celui-ci se comporte comme un générateur et la tension induite produit un courant induit ($I_2$) circulant dans le sens inverse du courant original ( $I_1$). On obtient alors un couple de freinage d'autant plus grand que le courant ($I_2$).

La résistance (R) est choisie de sorte à avoir un courant de freinage ($I_2$) qui est le double du courant nominal ($I_n$). Dans ce cas, le couple de freinage initial est le double du couple normal du moteur.

Le courant ($I_2$) diminue avec la diminution de la tension induite. De ce fait, le couple de freinage devient de plus en plus faible jusqu'à son annulation lorsque l'induit cesse de tourner.

### I.7.2 Freinage par inversion

Le freinage par inversion consiste à inverser brutalement le sens du courant dans l'induit en intervertissant les bornes de la source.

Cette méthode permet un freinage encore plus rapide, mais elle risque de détruire le moteur car le courant ($I_2$) peut atteindre 50 fois l'intensité nominale ($I_n$). Afin d'éviter l'endommagement du moteur, on introduit une résistance en série avec l'induit au moment de l'inversion des bornes. La résistance (R) sera ajustée tel que $I_2 = 2I_n$ (approximativement).

Contrairement au freinage dynamique, lorsque la vitesse de rotation du moteur est nulle, le courant ($I_2$) n'est qu'à la moitié de sa valeur initiale. Cela signifie qu'un couple est développé même à l'arrêt du moteur. Dans ce cas, si on veut arrêter le moteur, on doit ouvrir l'interrupteur dés que le moteur s'arrête. Sinon, le moteur repart dans le sens inverse.

## I.8 Bilan de puissance d'un moteur à courant continu

Le bilan de puissance d'un moteur à courant continu est représenté dans la **Figure I.1** :

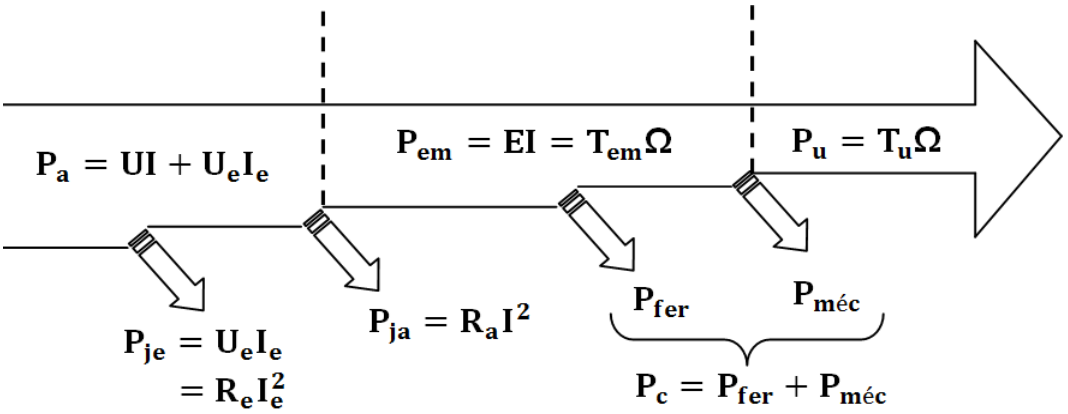



**Figure I.1 :** Bilan de puissance d'un moteur à courant continu.

Où :

- $P_a$ : puissance absorbée [W].
- $E$ : Force électromotrice [V].
- U et I : tension [V] et courant [A] dans l'induit respectivement.
- $U_e$ et $I_e$ : tension [V] et courant [A] d'excitation respectivement.
- $P_{je}$ et $P_{ja}$ : pertes joules dans l'inducteur et l'induit respectivement [W].
- $R_e$ et $R_a$ : résistance de l'inducteur et de l'induit respectivement [Ω].
- $P_{fer}$ : Pertes fer (ou ferromagnétiques) [W].
- $P_{méc}$ : Pertes mécaniques [W].
- $P_c$: Pertes collectives (constantes) [W].
- $P_u$ et $T_u$ : puissance [W] et couple utile [N.m] respectivement.
- $P_{em}$ et $T_{em}$ : puissance [W] et couple utile [N.m] respectivement.
- Ω : vitesse de rotation [rad/s].

## I.9 Modes d'excitation des moteurs à CC

Selon le raccordement de l'inducteur avec l'induit, on distingue trois principaux types d'excitation :

### I.9.1 Moteurs à excitation shunt

Appelée également excitation dérivée, le circuit d'excitation est connecté en parallèle avec l'induit (la tension est la même aux bornes de l'induit et l'inducteur). Il n'exige aucune alimentation externe.

Pour varier la tension induite, on suffit de varier rhéostat d'excitation.

### I.9.2 Moteurs à excitation série

Le circuit d'excitation est connecté en série avec l'induit. Dans ce mode, le flux magnétique dépend du courant dans l'induit.

Les moteurs séries sont caractérisés par une accélération rapide et un couple de démarrage puissant. Ils peuvent atteindre des vitesses très élevées (pour les faibles charges).

### I.9.3 Moteurs à excitation compound

L'excitation composée est une combinaison des deux modes d'excitations shunt et série (deux inducteurs). On peut réaliser un compoundage aditif si les flux magnétiques des deux circuits s'additionnent, ou bien un compoundage soustractif si les flux magnétiques des deux circuits se soustraient.

► On trouve également des moteurs à excitation séparée (indépendante). Ces types de moteurs ont besoin de deux alimentations : une pour l'inducteur et une autre pour l'induit.

## I.10 Hacheur

Un hacheur effectue une conversion continue-continue. Son entrée est une tension continue et sa sortie est une tension de valeur moyenne réglable.

### I.10.1 Principe de fonctionnement d'un hacheur série

♣ ***Remarque :*** *Les interrupteurs représentés dans les schémas ci-dessous seront remplacés en pratique par des transistors en commutations (interrupteurs électroniques).*

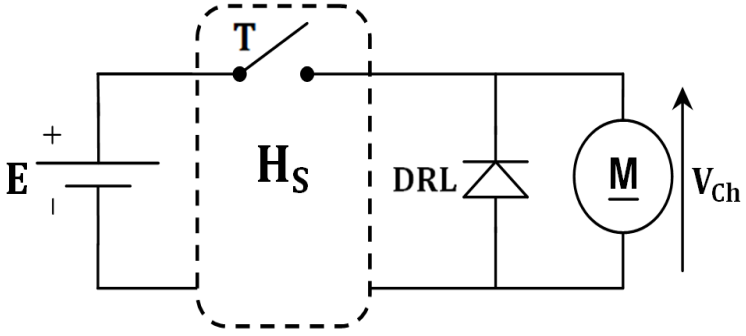



**Figure I.2 :** structure d'un hacheur série.

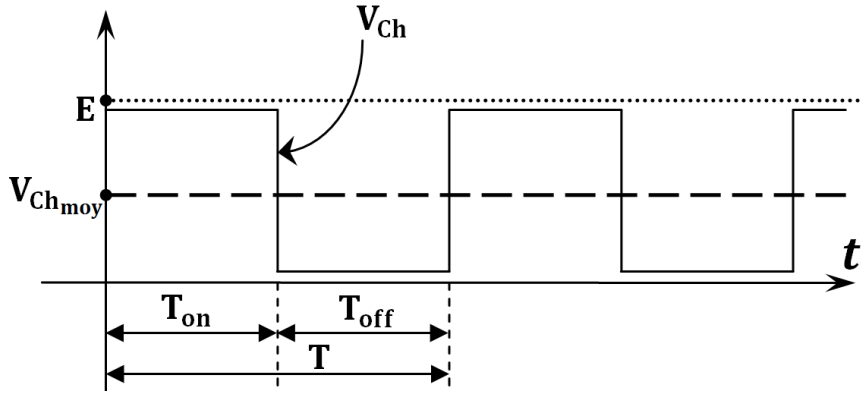



**Figure I.3 :** Courbe de tension d'un hacheur série.

Où :

- $T_{on}$ : Temps de fermeture. On a $\alpha T = T_{on}$.
- $T_{off}$ : Temps d'ouverture. On a $(1-\alpha)T = T_{off}$.
- $T$ : Période du hacheur, avec $T = T_{on} + T_{off}$.

Le hacheur dépend de la fréquence $f = \frac{1}{T}$ et du rapport cyclique $\alpha = \frac{T_{on}}{T}$.

D'après la **figure I.3**, on a :

- $0 \leq t < T_{on}$ : Interrupteur fermé (transistor saturé) $\rightarrow V_{Ch}(t) = E$.
- $T_{on} \leq t < T$ : Interrupteur ouvert (transistor bloqué) $\rightarrow V_{Ch}(t) = 0$.

La valeur moyenne de la tension de sortie ($V_{Ch_{Moy}}$) est donnée par :

$$V_{Ch_{Moy}}(t) = \frac{1}{T}\int_0^T V_{Ch}(t)dt = \frac{T_{on}}{T}E = \alpha E \qquad \textbf{(I.8)}$$

On déduit de la relation **(I.8)** qu'en variant le rapport cyclique (α) on peut agir directement sur la valeur moyenne de la tension.

Ce hacheur fonctionne dans un seul quadrant (on note 1Q) car la puissance transmise est toujours positive (courant et tension positifs) ou nulle. On dit que le hacheur est non réversible.

### I.10.2 Hacheur à 4 quadrants et principe de commande des MCC

Pour avoir un contrôle complet du moteur et gérer toutes les situations possibles, soit fonctionner à des vitesses et couples variables, et dans les deux sens de rotation, celui-ci

doit fonctionner dans 4 quadrants, (voir **figure I.4**). Ces derniers représentent le couple (T) du moteur en fonction de la vitesse de rotation (Ω). Le couple et la vitesse sont indépendants l'un de l'autre.

Pour réaliser ce fonctionnement, on utilise un hacheur à 4 quadrants (un pont H), (voir **figure I.5**).

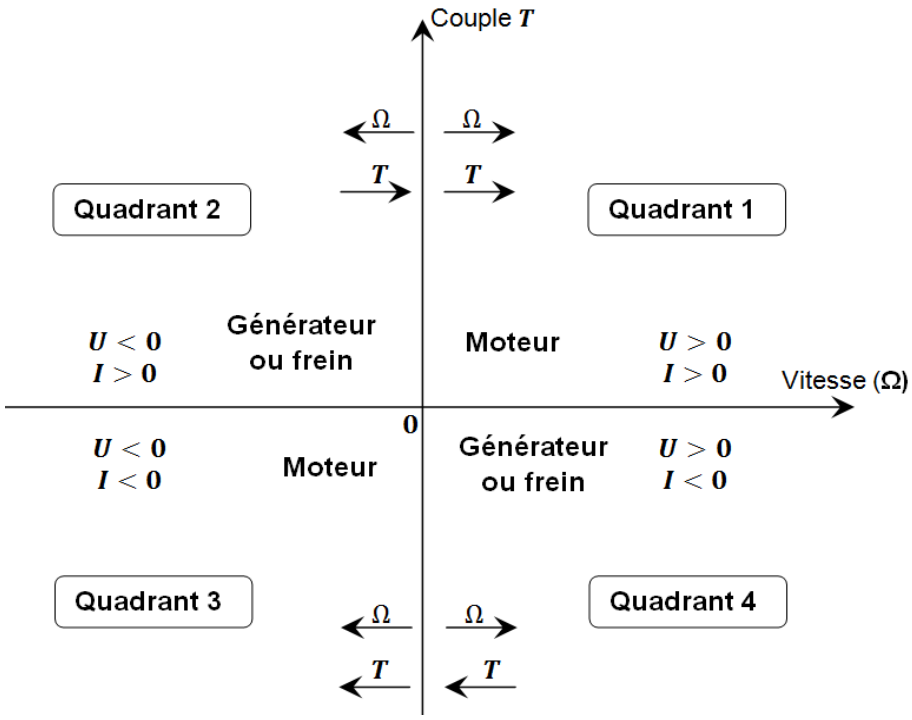



**Figure I.4 :** fonctionnement dans les 4 quadrants d'un MCC.

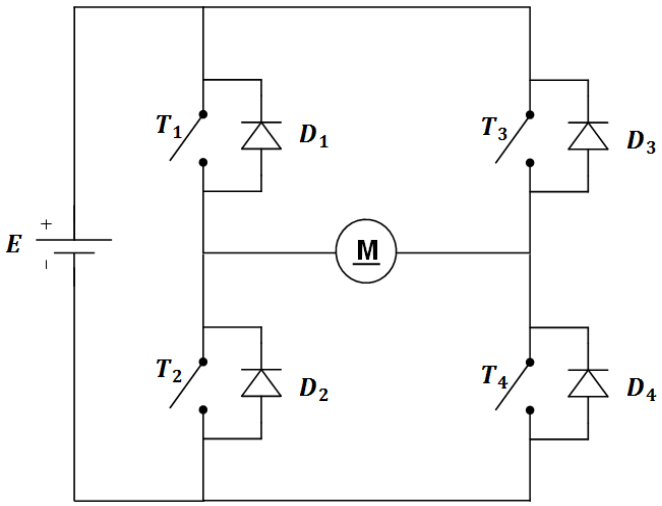



**Figure I.5 :** Montage d'un hacheur 4 quadrants.

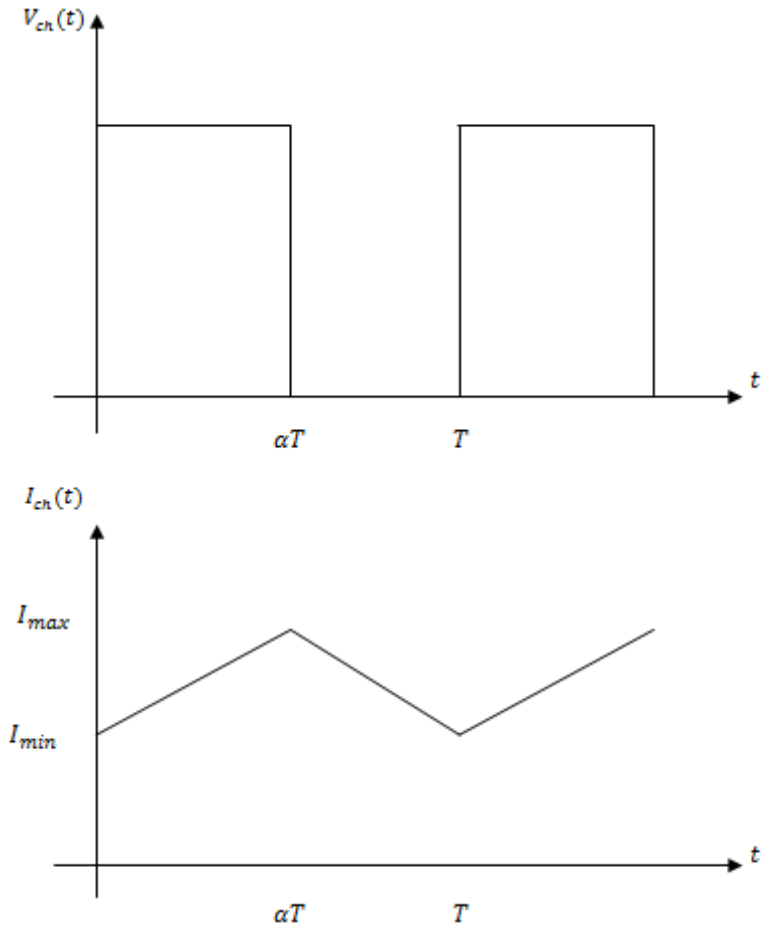



**Figure I.6 :** Diagramme tension et courant d'un hacheur.

➢ **Quadrant 1**

La puissance transmise est positive car les deux grandeurs couple et vitesse sont positives. Puisque ces deux grandeurs ont le même sens (on note sens 1), on aura donc un fonctionnement en moteur dans un sens 1.

- **Comportement du hacheur**

Les transistors $T_1$ et $T_4$ sont commandés à la fermeture (saturation) lorsque $t \in [0; \alpha T[$, soit pendant une durée $\alpha T$ , et bloqués lorsque $t \in [\alpha T; T[$, soit pendant une durée$(1 - \alpha)T$.

Dès le blocage de $T_1$ et $T_4$, le moteur passe du quadrant 1 (fonctionnement moteur sens 1) vers le quadrant 4 (fonctionnement générateur).

Les transistors $T_2$ et $T_3$ et les diodes $D_1$, $D_2$, $D_3$ et $D_4$ restent toujours bloqués.

