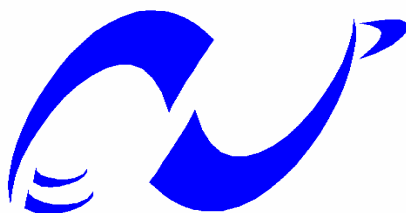


**RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE**  
**MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE**  
**SCIENTIFIQUE**



المدرسة الوطنية المتعددة التقنيات  
Ecole Nationale Polytechnique

**DÉPARTEMENT D'ÉLECTRONIQUE**

**MÉMOIRE DE MAGISTER EN ÉLECTRONIQUE**  
**OPTION : Énergie Solaire**

**Présenté par :**  
TOUBAL Abdelmoughni  
Ingénieur d'état C.U.de Médéa

*Réalisation d'une interface USB autour d'un  
microcontrôleur  
(PIC 18F4550) pour la commande d'un moteur à  
courant continu.*

<b>Président :</b>	M TRABELSI Mohamed	MC	(ENP)
<b>Rapporteur :</b>	M LARBES Chérif	MC	(ENP)
<b>Examineurs :</b>	M HADDADI Mourad	Pr	(ENP)
	M BOURI Mohamed	Dr. Ing	(EPF Lausanne)
	M MALEK Ali	MR	(CDER)

**Juin -2007**

## Remerciements

*Au terme de ce mémoire je tiens à remercier tout naturellement en premier lieu **DIEU Le Tout Puissant** qui nous a donné la force, le courage et la patience de bien mener ce travail.*

*Ce travail a été réalisé sous la direction de **M. LARBES chérif** notre promoteur, qu'il trouve ainsi l'expression de ma profonde reconnaissance pour son aide, ses encouragements et ses précieux conseils durant le déroulement de ce travail.*

*Je tiens à exprimer mes sincères remerciements à **tous les enseignants de PG Electronique option Energie Solaire** à l'Ecole Nationale Polytechnique, EL-HARRACH.*

*Je remercie **M TRABELSI**, qui a accepté d'assurer la présidence de notre jury ainsi que **M. HADDADI, M. BOURI et M. MALEK** d'avoir accepté l'évaluation de ce modeste travail.*

*Mes sentiments vont également à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce modeste travail.*

## *Dédicaces*

*Au delà des personnes, des lieux et des temps, je dédie ce travail avec ma profonde conviction, à tous ceux qui ont toujours cru sans doute à la science. Cette lumière qui éclaire les esprits et leurs permet de transcender les limites installées par les sociétés et les cultures dans leurs périodes de dégénérescence.*

*A ceux qui savent concrétiser leurs idées et travaillent pour réussir.*

*A ceux qui ne portent que du bien pour les autres.*

*A ceux qui ne vivent que pour la vérité.*

*Je dédie ce travail à mes **chers parents**.*

*A mes frères Ahmed, Abd-nour et Djamel-eddine, à mes soeurs.*

*A toute ma grande famille.*

*A tous mes amis de la promotion électronique 2003 du centre universitaire de Médéa sans exception.*

*A tous mes amis de la promotion PG électronique 2005 surtout « Energie solaire » de l'ENP EL-HARRACH.*

*A tous mes amis : Hamid, Mallem, Belaiche, Djamel, Dadou, les deux*

*Hamza , Hadj-Hacen, les deux Ahmed, Toufik, Abdelghani, Hichem,*

*Raouf, Hammouche, Nourredine, Ismail, Mustapha, AbdelKader, Belkacem,*

*Sid-Ahmed, et Ilyas.*

*Et à tous ceux qui me connaissent.*

## ملخص:

الهدف من عملنا هذا هو انجاز بطاقة الكترونية تعتمد على المعالج 18F4550 ، و تتواصل مع الحواسيب عن طريق الموصل التسلسلي الشامل. للتحكم في محركات التيار المستمر المستعملة في الروبوتات ، في الآلات ، في العربات الكهربائية... الخ.

يتضمن هذا العمل بعض المفاهيم الضرورية و الأساسية حول بروتوكول الموصل التسلسلي الشامل ، بالإضافة إلى لمحة عن المعالج 18F4550 و الذي صمم لأجل هذا النوع من التطبيقات ، من جانب الحاسوب التطبيق USB مرتبط ببرنامج DELPHI يسمح للمستخدم بالتحكم في محرك التيار المستمر بسهولة . أما من جانب المحرك فالتطبيق USB مرتبط بدارة الاستطاعة، وهي عبارة عن مقطع للتيار يعمل على استقبال التيار المستمر من مأخذ خارجي وتحويله إلى المحرك حسب النسبة المنتقاة من طرف المستخدم أو حسب احتياجات المحرك للوصول إلى السرعة التاموضع أو المطلوبة. كما تم توضيح مختلف مراحل انجاز كل من التطبيق USB ، دارة التشفير ، و دارة الاستطاعة.

إن التجارب التي تم إجرائها على عدة محركات للتيار المستمر قد بينت بشكل قاطع محاسن استعمال الموصل التسلسلي الشامل خاصة المرونة.

الكلمات المفاتيح: الموصل التسلسلي الشامل، المعالج المقطع، ت ع ن ( تغيير عرض النبضة) ، محرك التيار المستمر.

## Résumé:

L'objectif de notre travail consiste à développer une interface USB à base d'un microcontrôleur 18F4550 pour le pilotage des moteurs à courant continu utilisés pour les déplacements des robots, des machines, des véhicules électriques etc.....

Dans ce travail, quelques notions de base sur le protocole USB sont données. Un aperçu sur le PIC 18F4550 qui est dédié aux applications USB est ensuite présenté. Du côté ordinateur l'interface USB est associée à une interface DELPHI qui permet à l'utilisateur de piloter aisément le moteur à courant continu. Du côté moteur l'interface USB est reliée à un étage de puissance. C'est un hacheur dévolteur alimenté par une source continue externe qui commande le moteur selon un rapport cyclique choisi par l'utilisateur ou selon les besoins du moteur dictés par les consignes de vitesse ou de position. Les différentes étapes de réalisation de l'interface USB à base du 18F4550, de l'interface codeur, de l'étage de puissance sont alors détaillées. Les essais sur plusieurs moteurs continus ont été concluants et ont montré les différents avantages et surtout la flexibilité du bus USB.

Mots clés : USB (bus série universel), microcontrôleurs PIC, Hacheur, MLI, Moteur CC.

## Abstract:

The purpose of this work is to develop a USB interface based on the microcontroller 18F4550 for the control of DC motors, which are used in robotics, machines, electric vehicles etc....

In this work, some necessary and basic notions of the USB protocol are given. In addition, the PIC18F4550 which is dedicated for USB applications is introduced.

On the computer side the USB interface is associated to a DELPHI interface allowing the user to control a DC motor easily. On the motor side the USB interface is connected to a power stage, which is a Buck chopper powered by an external continuous power supply. This enables the control of motor speed to be carried out according to the cyclic ratio chosen by the user (manually) or according to