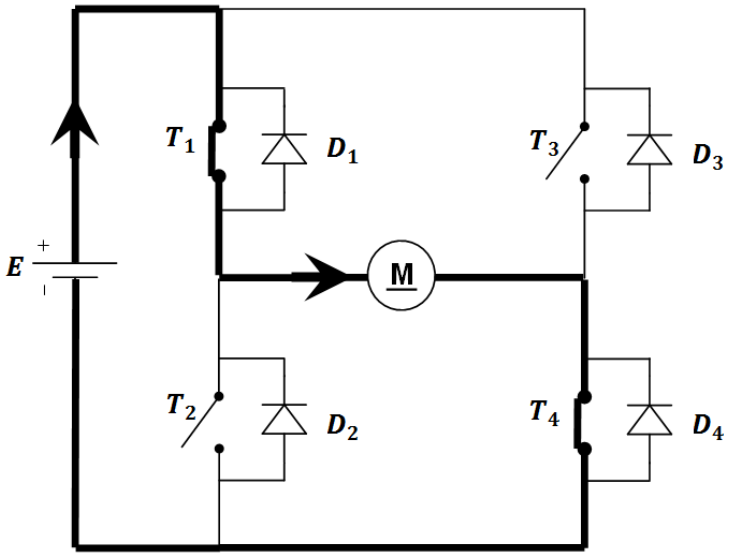



**Figure I.7 :** Comportement du hacheur 4 quadrants dans le quadrant 1.

- **Quadrant 2**

La puissance transmise est négative (on a un retour au réseau d'alimentation, soit un freinage par récupération) car la vitesse est négative et le couple est positif ($T\Omega < 0$). Ceci est dû au fait que la tension induite est supérieure à la tension d'alimentation. Le moteur aura alors un fonctionnement en générateur dans un sens 2.

Dans les petits moteurs, toute l'énergie est dissipée sous forme de chaleur à travers une résistance, donc le moteur fonctionne comme un frein.

- **Comportement du hacheur**

Les diodes $D_1$ et $D_4$ conduisent pendant une durée $(1-\alpha)T$ car le courant est négatif (sens 2), Elles rentrent en conduction dès le blocage (ouverture) des transistors $T_2$ et $T_3$.

Les diodes $D_2$ et $D_3$ et les transistors $T_1$, $T_2$, $T_3$ et $T_4$ restent toujours bloqués (ouverts).

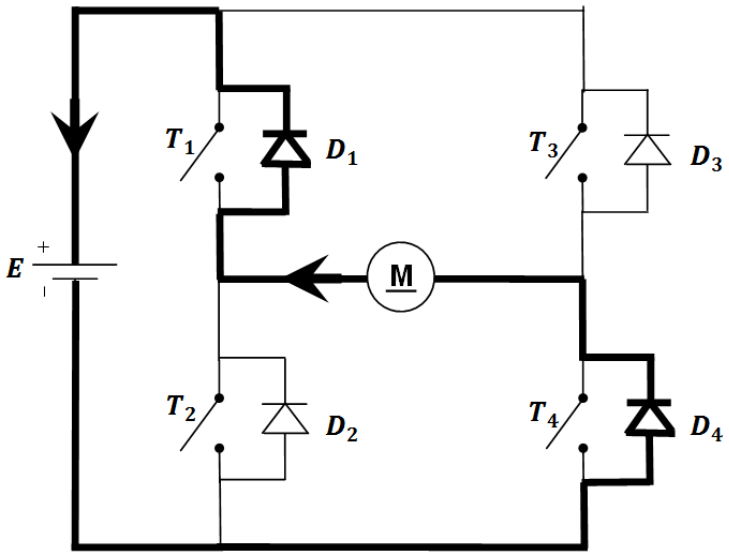



**Figure I.8 :** Comportement du hacheur 4 quadrants dans le quadrant 2.

➢ **Quadrant 3**

Le quadrant 3 a le même fonctionnement que le quadrant 1 sauf que dans le sens inverse. Il suffit simplement de changer la polarité de l'alimentation. La puissance est toujours positive car les deux grandeurs sont négatives ($T\Omega > 0$). On aura donc un fonctionnement en moteur dans un sens 2.

- **Comportement du hacheur**

Les transistors $T_2$ et $T_3$ sont commandés à la fermeture (saturation) lorsque $t \in [0;\ \alpha T[$, soit pendant une durée $\alpha T$ ; et bloqués (ouverts) lorsque $t \in [\alpha T; T[$, soit pendant une durée $(1-\alpha)T$.

Dès le blocage de $T_2$ et $T_3$, le moteur passe du quadrant 3 (fonctionnement moteur sens 2) vers le quadrant 2 (fonctionnement générateur ou frein sens 2).

Les transistors $T_1$ et $T_4$ et les diodes $D_1$, $D_2$, $D_3$ et $D_4$ restent toujours bloqués (ouverts).

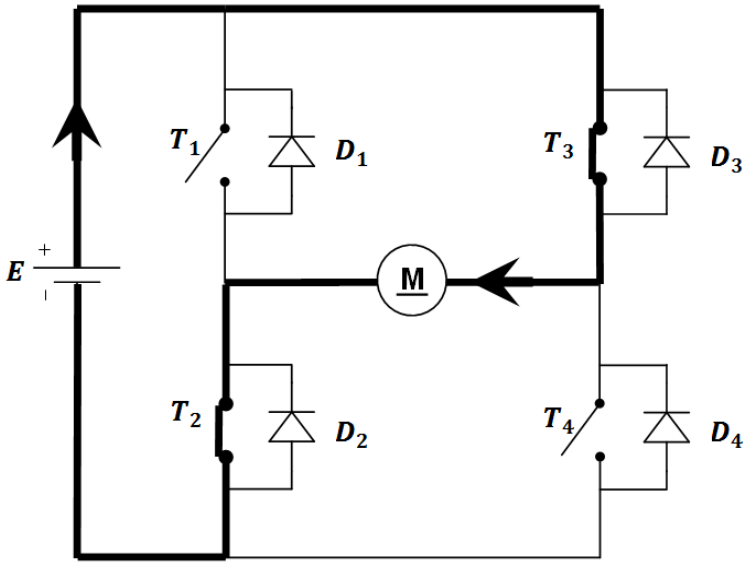



**Figure I.9 :** Comportement du hacheur 4 quadrants dans le quadrant 3.

➢ **Quadrant 4**

Le quadrant 4 a le même fonctionnement que le quadrant 2 sauf que dans le sens inverse (le couple, la vitesse et la polarité sont inversés). La puissance reste négative ($T\Omega < 0$). Le moteur aura un fonctionnement en générateur dans un sens 1.

- **Comportement du hacheur**

Les diodes $D_2$ et $D_3$ conduisent pendant une durée $(1-\alpha)T$ car le courant est positif (sens 1). Elles rentrent en conduction dès le blocage (ouverture) des transistors $T_1$ et $T_4$.

Les diodes $D_1$ et $D_4$ et les transistors $T_1$, $T_2$, $T_3$ et $T_4$ restent toujours bloqués (ouverts).

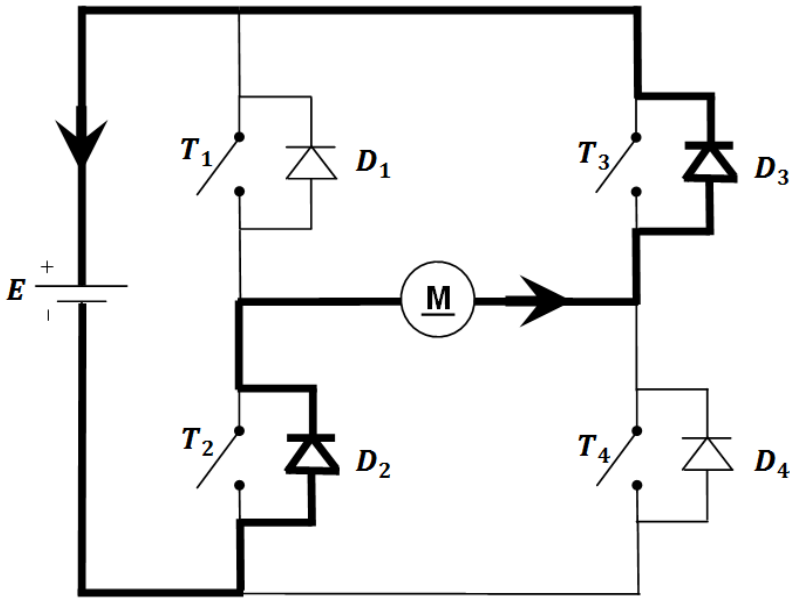



**Figure I.10 :** Comportement du hacheur 4 quadrants dans le quadrant 4.

### I.10.2.1 Relations de valeur moyenne

- **Valeur moyenne de la tension $V_{Ch_{Moy}}$ :**

$$V_{Ch_{Moy}} = \frac{1}{T}\int_0^T V_{Ch}(t)dt = \alpha E \qquad \textbf{(I.9)}$$

- **Valeur moyenne du courant $I_{Ch_{Moy}}$:**

$$I_{Ch_{Moy}} = \frac{I_{max}+I_{min}}{2} \qquad \textbf{(I.10)}$$

- **Valeur moyenne de la puissance $P_{Moy}$:**

$$P_{Moy} = V_{Ch_{Moy}} I_{Ch_{Moy}} \qquad \textbf{(I.11)}$$

Le signe de la puissance moyenne dépend du signe de la valeur moyenne du courant $I_{Ch_{Moy}}$.

## I.11 Conclusion

Dans ce chapitre, nous avons décrit brièvement la structure et le fonctionnement des moteurs à courant continu et présenté leurs caractéristiques. Ensuite, nous avons mentionné certains modes d'excitation des moteurs à CC. Puis, on à vu le principe de fonctionnement des hacheurs séries. Enfin, nous avons présenté le principe de commande d'un moteur à CC en utilisant un hacheur à quatre quadrants.

Dans le chapitre suivant, on présentera la structure et les fonctionnalités du PIC 16F877.

# *Chapitre II*

# *Présentation du PIC 16F877*

## II.1 Introduction

Les microcontrôleurs PIC sont des circuits intégrés programmables qui effectuent diverses taches plus au moins complexes. Ils sont essentiels pour le fonctionnement de nombreux appareils tels que les systèmes de contrôle et de surveillance, les moteurs, les systèmes embarqués ou les dispositifs médicaux avancés.

La société Microchip Technology propose trois grandes familles de microcontrôleurs PIC :

- Famille Base-Line, instructions sur 12 bits, comme PIC12F510.
- Famille Mid-Range, instructions sur 14 bits, comme PIC16F877.
- Famille High-End, instructions sur 16 bits, comme PIC18F6680.

Dans ce chapitre, nous allons décrire l'architecture interne et les fonctionnalités du microcontrôleur PIC 16F877.

## II.2 Choix du microcontrôleur

Le choix du PIC16F877 à été fait en considérant les caractéristiques matérielles suivantes :

- Nombre de broches d'E/S.
- Types d'interface.
- Taille de la mémoire programme (ROM).
- Taille de la mémoire vive (RAM).
- Vitesse de fonctionnement.
- Consommation électrique.
- Environement de développement (matériel et logiciel).

On doit également tenir compte de certains facteurs importants tels que :

- Faible coût.
- Documentation suffisante.
- Disponibilité du composant sur le marché.

Le choix a été fait de façon à répondre aux exigences materielles et logicièlles du projet de manière optimale et à coût minimal.

## II.3 Unité centrale de traitement

L'Unité centrale de traitement (CPU) est la partie qui récupère et exécute les instructions contenues dans un programme conservées en mémoire. Elle a un registre de travail principal (W), à travers lequel toutes les données doivent passer et une unité arithmétique et logique (ALU).

L'ALU effectue des opérations arithmétiques (au niveau du bit) et logiques. Elle a aussi comme rôle le contrôle des bits du registre STATUS au fur et à mesure qu'ils soient modifiés par l'exécution des différentes instructions du programme.

Le microcontrôleur PIC16F877 possède un processeur de conception RISC (Reduced Instruction Set Computer).

### II.3.1 Architecture RISC

La conception RISC procure un nombre d'instructions limité. Chaque instruction effectue plus d'opérations élémentaires, ce qui rend l'exécution plus rapide et la taille du programme réduite avec un accès optimisé à la mémoire principale.

Cette architecture est plus performante comparée à l'architecture CISC qui a un jeu d'instructions complexe (nombre d'instructions élevé), et par conséquent, un temps de décodage plus long.

## II.4 Oscillateur

Un périphérique externe est requis pour produire les cycles d'horloge nécessaires au fonctionnement du PIC.

Le système d'horloge est conçu pour qu'une instruction soit extraite, décodée et exécutée tous les quatre cycles d'horloge (soit un oscillateur de fréquence $F_{OSC}$ aura une fréquence d'horloge de $F_{OSC}/4$ cycles par seconde) **[2]**.

Le PIC 16F877 prend en charge huit modes d'oscillateurs différents. Le mode d'oscillateur est définit lors de la programmation de l'appareil à travers des bits de configuration, qui sont des indicateurs non volatils. Le mode d'oscillateur ne peut pas être modifié lors de l'exécution.

## II.5 Réinitialisation du système

Le mécanisme de réinitialisation place le PIC dans un état connu. Lors de la mise sous tension, le processeur déclenche lui-même une réinitialisation. Cette dernière est aussi utilisée pour maîtriser un programme qui a cessé de fonctionner, ou pour préparer le chargement d'un programme vers le dispositif.

La RAM de l'utilisateur n'est pas affectée par une réinitialisation car les registres GPR sont dans un état inconnu lors de la mise sous tension, donc ils ne seront pas modifiés par réinitialisation. Les registres SFR, par contre, sont réinitialisés à un état initial. Le plus important d'entre eux est le compteur de programmes (PC) qui est remis à zéro. Cette action dirige l'exécution vers la première instruction et redémarre efficacement le programme **[2]**.

## II.6 Interruptions

Le mécanisme d'interruption est essentiel pour répondre rapidement aux événements internes ou externes au fur et à mesure qu'ils surviennent.

Lorsqu'une interruption se produit, le microcontrôleur quitte son flux normal d'exécution de programme et passe à une partie spéciale du programme connue sous le nom de routine de service d'interruption (ISR). Le code du programme à l'intérieur de l'ISR est exécuté, et au retour de l'ISR, le programme reprend son flux normal d'exécution **[3]**.

L'activation et le contrôle des interruptions se font à travers le registre INTCON. Ces interruptions peuvent être ignorées par le processeur.

On peut avoir plusieurs sources d'interruption. Par exemple, une Interruption CCP, une Interruption externe (broche INT), Interruption de réception et de transmission.

## II.7 Registres OPTION_REG et STATUS

Le registre OPTION_REG contient des bits liés aux timers internes, au timer chien de garde (WDT) et aux interruptions.

Le registre STATUS fait partie des registres SFR et peut être la destination de toute instruction. Il est accessible depuis n'importe quelle banque. Il contient les bits de sélection de banques, l'état du reset et l'état arithmétique de l'ALU.

## II.8 Organisation des mémoires

- **Architecture Harvard**

Dans l'architecture Harvard, les données et les instructions se trouvent dans des zones mémoire distinctes. Elles utilisent des zones de stockage et chemins de signaux différents.

L'architecture Harvard permet au processeur de lire et écrire des instructions et des données vers et depuis la mémoire en même temps (donc un accès plus rapide à la mémoire), contrairement à l'architecture traditionnelle Von Neumann dont la lecture et l'écriture des instructions ou des données par le processeur ne se font pas simultanément, car elles utilisent les mêmes lignes de signal.

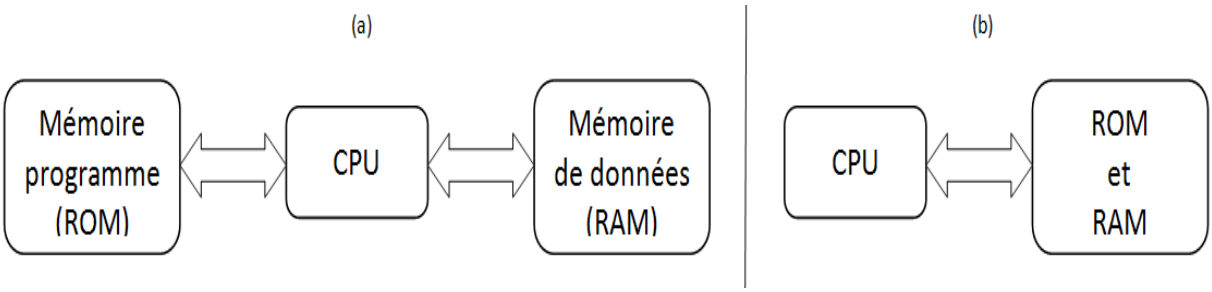



**Figure II.1: (a)** Architecture Harvard, **(b)** Architecture Von Neumann.

### II.8.1 Mémoire de données (RAM)

La mémoire de données est organisée en 4 banques (0 à 3) de 128 adresses chacune (soit une taille totale de 512 adresses). Elle contient à la fois des registres de fonctions spéciales (SFR), qui ont un objectif spécifique, et des registres à usage général (GPR).

### II.8.1.1 Registres de données

- **Banques mémoire**

Le nombre de banques varie en fonction de la quantité de RAM disponible, toujours par multiples de 128 octets. Dans le PIC16F877, on trouve quatre banques dont l'accès s'effectue via les bits de sélection de banque spéciaux (RP0 et RP1) dans le registre STATUS.

Tous les registres de données sont sur 8 bits, et on distingue deux types :

- **Registres à fonctions spéciales (SFR)**

Les SFR ont des noms réservés tels que STATUS et Timer0. Ils sont situés aux adresses basses de chaque banque RAM. On peut trouver des SFR en mode lecture et écriture ou bien en lecture seule.

Étant donné que certains registres sont accessibles dans plusieurs banques (registres mis en miroir), comme le registre PORTB de données qui est accessible dans la banque 0 et dans la banque 2, le changement de banque peut être minimisé par le compilateur lors de l'assemblage du code machine, économisant ainsi de l'espace et du temps d'exécution du programme **[4]**.

- **Registres à usage général (GPR)**

Les registres à usage général occupent une zone mémoire de 368 octets. Ces registres ont une zone commune de 16 octets dans la banque 0 (de 070h à 07Fh) qui est mise en miroir dans les trois banques 1, 2 et 3, voir **Figure II.2**.

L'intérêt de la zone commune est qu'elle permet l'accès aux variables utilisateur depuis n'importe quelle banque.

Lorsque des variables sont créées en langage C, elles sont stockées dans les GPR, à partir de l'adresse 0020h **[4]**.

| | Banques (**512 octets**) | | | |
|---|---|---|---|---|
| | Banque 0 (**128 octets**) | Banque 1 (**128 octets**) | Banque 2 (**128 octets**) | Banque 3 (**128 octets**) |
| SFR | Registre STATUS | Registre STATUS | Registre STATUS | Registre STATUS |
| | Port B de données | | Port B de données | |
| GPR | **96 octets** (020h à 07F) | **80 octets** | **96 octets** | **96 octets** |
| | Accès commun GPR **16 octets** (070h à 07Fh) | Accès (070h à 07Fh) | Accès (070h à 07Fh) | Accès (070h à 07Fh) |

**Figure II.2 :** Représentation simplifiée des registres de données SFR et GPR.

### II.8.2 Mémoire programme

Le PIC16F877 dispose de 8Kx14 mots de mémoire programme flash avec un compteur de programme sur 13 bits. Cet espace mémoire est divisé en quatre pages de 2K mots chacune. Seule une partie de cette mémoire peut être implémentée **[5]**.

Pour basculer entre les pages de la mémoire programme, les bits hauts du compteur de programme (PC) doivent être modifiés. Cela se fait en écrivant la valeur souhaitée dans un SFR appelé PCLATH (Program Counter Latch High) **[5]**.

### II.8.3 Mémoire EEPROM

La mémoire EEPROM est adressée via les registres SFR et permet des opérations de lecture et d'écriture. Sa taille est de 256 octets.

L'EEPROM est séparée de la mémoire de données et la mémoire programme. Elle est utilisée comme stockage non volatile, c'est-à-dire qu'elle conserve ses données une fois l'alimentation coupée.

## II.9 Interfaces série

On distingue principalement deux types d'interfaces série :

### II.9.1 Module USART

Le module émetteur-récepteur synchrone asynchrone universel (USART) est également connu sous le nom d'interface de communication série (SCI). Le module USART est utilisé pour communiquer avec les appareils et les systèmes qui prennent en charge les communications RS-232 **[2]**.

#### II.9.1.1 Modes de configuration de l'USART

On trouve trois modes de configuration de l'USART **[7]** :

- **Mode asynchrone USART**

Données transmises en full-duplex. L'émetteur et le récepteur sont fonctionnellement indépendants, mais utilisent le même format de données et la même vitesse de transmission. Il peut communiquer avec des périphériques tels que les terminaux CRT et les ordinateurs personnels. Ce mode n'utilise pas de connexion d'horloge.

- **Mode maître synchrone USART**

Données transmises de manière half-duplex. Lors de la transmission de données, la réception est inhibée et vice versa. Il est utilisé pour une communication avec un circuit intégré A/N et N/A ou bien un accès à la mémoire EEPROM série.

Il utilise un signal d'horloge séparé. Le mode maître indique que le processeur transmet l'horloge maître sur la ligne CK.

- **Mode esclave synchrone USART**

Diffère du mode maître par le fait que l'horloge de décalage est fournie en externe au niveau de la broche RC6/TX/CK (au lieu d'être fournie en interne en mode maître).

### II.9.2 Module de port série synchrone maître (MSSP)

Le module MSSP fournit deux principaux types de communication, SPI et I2C **[4]**.

#### II.9.2.1 Bus SPI (Serial Peripheral Interface)

Le bus d'interface périphérique série (SPI) permet un échange de données synchrone à grande vitesse sur des distances relativement courtes (généralement au sein d'un ensemble de cartes connectées), en utilisant un système maître/esclave avec sélection matérielle des esclaves.

Un processeur doit agir en tant que maître, générant l'horloge. Les esclaves peuvent être d'autres microcontrôleurs ou périphériques avec une interface SPI.

#### II.9.2.2 Bus I2C (Inter-Integrated circuit)

Le circuit inter-intégré (I2C) est conçu pour la communication à courte portée entre les puces d'un même système à l'aide d'un système d'adressage logiciel. Il ne nécessite que deux fils de signal et fonctionne comme un réseau local simplifié.

Il permet à un contrôleur maître d'être connecté à jusqu'à 1023 appareils, comme d'autres microcontrôleurs, des convertisseurs analogiques ou bien une mémoire EEPROM externe, qui serait utilisée pour étendre le stockage de données non volatile **[6]**.

## II.10 Port esclave parallèle (PSP)

Le port esclave parallèle est sur 8 bits. Il permet de communiquer (en parallèle) avec un périphérique ou un bus de données externe.

Les huit PSP du PIC16F877 sont fournis par le port D. Le port E dispose de trois lignes de contrôle qui sont $\overline{RD}$ pour la lecture, $\overline{WR}$ pour l'écriture et $\overline{CS}$ pour le contrôle de sélection.

## II.11 Convertisseur Analogique-Numérique (module A/N)

Le convertisseur A/N est utilisé pour convertir un signal analogique, comme la tension issue d'un capteur, sous forme numérique afin qu'il soit lu et traité par le microcontrôleur. Une interruption est générée à la fin de la conversion pour lire les données converties.

Le PIC16F877 contient huit canaux d'entrées analogiques avec une résolution de 10 bits. D'autres convertisseurs A/N externes peuvent également être connectés au microcontrôleur.

## II.12 Ports d'entrées/sorties (E/S)

Les ports permettent aux PIC d'accéder au monde extérieur et sont mappés aux broches physiques de l'appareil. Certaines broches sont multiplexées avec des fonctions alternatives des modules périphériques. Lorsqu'un module périphérique est activé, cette broche cesse d'être une E/S à usage général **[2]**.

- **Registre TRIS**

Les ports généraux étant bidirectionnels, le registre TRIS permet de déterminer la configuration de chaque port comme entrée ou sortie. Chaque port à son propre registre TRIS.

A la mise sous tension ou à la réinitialisation, les ports B, C et D sont définis par défaut comme entrées numériques et les ports A et E sont définis par défaut comme entrées analogiques. On peut cependant configurer les ports A et E de tel sorte à avoir un mélange d'E/S analogiques et numériques selon nos besoins.

Dans le PIC 16F877, on a cinq ports parallèles (A, B, C, D et E) avec un total de 33 broches d'E/S numériques :

- Port A sur 6 bits (RA0 – RA5).
- Port B sur 8 bits (RB0 – RB7).
- Port C sur 8 bits (RC0 – RC7).
- Port D sur 8 bits (RD0 – RD7).
- Port E sur 3 bits (RE0 – RE2).

## II.13 Modules Timer

Le PIC16F877 possède trois timers :

### II.13.1 Module Timer0

Le timer/compteur du module Timer0 présente les caractéristiques suivantes **[7]**:

- Timer/compteur 8 bits.
- Lecture/écriture.
- pré-diviseur 8 bits programmable par logiciel.
- Sélection d'horloge interne *(Timer)* ou externe *(compteur)*.
- Interruption sur débordement de FFh à 00h.
- Sélection du front de l'horloge en mode horloge externe.

Les bits de configuration de TMR0 sont dans le registre OPTION. En Mode Timer, le registre TMR0 est incrémenté à chaque cycle d'instructions, en considérant le pré-diviseur avec un rapport de 1 (donc sans pré-diviseur), ce qui signifie que l'entrée d'horloge externe est la même que la sortie du pré-diviseur.

En Mode Compteur, le registre TMR0 est incrémenté à chaque front montant ou descendant de la broche RA4/T0CKI (Pin 6).

Le pré-diviseur est mutuellement et exclusivement partagé entre le module Timer0 et le Watchdog Timer. Lorsque le pré-diviseur est utilisé pour le module Timer0, cela signifie qu'il n'y a pas de pré-diviseur pour le Watchdog Timer, et vice-versa **[7]**.

Une interruption peut être générée pour effectuer des opérations de synchronisation précises à l'intérieur du microcontrôleur.

- **Chien de garde (WDT)**

Le timer chien de garde est utilisé pour détecter les disfonctionnements du système tel que les boucles infinies. On le trouve généralement dans les systèmes en temps réel où la fin réussie d'une ou plusieurs activités doit être vérifiée régulièrement.

Le WDT réinitialise automatiquement le processeur après une période donnée (18ms par défaut). Pour maintenir un fonctionnement normal, le WDT doit être désactivé ou réinitialisé dans la boucle du programme avant la fin du délai d'expiration défini. Sinon, le programme est susceptible de mal se comporter, en raison de la réinitialisation aléatoire du microcontrôleur **[4]**.

### II.13.2 Module Timer1

Le module Timer1 est un Timer/compteur 16 bits composé de deux registres TMR1H et TMR1L de 8 bits chacun. Ces derniers permettent la lecture et l'écriture. En cas de débordement, une interruption Timer1 est générée. Celle-ci peut être activée/désactivée à travers des bits d'activation/désactivation d'interruption. Timer1 peut également être activé/désactivé.

Le module Timer1 possède une entrée reset interne. Ce reset peut être généré par l'un des deux modules CCP **[7]**.

#### II.13.2.1 Fonctionnement de Timer1 en mode Timer

En mode Timer, Timer1 incrémente chaque instruction de cycle. Dans ce mode, l'horloge d'entrée du timer est $F_{OSC}/4$ .

#### II.13.2.2 Fonctionnement du Timer1 en mode compteur

En mode compteur, Timer1 est incrémenté à chaque front montant de l'entrée d'horloge externe. Timer1 peut fonctionner en mode synchrone ou asynchrone.

##### II.13.2.2.1 Fonctionnement de Timer1 en mode compteur synchrone

Ce mode permet au timer de s'incrémenter sur chaque front montant de l'entrée d'horloge sur la broche 16, ou sur la broche 15 (selon le bit activé).

L'entrée d'horloge externe est synchronisée avec l'horloge interne.

##### II.13.2.2.2 Fonctionnement de Timer1 en mode compteur asynchrone

En mode compteur asynchrone, Timer1 ne peut pas être utilisé comme base de temps pour les opérations de capture ou de comparaison. L'entrée d'horloge externe n'est pas synchronisée avec l'horloge interne **[7]**.

#### II.13.2.3 Oscillateur du Timer1

Le PIC16F877 intègre un oscillateur à cristal de faible puissance (jusqu'à 200 kHz) entre les broches 16 (entrée) et 15 (sortie). Lorsque l'oscillateur du Timer1 est activé, les broches 16 et 15 sont définies comme étant des entrées.

### II.13.3 Module Timer2

Le module Timer2 est un timer 8 bits avec un pré-diviseur et un post-diviseur. Le registre TMR2 permet la lecture et l'écriture.

Le module peut être utilisé pour générer une sortie PWM qui est utile pour piloter des moteurs à courant continu. Il est également utilisé dans les modes de capture et de comparaison, pour mesurer plus facilement les signaux externes **[6]**.

Le Timer2 peut être désactivé (registre T2CON) pour réduire la consommation d'énergie.

## II.14 Module Capture/Comparaison/PWM

Les modules de capture et de comparaison CCP1 et CCP2 sont multiplexés sur les broches 17 et 16 respectivement. Parmi leurs fonctions, on peut citer : le comptage, la génération d'impulsions et la moyenne de tension.

Chaque module contient un registre 16 bits pouvant fonctionner comme registre de capture ou registre de comparaison et un autre registre de cycle de service maître/esclave PWM **[2]**.

En mode Comparaison ou Capture, les modules utilisent le Timer1. En mode PWM, ils utilisent le Timer2.

Les modules CCP1 et CCP2 sont identiques en fonctionnement, à l'exception du fonctionnement du déclencheur d'événement spécial **[7]**:

- **Module CCP1**

Le registre CCPR1 est composé de deux registres 8 bits : CCPR1L et CCPR1H. Le registre CCP1CON contrôle le fonctionnement de CCP1.

Le déclencheur d'événement spécial est généré par une correspondance de comparaison et réinitialise Timer1

- **Module CCP2**

Le registre CCPR2 est composé de deux registres 8 bits : CCPR2L et CCPR2H. Le registre CCP2CON contrôle le fonctionnement de CCP2.

Le déclencheur d'événement spécial est généré par une correspondance de comparaison et réinitialise Timer1 et démarre une conversion A/N (si le module A/N est activé).

### II.14.1 Mode de capture

En mode capture, CCPR1 capture la valeur 16 bits du registre TMR1 lorsqu'un événement se produit sur la broche RC2/CCP1. Un événement est défini comme l'un des suivants **[5]** :

- Chaque front descendant.
- Chaque front montant.
- Chaque 4ème front montant.
- Chaque 16ème front montant.

Lorsqu'un front montant ou descendant (sélectionnable) est détecté, le registre TMR1 est effacé et commence à compter à la fréquence d'horloge interne. Lorsque le prochain front actif est détecté à l'entrée, la valeur du registre de TMR1 est copiée dans le registre CCP1 **[4]**.

Le Timer1 doit être exécuté en mode Timer, ou en mode Compteur synchronisé et la broche 17 (RC2/CCP1) doit être configurée comme entrée. L'opération de capture peut ne pas fonctionner en mode Compteur asynchrone **[7]**.

Lors d'un reset, ou lorsque le module CCP n'est pas en mode capture ou éteint le pré-diviseur est effacé **[5]**.

### II.14.2 Mode de comparaison

En mode comparaison, le registre CCPR1 de 16 bits est pré-chargé avec une valeur définie. Cette valeur est continuellement comparée à la valeur des registres Timer1 (TMR1H et TMR1L). Lorsqu'il y a correspondance, la broche de sortie bascule et une interruption CCP1 est générée **[4]**.

La broche 17 (RC2/CCP1) doit être configurée comme sortie et le Timer1 doit être exécuté en mode timer ou en mode Compteur synchronisé. L'opération de comparaison risque de ne pas fonctionner en mode compteur asynchrone. **[7]**.

### II.14.3 Mode PWM (Pulse Width Modulation)

La PWM ou MLI (Modulation de Largeur d'impulsion) est une technique simple et efficace utilisée pour générer une tension analogique variable à l'aide de signaux numériques. Elle permet de varier la durée de l'impulsion tout en gardant le nombre d'impulsions constant.

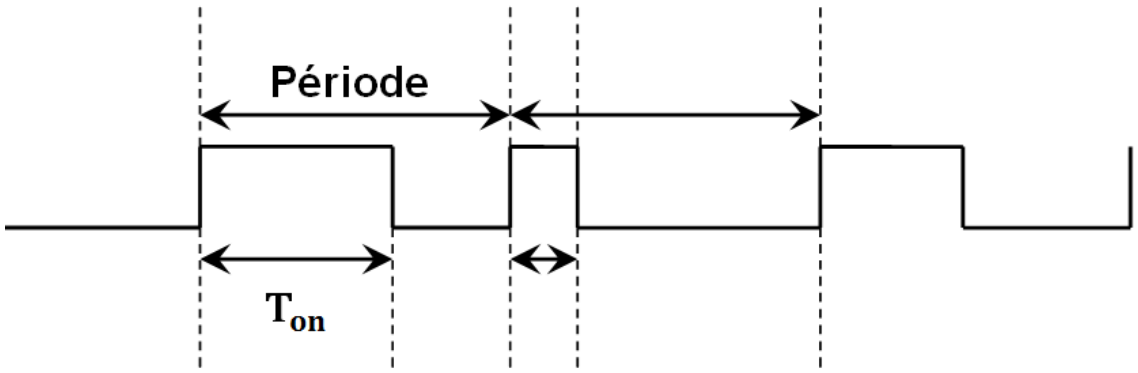



**Figure II.3 :** Sortie du module PWM.

Cette méthode a diverses applications, comme le contrôle de la puissance moyenne délivrée à une charge ou le contrôle de la vitesse d'un moteur à courant continu.

Le PIC 16F877 a deux modules PWM connectés aux broches 17 et 16. Le module produit une sortie PWM à une résolution de 10 bits. Cette sortie est essentiellement une forme d'onde carrée avec une période et un rapport cyclique spécifiés.

#### II.14.3.1 Période PWM

La période PWM est donnée par **[3]** :

$$\mathbf{Période\ PWM} = (\mathbf{PR2} + \mathbf{1}) * (\mathbf{TMR2PS}) * \mathbf{4} * \mathbf{T_{OSC}} \qquad \mathbf{(II.1)}$$

Où :

- PR2 est la valeur chargée dans le registre Timer 2.
- TMR2PS est la valeur du pré-diviseur du Timer 2.
- $T_{OSC}$ est la période de l'oscillateur d'horloge (en secondes).

La fréquence PWM est définie comme :

$$\mathbf{F_{PWM}} = \frac{\mathbf{1}}{\mathbf{Période\ PWM}} \qquad \mathbf{(II.2)}$$

La période PWM est spécifiée en écrivant dans le registre PR2. Lorsque TMR2 est égal à PR2, les trois événements suivants se produisent lors du cycle d'incrémentation suivant **[5]**:

- Le TMR2 est effacé.
- La broche CCP1 est définie (exception, si le rapport cyclique PWM = 0%).
- Le cycle de service PWM est verrouillé de CCPR1L à CCPR1H.

Le post-diviseur Timer2 n'est pas utilisé dans la détermination de la fréquence PWM **[5]**.

### II.14.3.2 Rapport cyclique de la PWM

La résolution du rapport cyclique PWM est de 10 bits. Le rapport cyclique PWM est sélectionné en écrivant les huit bits les plus significatifs (MSb) dans le registre CCPR1L et les deux bits les moins significatifs (LSb) dans les bits 4 et 5 du registre CCP1CON.

Le rapport cyclique est donné par **[3]** :

$$\textbf{Rapport cyclique PWM} = (\textbf{CCPR1L} : \textbf{CCP1CON} < 5:4 >) * \textbf{TMR2PS} * \mathbf{T_{OSC}} \quad \textbf{(II.3)}$$

CCPR1L et CCP1CON <5:4> peuvent être écrits à tout moment, mais la valeur du rapport cyclique n'est pas verrouillée dans CCPR1H jusqu'à ce qu'une correspondance entre PR2 et TMR2 se produise (période terminée). En mode PWM, CCPR1H est un registre en lecture seule **[7]**.

La résolution PWM maximale (en bits) pour une fréquence PWM donnée est **[5]**:

$$\textbf{Résolution} = \frac{\textbf{Log}\left(\frac{\mathbf{F_{OSC}}}{\mathbf{F_{PWM}}}\right)}{\textbf{Log}(\mathbf{2})} \quad \textbf{(II.4)}$$

### II.14.3.3 Configuration de la PWM

Les étapes suivantes doivent être suivies lors de la configuration du module CCP pour un fonctionnement PWM **[7]**:

- Régler la période PWM en écrivant dans le registre PR2.
- Définir le rapport cyclique PWM en écrivant dans le registre CCPR1L et les bits CCP1CON <5:4>.
- Faire de la broche CCP1 une sortie en effaçant le bit TRISC <2>.

- Définir la valeur du pré-diviseur TMR2 et activer Timer2 en écrivant dans T2CON.
- Configurer le module CCP1 pour le fonctionnement PWM.

## II.15 Conclusion

Dans ce chapitre, nous avons présenté brièvement la structure du PIC16F877 en décrivant le fonctionnement de ses différents modules, ses modes de communication ainsi que ses interfaces.

Dans le prochain chapitre, nous aborderons l'étude et la réalisation de la carte du variateur de vitesse.

# *Chapitre III*

# *Etude et Réalisation du circuit du variateur de vitesse*

## III.1 Introduction

Il existe différents types de variateurs de vitesse. Les différences se situent au niveau de la méthode de commande, la topologie et le type de charge.

Afin de réaliser la simulation (interactive) de la carte, nous utiliserons le logiciel *Proteus ISIS* pour créer le schéma du circuit et le tester avant la fabrication de la carte.

Pour la partie programmation, nous utiliserons le compilateur *mikroC PRO for PIC* qui est un compilateur de langage C.

La réalisation sera faite à l'aide du logiciel ARES à travers lequel on effectuera le routage des pistes du circuit du variateur de vitesse.

## III.2 Schéma synoptique

La figure ci-dessous représente le schéma synoptique du variateur de vitesse :

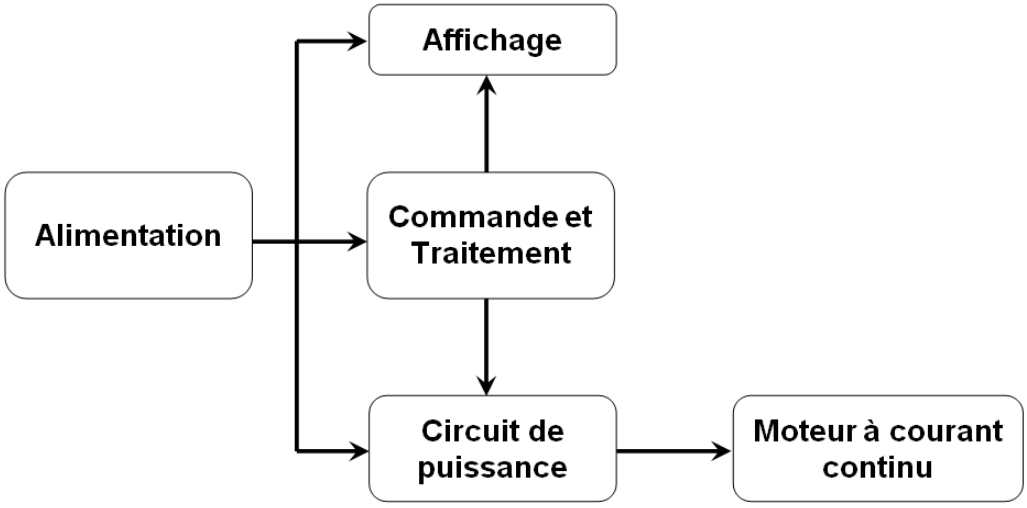



**Figure III.1 :** Schéma synoptique du variateur de vitesse.

## III.3 Description des différentes parties du circuit

### III.3.1 Alimentation

L'alimentation permet comme son nom l'indique d'alimenter les différentes parties du circuit du variateur de vitesse. Dans cette réalisation, nous avons besoin de deux tensions :

une tension de 12V notée (Vs2) pour alimenter le hacheur quatre quadrants, et une tension 5V notée (Vs1) pour alimenter les autres parties du circuit comme l'afficheur LCD et les différentes parties du PIC 16F877 ($V_{dd}$, $\overline{MCLR}$, ...).

Le circuit d'alimentation est représenté dans la figure suivante :

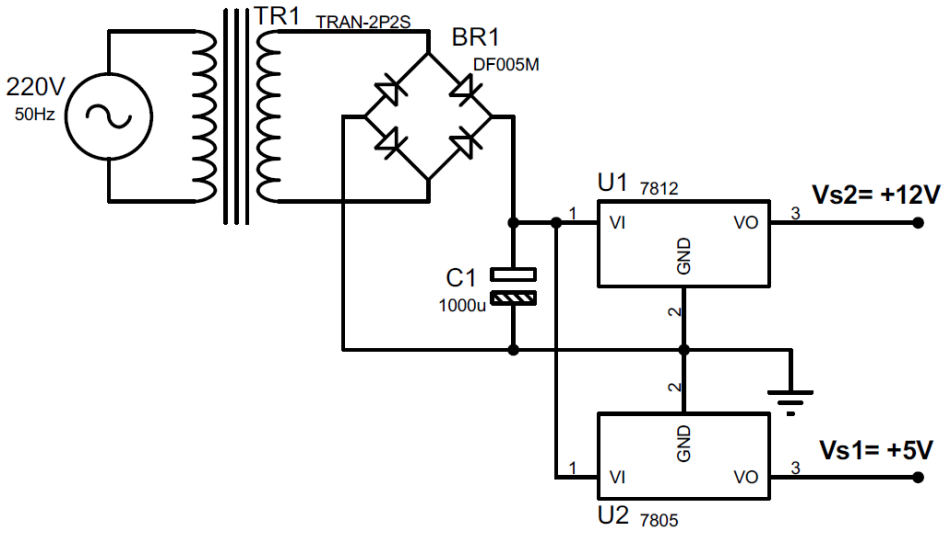



**Figure III.2 :** Circuit d'alimentation régulée.

Lorsqu'on alimente le circuit à travers le réseau (220V, 50Hz), le transformateur abaisse cette tension à la valeur désirée (24V pour ce circuit) en gardant la même fréquence.

Puis, le pont de diode fait un redressement double alternance qui inverse la composante négative.

Ensuite, la tension redressée passe à travers le filtre capacitif composé du condensateur (C1) qui a pour but de maintenir la tension près de la valeur maximale 24V.

Enfin, la tension filtrée passe par des régulateurs qui permettent de maintenir les valeurs désirées (tensions de sortie) quelle que soit la variation des paramètres dans la marge de tolérance.

### III.3.2 Commande et traitement

Cette partie est composée du microcontrôleur PIC 16F877, d'un potentiomètre de 1kΩ (RV1) et de trois boutons poussoirs.

Le microcontrôleur contient le programme écrit en language C qui permet de commander les transistors du hacheur quatre quadrants (circuit de puissance) et l'affichage sur l'écran LCD.

Les boutons poussoirs raccordés aux broches 33, 34 et 35 permettent de gérer le sens de rotation du moteur ou bien de l'arrêter. Lorsque ces boutons sont au repos, les broches associées sont reliées à 5V. Dés qu'on appuie sur l'un des boutons, la broche associée sera reliée à la masse et le microcontrôleur exécute les instructions liées à ce changement d'état. Ces boutons permettent également à travers le microcontrôleur d'afficher le sens de rotation du moteur (Left ou Right) sur l'afficheur LCD.

Le potentiomètre permet de varier la vitesse de rotation du moteur en agissant sur le rapport cyclique α qui varie entre 0% (moteur à l'arrêt) et 100% (moteur tournant à vitesse maximale dans le sens indiqué). Le potentiomètre est relié à la broche 2 (analogique), ce qui permet de varier la tension sur la broche entre 0V qui représente 0% et 5V qui représente 100%.

Afin de varier cette tension, le module ADC doit être initialisé puis indiquer la broche utilisée et enfin lire la valeur sur cette broche comme suit :

```
ADC_Init();                 // Initialiser le module ADC.
ADC_Get_Sample(0);  // Lire la valeur de la broche 0 (ou RA0/AN0).
ADC_Read(0);              // Lire la valeur sur la broche 0.
```

La tension aux bornes du potentiomètre qui représente le rapport cyclique est affichée sur l'écran LCD en pourcentage.

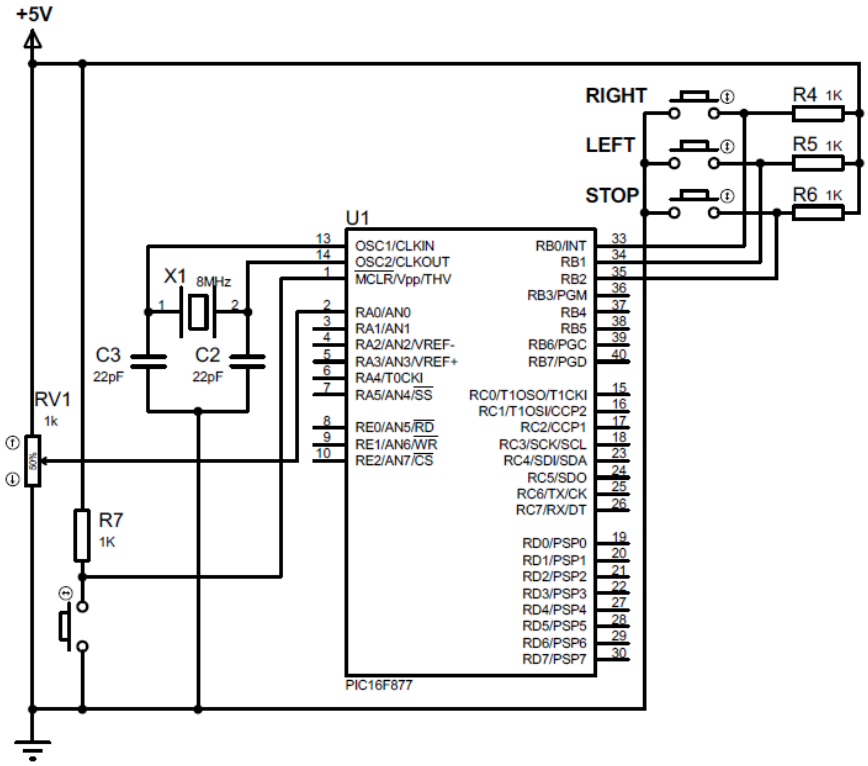



**Figure III.3 :** Circuit de commande.

### ♣ ***Remarque :***

Afin de permettre au PIC 16F877 de fonctionner convenablement, il est impératif de fournir les connexions de base suivantes :

- Alimentation : les broches 11 et 32 ($V_{DD}$) doivent être connectées à 5V et les broches 12 et 31 ($V_{SS}$) doivent être connectées à la masse (0V).
- Signal d'horloge : En plus de l'oscillateur intégré au microcontrôleur, un oscillateur externe doit être connecté aux broches 13 et 14 (un quartz associé à deux condensateurs) pour générer un signal d'horloge externe qui stabilise et détermine la fréquence de fonctionnement du microcontrôleur.
- Réinitialisation : un '1' logique doit être appliqué à la broche 1 ($\overline{MCLR}$). Le bouton poussoir reliant $\overline{MCLR}$ à la masse permet un retour au fonctionnement normal du programme en cas de problèmes.

Ces connexions sont valables uniquement pour les circuits simples. Autrement, on fait recours à d'autres types de composants et de connexions.

### III.3.3 Affichage

L'affichage est réalisé grâce à un afficheur LCD (Liquid Crystal Display). La commande issue du microcontrôleur permet d'afficher le sens de rotation du moteur (selon le bouton appuyé) ainsi que le rapport cyclique en pourcentage (selon la variation du potentiomètre).

| Broche | Nom | Description |
|---|---|---|
| 1 | Vss | Masse (0V) |
| 2 | Vdd | Alimentation (+5V) |
| 3 | Vee | Contraste de l'écran (entre 0 et 5V) |
| 4 | RS | Registre de Sélection (0 : commande ; 1 : donnée) |
| 5 | RW | Lecture Ecriture (0 : écriture ; 1 : lecture) |
| 6 | E | Validation |
| 7 | D0 | D0 à D7 pour un transfert sur 8bits.<br>D4 à D7 pour un transfert sur 4bits |
| 8 | D1 | |
| 9 | D2 | |
| 10 | D3 | |
| 11 | D4 | |
| 12 | D5 | |
| 13 | D6 | |
| 14 | D7 | |
| 15 | A | LED de rétro-éclairage<br>A : Anode ; K : Cathode |
| 16 | K | |

**Tableau III.1 :** Description des broches de l'afficheur LCD.

Pour pouvoir afficher les caractères, on doit d'abord définir les variables de connexion et les variables de direction du module LCD (code à copier dans la librairie) puis l'initialiser et ensuite utiliser les commandes d'affichage :

```
Lcd_Init();                          // Initialiser l'afficheur LCD.
Lcd_Cmd(_LCD_CLEAR);                 // Effacer l'affichage.
Lcd_Cmd(_LCD_CURSOR_OFF); // Désactiver le curseur.
Lcd_Out(1,1,''texte''); // Ecrire ''texte'' en 1ère ligne 1ère colonne.
intTostr(a,txt);          // convertir le nombre entier a en chaine de caractères.
```

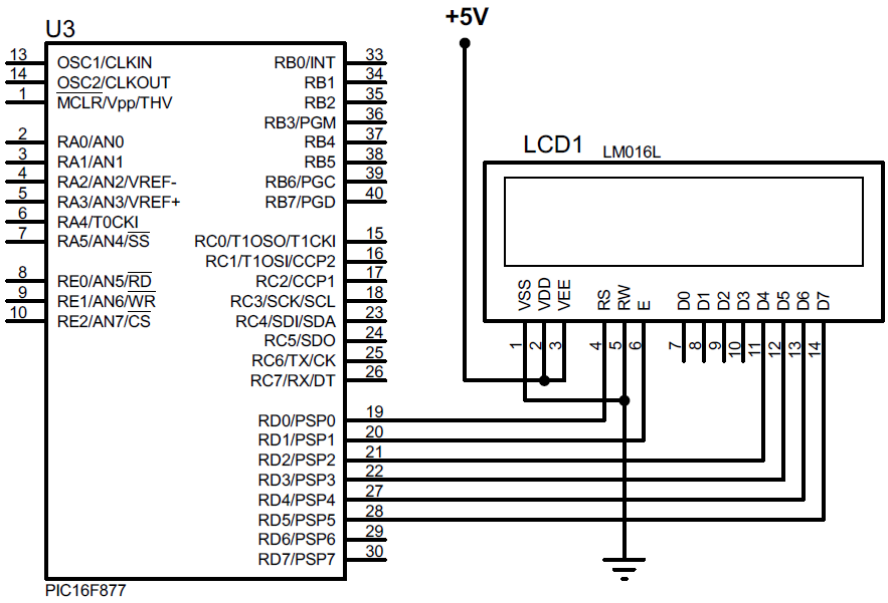



**Figure III.4 :** Circuit de l'afficheur LCD (associé au PIC).

### III.3.4 Circuit de puissance

Lorsque la broche 17 émet un signal PWM1, les transistors T3 et T2 reçoivent simultanément un signal sur leurs bases permettant leurs saturations et ainsi le passage d'un courant (émetteur-collecteur) faisant tourner le moteur dans le sens 1 (RIGHT). Pendant ce temps, la broche 16 reste à 0V (aucun signal).

Au niveau du programme, le module PWM1 doit être initialisé à une fréquence donnée, puis modifier le rapport cyclique, ensuite activer le signal PWM1 et enfin mettre fin au signal. Les instructions sont les suivantes :

```
PWM1_Init(500);      //Initialiser le module PWM1 à 500Hz.
PWM1_Set_Duty(r);  //Modifier le rapport cyclique 'r' du signal PWM1.
PWM1_Start();         //Activer le signal PWM1.
PWM1_Stop();        //Mettre fin au signal PWM1
```

Lorsque la broche 16 émet un signal PWM2, les transistors T1 et T4 reçoivent simultanément un signal sur leurs bases permettant leurs saturations et ainsi le passage d'un courant (émetteur-collecteur) faisant tourner le moteur dans le sens 2 (LEFT). Pendant ce temps, la broche 17 reste à 0V (aucun signal).

Au niveau du programme, la procédure pour le signal PWM2 est identique au signal PWM1. Il suffit seulement de remplacer le '1' de PWM1 par '2'.

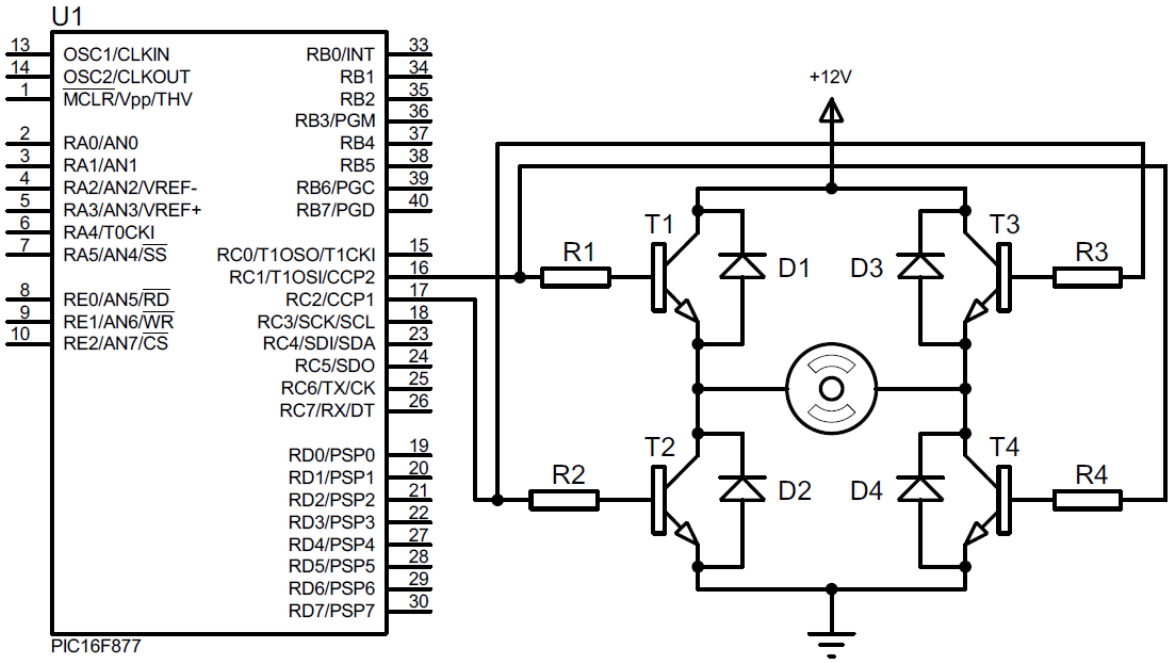



**Figure III.5 :** circuit de puissance.

## III.4 Résumé détaillé du fonctionnement de la carte du variateur de vitesse

Lorsqu'on branche la carte sur le secteur (220V/50Hz) le transformateur (220V/24V) abaisse la tension à 24V en gardant la même fréquence. Le pont de diodes redresse cette tension pour des cycles doubles alternances positives. Pour avoir une tension continue, le condensateur (de filtrage) réduit les ondulations et lisse la tension de manière à se rapprocher le plus possible de la tension maximale (24V continu). Pour éviter les variations de tension au niveau de l'alimentation du circuit, on utilise des régulateurs de tension qui permettent de délivrer une tension constante quelque soient les variations des paramètres extérieurs (dans une marge donnée). On utilise un régulateur de 12V pour alimenter le hacheur quatre quadrants (commande du moteur) et un deuxième régulateur de 5V pour alimenter les autres parties du circuit.

Pour le bon fonctionnement du microcontrôleur, il est nécessaire de lui rajouter, en plus de l'alimentation, un oscillateur externe (un quartz et deux condensateurs) ainsi qu'une réinitialisation. Cette dernière est associée à un bouton poussoir pour assurer une réinitialisation en cas de problème dans le programme.

Le programme est codé par le compilateur *mikroC PRO for PIC* et injecté dans le PIC 16F877 avec le programmateur *Pickit2*.

Lorsqu'on appuis sur le bouton (R), le moteur tourne à droite (sens horaire). Cela se fait grâce au signal PWM1 qui active les deux transistors (T1 et T4), les deux autres transistors restent toujours bloqués. Pour varier la vitesse dans ce sens, il suffit de tourner le potentiomètre (varier sa valeur entre 0 et 100%). Le sens de rotation et la valeur du potentiomètre en pourcentage sont affichés sur l'écran LCD.

Lorsqu'on appuie sur le bouton (L), le moteur tourne à gauche (sens antihoraire ou trigonométrique). Cela se fait grâce au signal PWM2 qui active les deux transistors (T3 et T2), les deux autres transistors restent toujours bloqués. Pour varier la vitesse dans ce sens, il suffit de tourner le potentiomètre (varier sa valeur entre 0 et 100%). Le sens de rotation et la valeur du potentiomètre en pourcentage sont affichés sur l'écran LCD.

Lorsqu'on appuie sur le bouton (S), le moteur est arrêté.

Le schéma final et complet du circuit du variateur de vitesse est présenté dans la figure suivante :

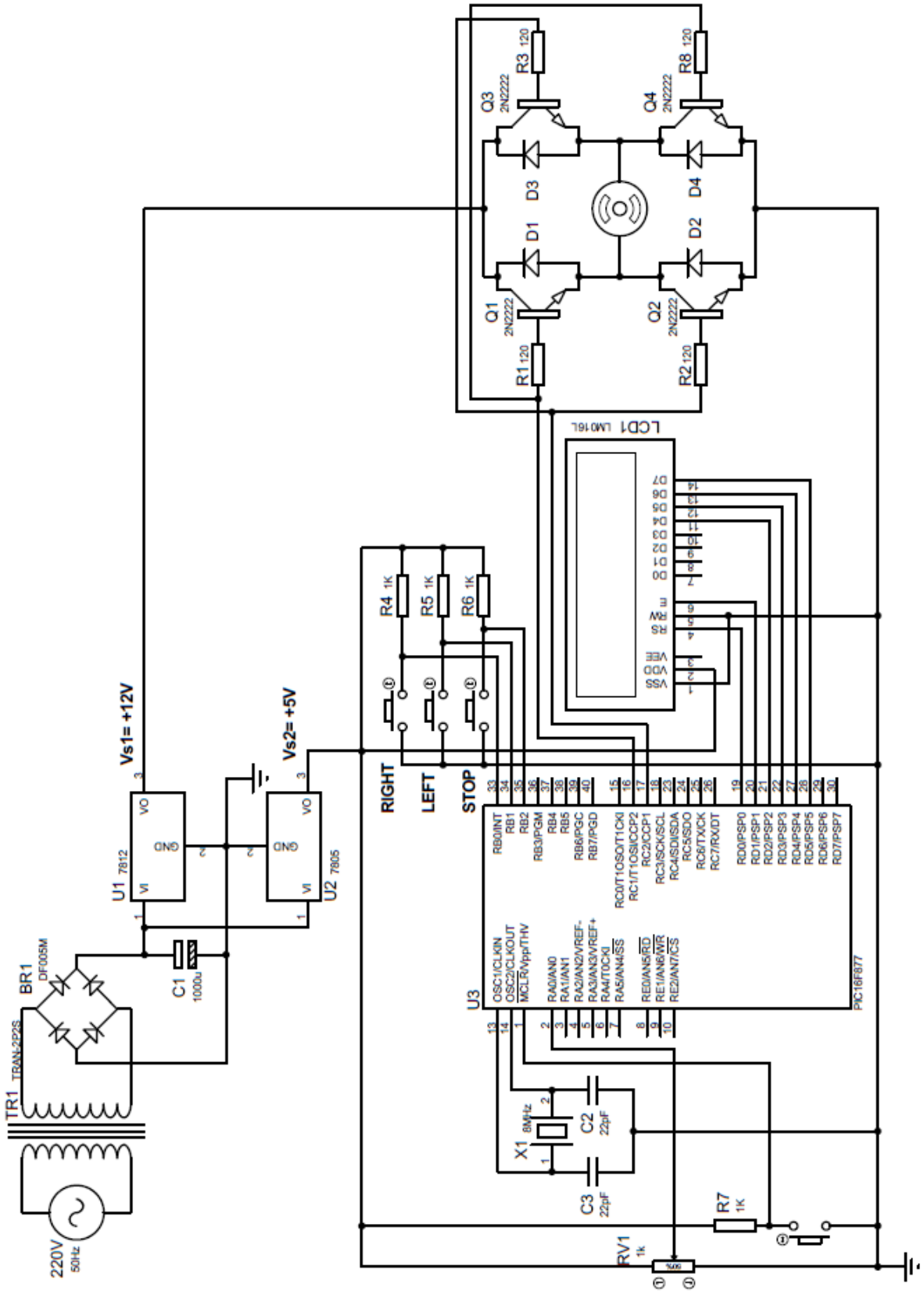
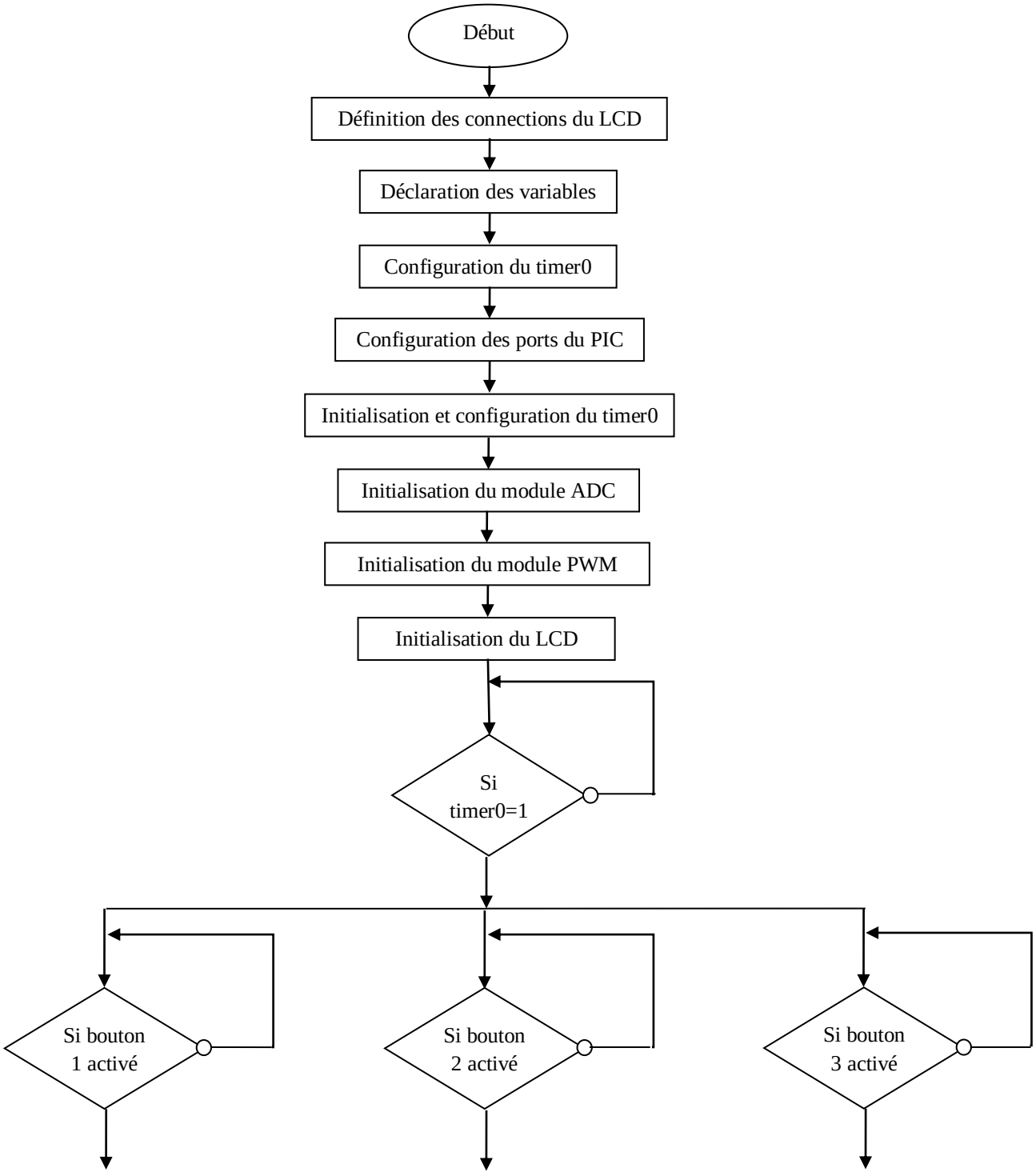
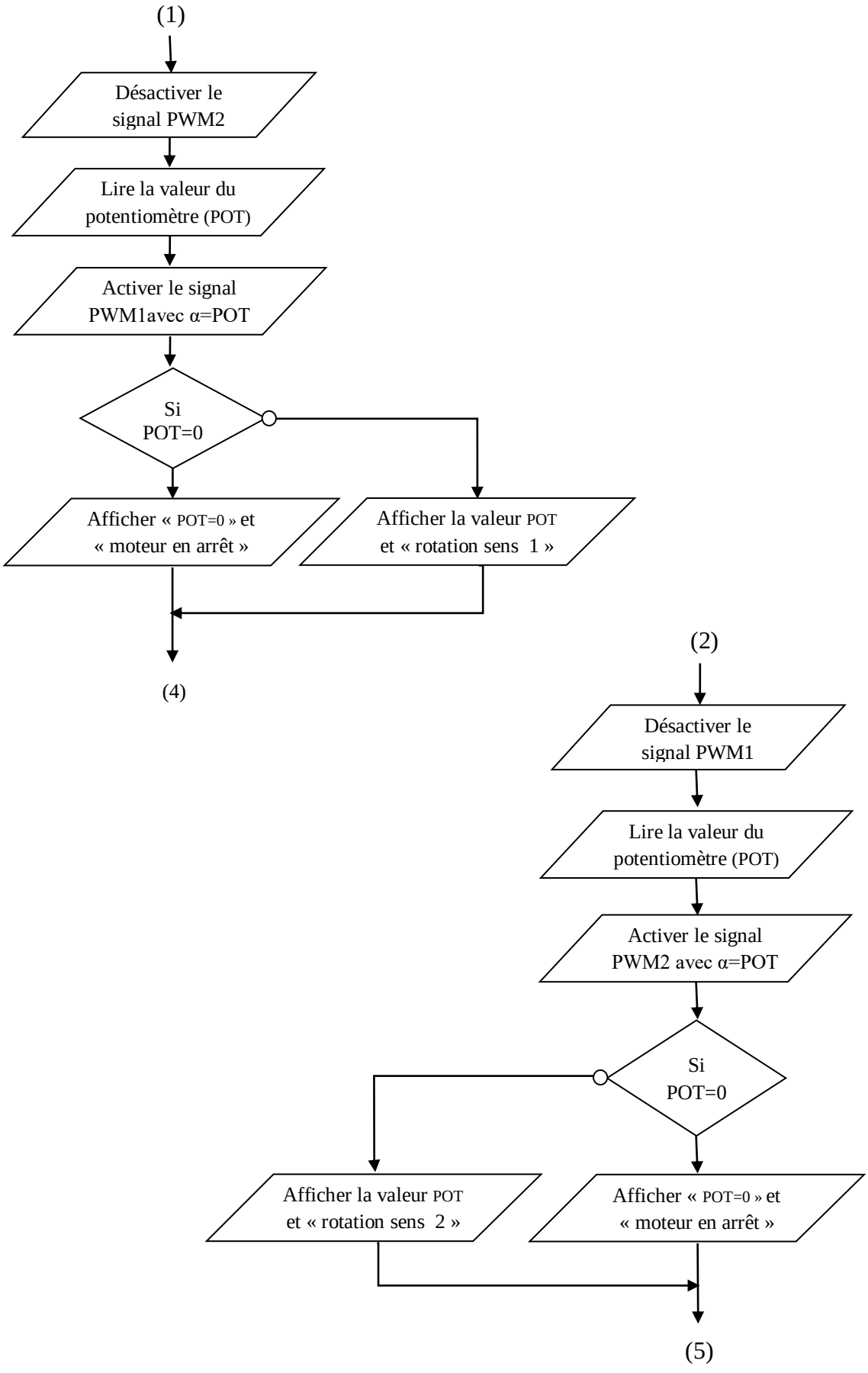


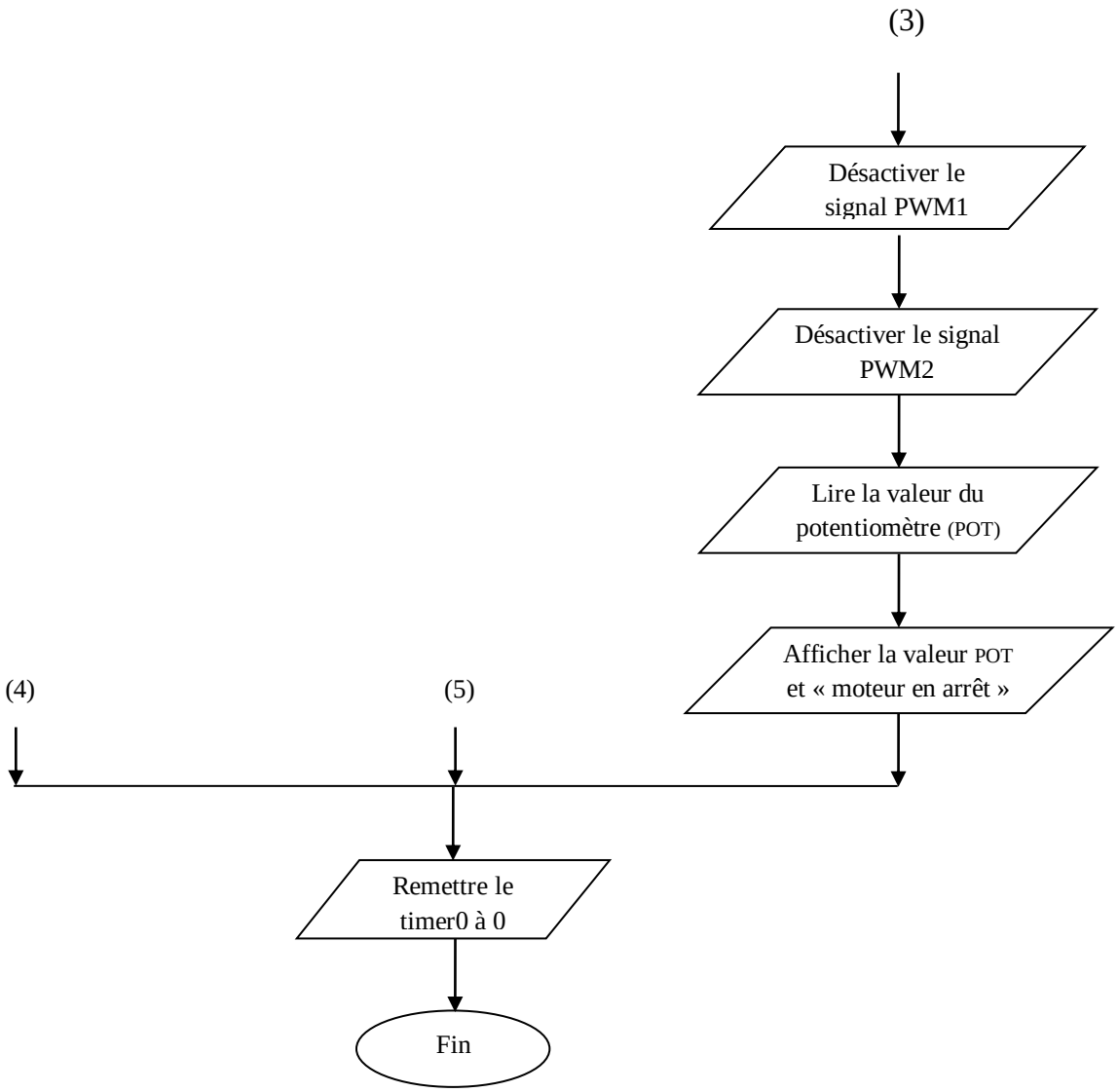

**Figure III.6 :** Circuit complet du variateur de vitesse.

## III.5 Organigramme de programmation

L'organigramme qui illustre l'enchainement des instructions réalisées par le programme contenu dans le PIC 16F877 est le suivant :




(1)
Désactiver le signal PWM2
Lire la valeur du potentiomètre (POT)
Activer le signal PWM1avec α=POT
Si POT=0
Afficher « POT=0 » et « moteur en arrêt »
Afficher la valeur POT et « rotation sens 1 »
(4)
(2)
Désactiver le signal PWM1
Lire la valeur du potentiomètre (POT)
Activer le signal PWM2 avec α=POT
Si POT=0
Afficher la valeur POT et « rotation sens 2 »
Afficher « POT=0 » et « moteur en arrêt »
(5)




## III.6 Réalisation pratique

La réalisation passe par plusieurs étapes :

### III. 6. 1 Liste des composants

Les composants utilisés dans la réalisation pratique sont :

| Composant | Valeur | Référence | Quantité |
|---|---|---|---|
| Régulateur | 5V | 7805 | 01 |
| Microcontrôleur PIC | 20MHz | PIC 16F877 | 01 |
| Afficheur LCD | 2x16 | LM016L | 01 |
| Pont H (Hacheur) | - | L293D | 01 |
| Resistances | 100Ω | - | 05 |
| Potentiomètre | 1kΩ | - | 01 |
| Boutons poussoirs | - | - | 04 |
| Oscillateur à quartz | 8MHz | CQ 8.000 | 01 |
| Condensateurs | 22pF | (céramique) | 02 |
| Transformateur | 220VAC/24VDC | - | 01 |
| Moteur | 12VDC | - | 01 |

**Tableau III.2 :** Composants utilisés pour la réalisation du circuit final.

En plus des composants, on doit prévoir :

- Des fils.
- Une plaque d'essai.
- Une carte de prototypage.
- Un fer à souder et de l'étain.
- Un multimètre (pour tester la continuité des connexions et éviter d'éventuels courts circuits).
- Un programmateur pour PIC (Pickit 2).
- Logiciels : Proteus ISIS, Proteus ARES, mikroC PRO for PIC, Pickit 2.

### III.6.2 Tests sur plaque d'essai

Une fois la simulation et le programme sont parfaitement fonctionnels sur logiciel (théorie), on passe aux tests du circuit sur plaque d'essai pour s'assurer de son bon fonctionnement en pratique.

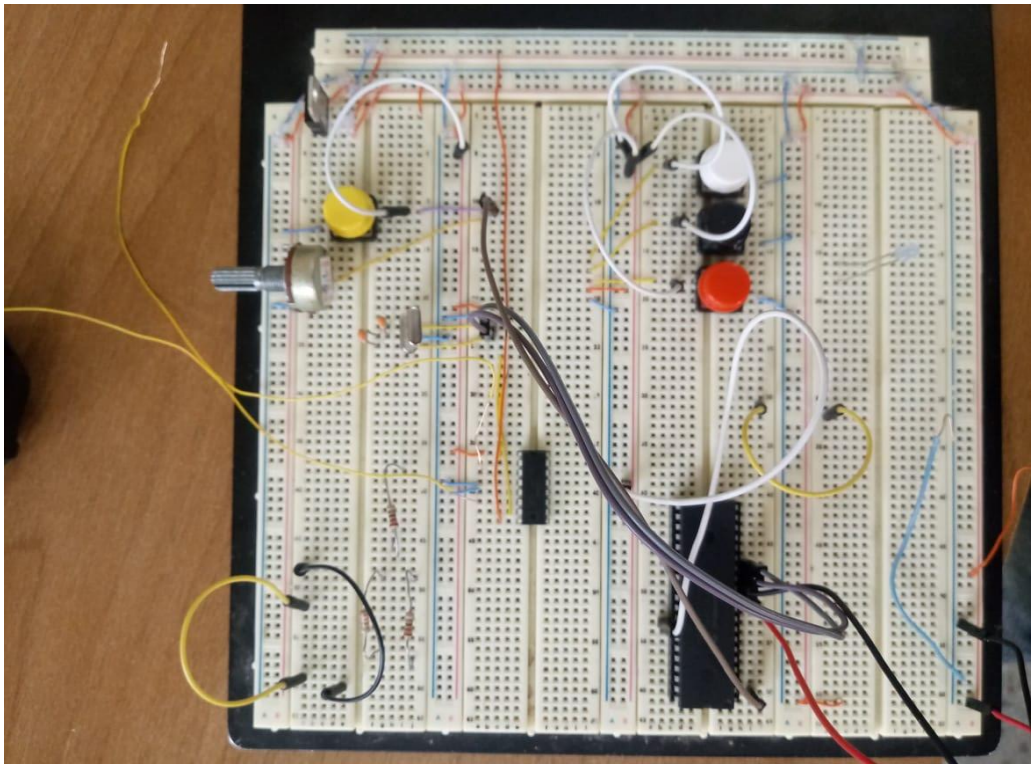


**Figure III.7 :** Réalisation et test du circuit sur plaque d'essai.

### III.6.3 Réalisation du circuit imprimé

Après avoir testé le bon fonctionnement du circuit sur la plaque d'essai, on est passé à la conception du circuit imprimé sur le logiciel ARES. Ce dernier nous permet à partir du circuit réalisé sur ISIS de faire le routage des pistes du circuit.

La **Figure III.8** met en évidence la conception du circuit du variateur de vitesse réalisé sur le logiciel ARES.

La **Figure III.9** montre une visualisation 3D du montage final à obtenir après la réalisation de la carte avec le circuit imprimé associé.

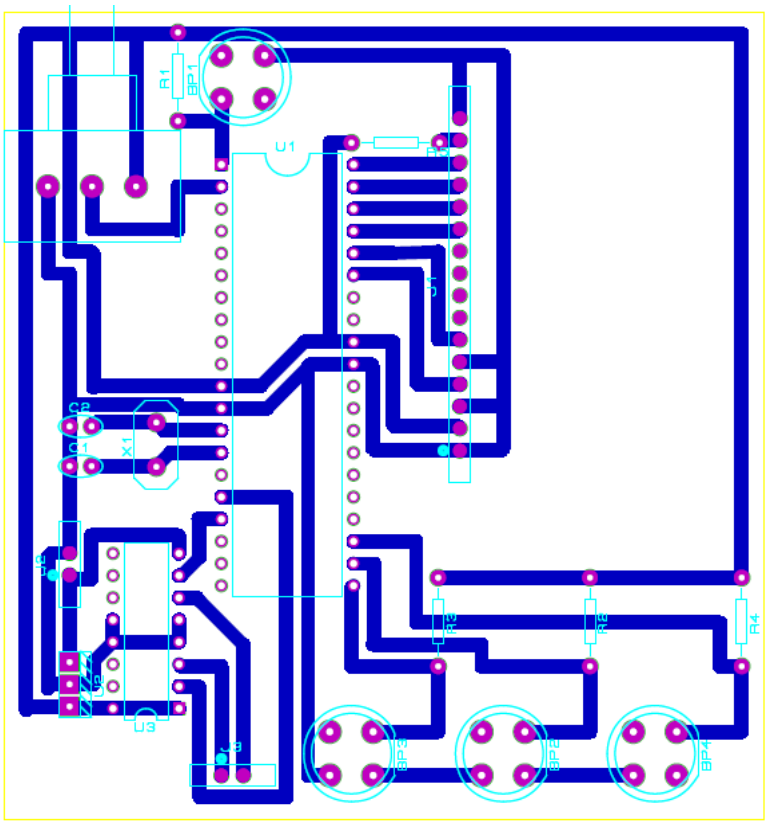



**Figure III.8 :** Conception (routage) du circuit sous ARES.

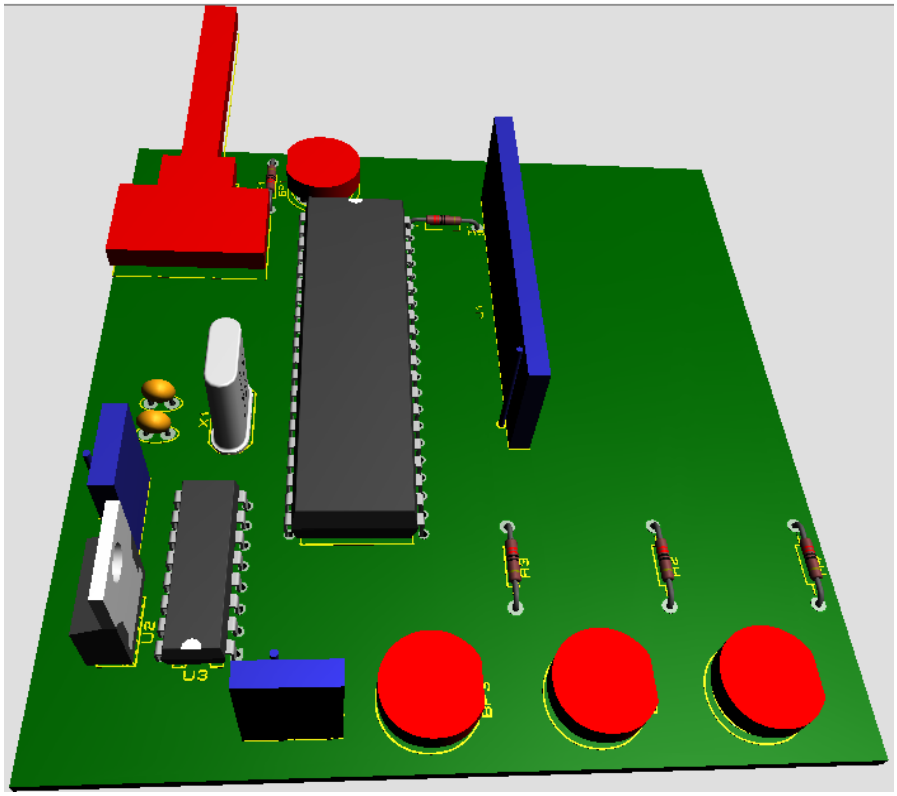



**Figure III.9 :** Visualisation 3D du circuit sous ARES.

### III.6.4 Réalisation de la carte finale

Pour réaliser la carte finale du variateur de vitesse du moteur à courant continu, nous avons utilisé une carte de prototypage perforée illustrée dans l'image suivante :

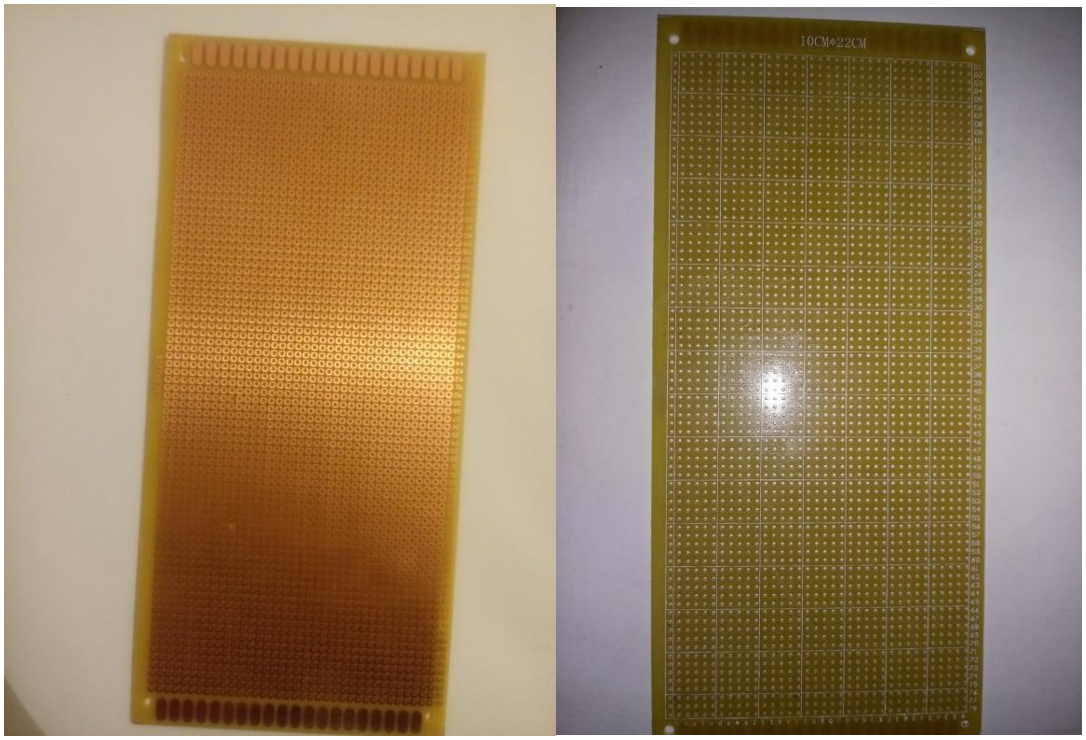



**Figure III.10 :** Carte de prototypage perforée.

Les soudures sur les plaques de prototypage perforées se font directement sur la carte sans passer par le tracé des pistes en cuivre sur la plaque en utilisant le circuit imprimé. Mais pour éviter les erreurs de connections lors de la soudure, nous avons utilisé le circuit imprimé pour effectuer un tracé des pistes sur papier calque avant de souder les composants comme illustré dans les figures suivantes :

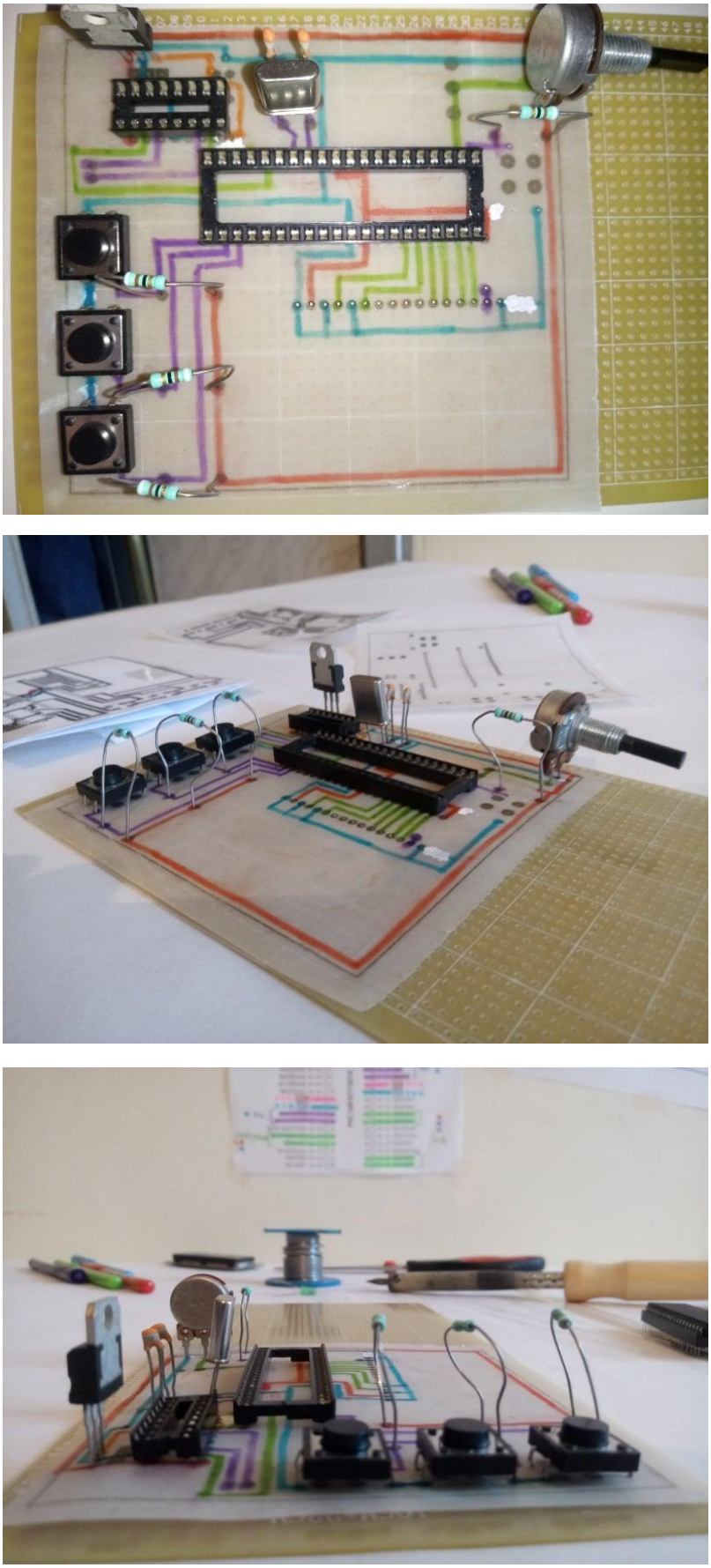


**Figure III.11 :** Utilisation du papier calque pour mettre en évidence les pistes.

### III.6.4.1 Soudure des composants sur la carte

Pour cette étape nous avons utilisé un fer à souder et de l'étain pour fixer et relier les différentes parties du circuit sur la carte. Une fois terminé, nous avons testé la continuité de chaque connexion à l'aide d'un multimètre pour éviter les faux contacts et s'assurer de la bonne continuité des connections.

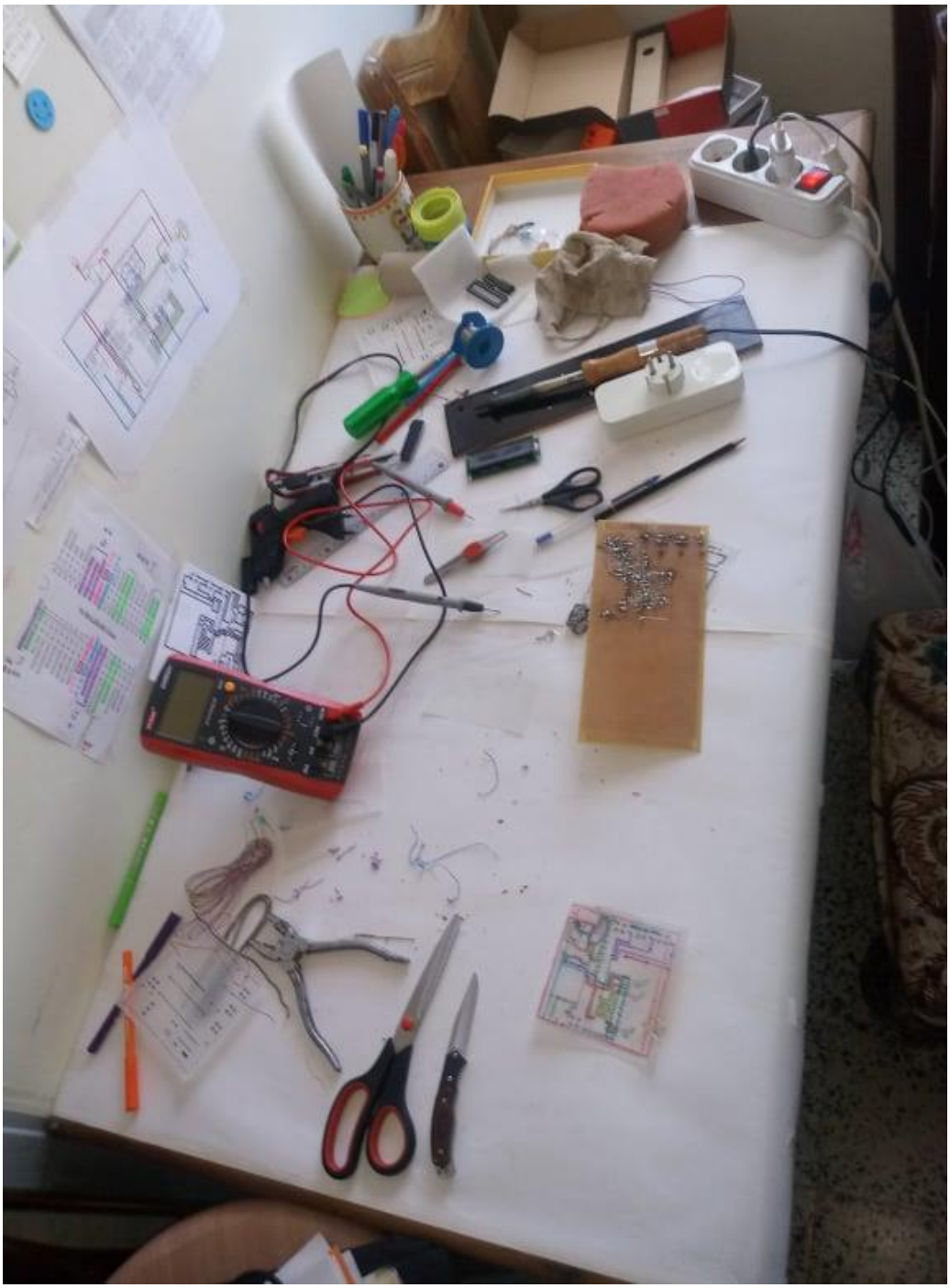
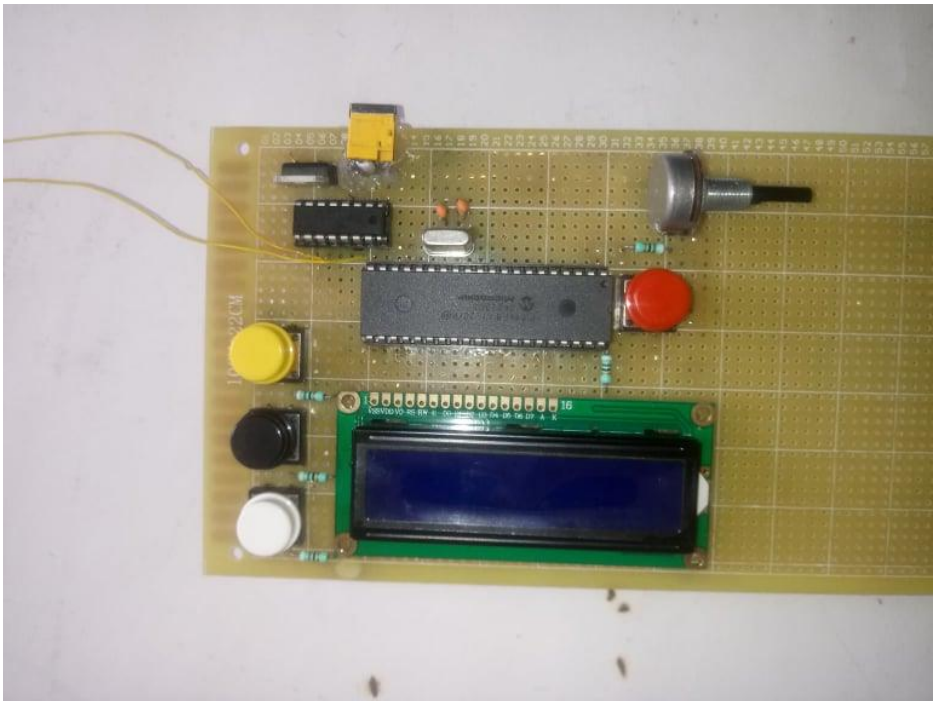


**Figure III.12 :** Soudure des composants et test des connexions.

### III.6.4.2 Résultat final de la réalisation de la carte du variateur de vitesse du moteur à courant continu par PIC 16F877

Le travail réalisé a aboutit à la réalisation de la carte du variateur de vitesse illustrée par les deux figures suivantes :



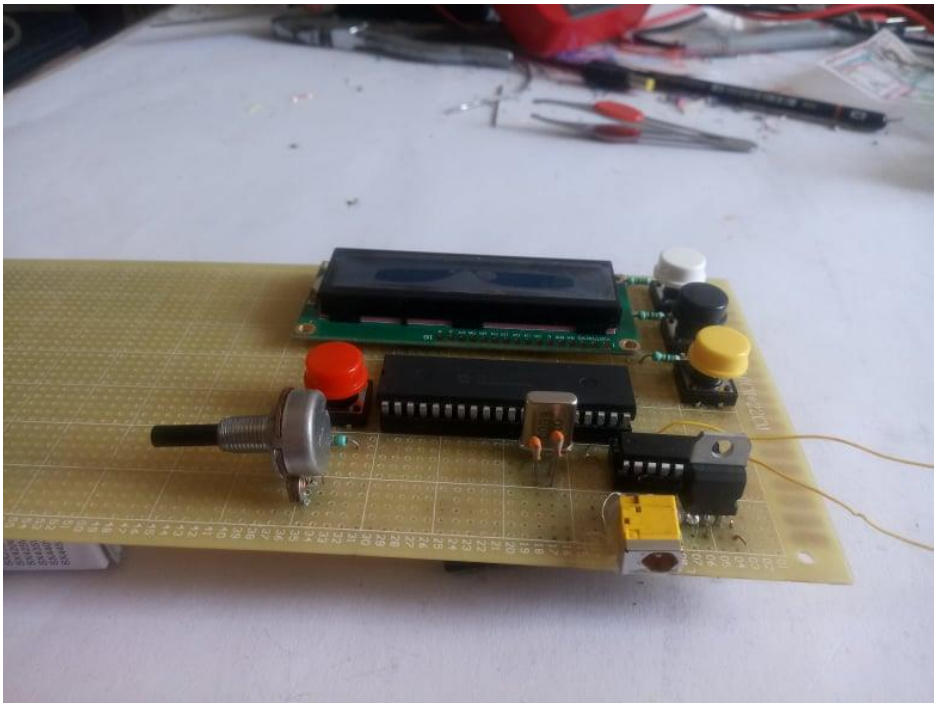


**Figure III.13 :** Carte finale du variateur de vitesse réalisée après soudure des composants..

## III.7 Conclusion

Dans ce chapitre, nous avons procédé à l'étude et à la réalisation de la carte du variateur de vitesse du moteur à CC par PIC 16F877. Nous avons d'abord étudié le fonctionnement de chaque partie du variateur de vitesse (alimentation, commande et traitement, affichage et enfin la partie puissance).

Ensuite nous avons expliqué les différentes étapes de réalisation de la carte du variateur de vitesse en commençant par la conception du circuit imprimé sur le logiciel ARES jusqu'à la réalisation de la carte finale.

# *Conclusion Générale*

# *Conclusion Générale*

Ce travail avait pour but de faire l'étude du circuit d'un variateur de vitesse pour un moteur à courant continu en utilisant le PIC 16F877 et procéder à la réalisation pratique du circuit électronique.

Il a fallu dans un premier temps expliquer certaines notions de bases sur les moteurs à courant continu, les hacheurs et le microcontrôleur PIC 16F877 afin de comprendre leurs fonctionnement et leurs rôle.

A l'aide du logiciel *Proteus ISIS* et du compilateur *mikroC PRO for PIC*, nous avons étudié et simulé chaque partie du circuit du variateur de vitesse. Cette simulation, qui est interactive, nous a permit d'ajuster les différents paramètres du circuit et d'analyser les possibilités de réalisation.

Dans la partie réalisation, nous avons utilisé le logiciel *Proteus ARES* pour tracer les pistes du circuit. Celles-ci ont été placées sur la plaque perforée pour positionner correctement les composants avant d'entamer leur soudure.

L'utilisation de la méthode de modulation de largeur d'impulsion associée à un hacheur quatre quadrants à base de transistors nous a permit d'optimiser la commande du moteur à CC en réduisant la consommation d'énergie tout en ayant un rendement élevé.

Toutefois, notre étude n'a porté que sur la variation de vitesse des moteurs à courant continu. L'étude pourrait s'étendre sur d'autres types de charge.

On pourrait aller plus loin dans cette étude en apportant des améliorations considérables et très avantageuses comme par exemple la commande de vitesse à distance ou bien la réalisation d'une carte universelle qui prend en charge n'importe quel type de moteur.

# *Références Bibliographiques*

# *Références Bibliographiques*

**[1] :** Théodore Wildi, Gilbert Sybille, "*Electrotechnique*", De Boeck, 2015.

**[2] :** Julio Sanchez, Maria P. Canton, "*Microcontroller Programming The Microchip PIC*", CRC Press, 2018.

**[3] :** Dogan Ibrahim, "*Advanced PIC Microcontroller Projects In C From USB to ZIGBEE With the PIC 18F Series*", Newnes, 2011.

**[4] :** Martin P. Bates, "*Programming 8-bit PIC Microcontrollers in C With Interactive Hardware Simulation*", Newnes, 2008.

**[5] :** Microchip Technology, "*PICmicro Mid-Range MCU Family Reference Manual*".

**[6] :** Martin Bates, "*Interfacing PIC Microcontrollers Embedded Design By Interactive Simulation*", Elsevier, 2013.

**[7] :** Microchip Technology, "*PIC16F87X : 28/40-pin 8-Bit CMOS FLASH Microcontrollers*".

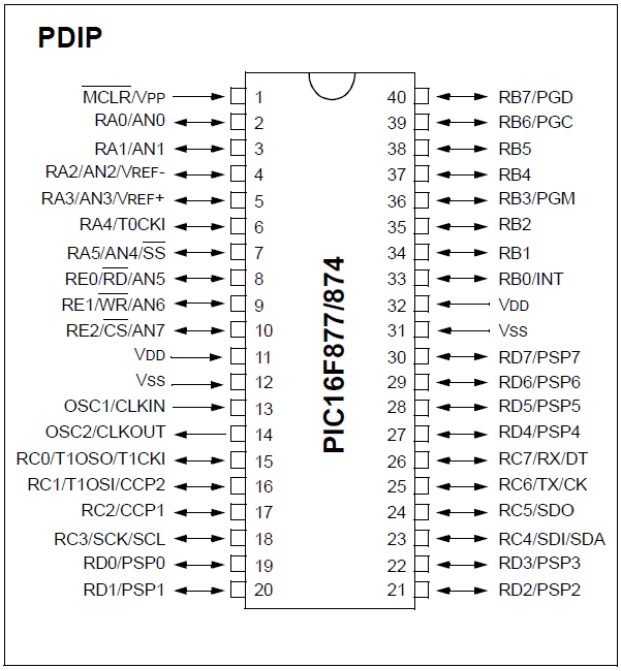
# *Annexe*

# Annexe

## Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory, Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 μA typical @ 3V, 32 kHz
  - < 1 μA typical standby current

Pin Diagram



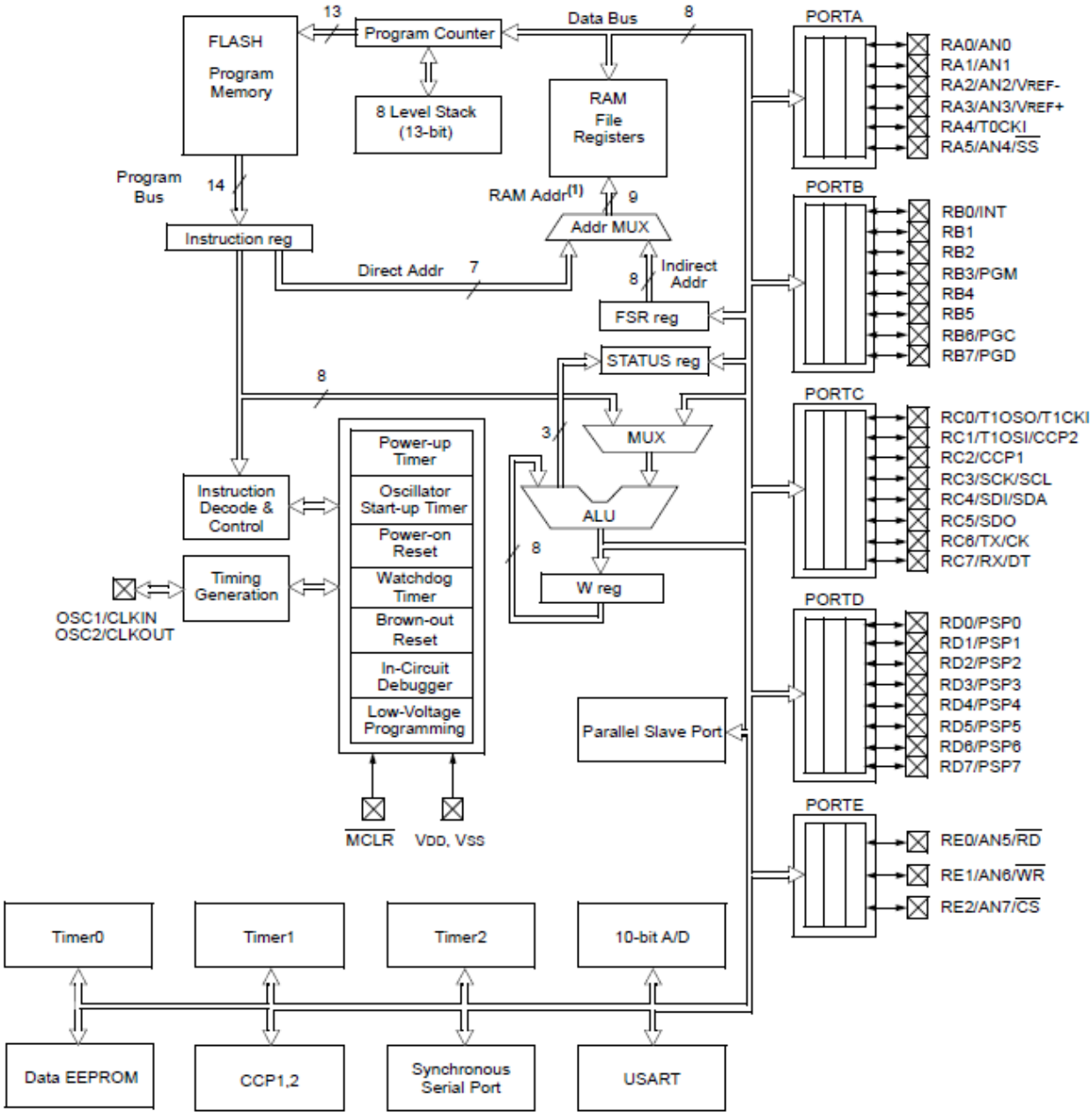

## Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI (Master mode) and $I^2C$™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)

Brown-out detection circuitry for Brown-out Reset (BOR)

**Source :** DataSheet du PIC16F877.

- **PIC PIC16F877 BLOCK DIAGRAM:**




**Note 1:** Higher order bits are from the STATUS register.

**Source :** DataSheet du PIC16F877.

➢ **PIC16F877/876 REGISTER FILE MAP**

| Bank 0 | File Address | Bank 1 | File Address | Bank 2 | File Address | Bank 3 | File Address |
|---|---|---|---|---|---|---|---|
| Indirect addr.(*) | 00h | Indirect addr.(*) | 80h | Indirect addr.(*) | 100h | Indirect addr.(*) | 180h |
| TMR0 | 01h | OPTION_REG | 81h | TMR0 | 101h | OPTION_REG | 181h |
| PCL | 02h | PCL | 82h | PCL | 102h | PCL | 182h |
| STATUS | 03h | STATUS | 83h | STATUS | 103h | STATUS | 183h |
| FSR | 04h | FSR | 84h | FSR | 104h | FSR | 184h |
| PORTA | 05h | TRISA | 85h | | 105h | | 185h |
| PORTB | 06h | TRISB | 86h | PORTB | 106h | TRISB | 186h |
| PORTC | 07h | TRISC | 87h | | 107h | | 187h |
| PORTD(1) | 08h | TRISD(1) | 88h | | 108h | | 188h |
| PORTE(1) | 09h | TRISE(1) | 89h | | 109h | | 189h |
| PCLATH | 0Ah | PCLATH | 8Ah | PCLATH | 10Ah | PCLATH | 18Ah |
| INTCON | 0Bh | INTCON | 8Bh | INTCON | 10Bh | INTCON | 18Bh |
| PIR1 | 0Ch | PIE1 | 8Ch | EEDATA | 10Ch | EECON1 | 18Ch |
| PIR2 | 0Dh | PIE2 | 8Dh | EEADR | 10Dh | EECON2 | 18Dh |
| TMR1L | 0Eh | PCON | 8Eh | EEDATH | 10Eh | Reserved(2) | 18Eh |
| TMR1H | 0Fh | | 8Fh | EEADRH | 10Fh | Reserved(2) | 18Fh |
| T1CON | 10h | | 90h | | 110h | | 190h |
| TMR2 | 11h | SSPCON2 | 91h | | 111h | | 191h |
| T2CON | 12h | PR2 | 92h | | 112h | | 192h |
| SSPBUF | 13h | SSPADD | 93h | | 113h | | 193h |
| SSPCON | 14h | SSPSTAT | 94h | | 114h | | 194h |
| CCPR1L | 15h | | 95h | | 115h | | 195h |
| CCPR1H | 16h | | 96h | | 116h | | 196h |
| CCP1CON | 17h | | 97h | General Purpose Register 16 Bytes | 117h | General Purpose Register 16 Bytes | 197h |
| RCSTA | 18h | TXSTA | 98h | | 118h | | 198h |
| TXREG | 19h | SPBRG | 99h | | 119h | | 199h |
| RCREG | 1Ah | | 9Ah | | 11Ah | | 19Ah |
| CCPR2L | 1Bh | | 9Bh | | 11Bh | | 19Bh |
| CCPR2H | 1Ch | | 9Ch | | 11Ch | | 19Ch |
| CCP2CON | 1Dh | | 9Dh | | 11Dh | | 19Dh |
| ADRESH | 1Eh | ADRESL | 9Eh | | 11Eh | | 19Eh |
| ADCON0 | 1Fh | ADCON1 | 9Fh | | 11Fh | | 19Fh |
| General Purpose Register 96 Bytes | 20h | General Purpose Register 80 Bytes | A0h | General Purpose Register 80 Bytes | 120h | General Purpose Register 80 Bytes | 1A0h |
| | 7Fh | accesses 70h-7Fh | EFh F0h FFh | accesses 70h-7Fh | 16Fh 170h 17Fh | accesses 70h - 7Fh | 1EFh 1F0h 1FFh |

☐ Unimplemented data memory locations, read as '0'.
* Not a physical register.

**Note 1:** These registers are not implemented on the PIC16F876.
**2:** These registers are reserved, maintain these registers clear.

**Source :** DataSheet du PIC16F877.

# Annexe

## ➢ PORTA FUNCTIONS:

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RA0/AN0 | bit0 | TTL | Input/output or analog input. |
| RA1/AN1 | bit1 | TTL | Input/output or analog input. |
| RA2/AN2 | bit2 | TTL | Input/output or analog input. |
| RA3/AN3/$V_{REF}$ | bit3 | TTL | Input/output or analog input or $V_{REF}$. |
| RA4/T0CKI | bit4 | ST | Input/output or external clock input for Timer0. Output is open drain type. |
| RA5/$\overline{SS}$/AN4 | bit5 | TTL | Input/output or slave select input for synchronous serial port or analog input. |

Legend: TTL = TTL input, ST = Schmitt Trigger input

## ➢ PORTB FUNCTIONS:

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RB0/INT | bit0 | TTL/ST(1) | Input/output pin or external interrupt input. Internal software programmable weak pull-up. |
| RB1 | bit1 | TTL | Input/output pin. Internal software programmable weak pull-up. |
| RB2 | bit2 | TTL | Input/output pin. Internal software programmable weak pull-up. |
| RB3/PGM(3) | bit3 | TTL | Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up. |
| RB4 | bit4 | TTL | Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. |
| RB5 | bit5 | TTL | Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. |
| RB6/PGC | bit6 | TTL/ST(2) | Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming clock. |
| RB7/PGD | bit7 | TTL/ST(2) | Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming data. |

Legend: TTL = TTL input, ST = Schmitt Trigger input

**Note 1:** This buffer is a Schmitt Trigger input when configured as the external interrupt.

**2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.

**3:** Low Voltage ICSP Programming (LVP) is enabled by default, which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

**Source :** DataSheet du PIC16F877.

## ➢ PORTC FUNCTIONS:

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RC0/T1OSO/T1CKI | bit0 | ST | Input/output port pin or Timer1 oscillator output/Timer1 clock input. |
| RC1/T1OSI/CCP2 | bit1 | ST | Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output. |
| RC2/CCP1 | bit2 | ST | Input/output port pin or Capture1 input/Compare1 output/ PWM1 output. |
| RC3/SCK/SCL | bit3 | ST | RC3 can also be the synchronous serial clock for both SPI and I2C modes. |
| RC4/SDI/SDA | bit4 | ST | RC4 can also be the SPI Data In (SPI mode) or data I/O (I2C mode). |
| RC5/SDO | bit5 | ST | Input/output port pin or Synchronous Serial Port data output. |
| RC6/TX/CK | bit6 | ST | Input/output port pin or USART Asynchronous Transmit or Synchronous Clock. |
| RC7/RX/DT | bit7 | ST | Input/output port pin or USART Asynchronous Receive or Synchronous Data. |

Legend: ST = Schmitt Trigger input

## ➢ PORTD FUNCTIONS:

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RD0/PSP0 | bit0 | ST/TTL(1) | Input/output port pin or parallel slave port bit0. |
| RD1/PSP1 | bit1 | ST/TTL(1) | Input/output port pin or parallel slave port bit1. |
| RD2/PSP2 | bit2 | ST/TTL(1) | Input/output port pin or parallel slave port bit2. |
| RD3/PSP3 | bit3 | ST/TTL(1) | Input/output port pin or parallel slave port bit3. |
| RD4/PSP4 | bit4 | ST/TTL(1) | Input/output port pin or parallel slave port bit4. |
| RD5/PSP5 | bit5 | ST/TTL(1) | Input/output port pin or parallel slave port bit5. |
| RD6/PSP6 | bit6 | ST/TTL(1) | Input/output port pin or parallel slave port bit6. |
| RD7/PSP7 | bit7 | ST/TTL(1) | Input/output port pin or parallel slave port bit7. |

Legend: ST = Schmitt Trigger input, TTL = TTL input

**Note 1:** Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

**Source :** DataSheet du PIC16F877.

- **PORTE FUNCTIONS:**

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RE0/$\overline{RD}$/AN5 | bit0 | ST/TTL(1) | I/O port pin or read control input in Parallel Slave Port mode or analog input: RD<br>1 = Idle<br>0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected) |
| RE1/$\overline{WR}$/AN6 | bit1 | ST/TTL(1) | I/O port pin or write control input in Parallel Slave Port mode or analog input: WR<br>1 = Idle<br>0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected) |
| RE2/$\overline{CS}$/AN7 | bit2 | ST/TTL(1) | I/O port pin or chip select control input in Parallel Slave Port mode or analog input: CS<br>1 = Device is not selected<br>0 = Device is selected |

Legend: ST = Schmitt Trigger input, TTL = TTL input

**Note 1:** Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

**Source :** DataSheet du PIC16F877.

- **T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h):**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| bit 7 | | | | | | | bit 0 |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits

11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit

1 = Oscillator is enabled
0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain) bit 2 **T1SYNC**: Timer1 External Clock Input Synchronization Control bit

When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input

When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit

1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON**: Timer1 On bit

1 = Enables Timer1
0 = Stops Timer1

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

**Source :** DataSheet du PIC16F877.

- **T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h):**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| bit 7 | | | | | | | bit 0 |

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits

0000 = 1:1 Postscale
0001 = 1:2 Postscale
0010 = 1:3 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit

1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits

00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| - n = Value at POR | ’1’ = Bit is set | ’0’ = Bit is cleared | x = Bit is unknown |

**Source :** DataSheet du PIC16F877.

- **CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS: 17h/1Dh):**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | CCPxX | CCPxY | CCPxM3 | CCPxM2 | CCPxM1 | CCPxM0 |
| bit 7 | | | | | | | bit 0 |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY**: PWM Least Significant bits

Capture mode: Unused

Compare mode: Unused

PWM mode: These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)
0100 = Capture mode, every falling edge 0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge 0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

- **INTERACTION OF TWO CCP MODULES**

| CCPx Mode | CCPy Mode | Interaction |
|---|---|---|
| Capture | Capture | Same TMR1 time-base |
| Capture | Compare | The compare should be configured for the special event trigger, which clears TMR1 |
| Compare | Compare | The compare(s) should be configured for the special event trigger, which clears TMR1 |
| PWM | PWM | The PWMs will have the same frequency and update rate (TMR2 interrupt) |
| PWM | Capture | None |
| PWM | Compare | None |

**Source :** DataSheet du PIC16F877.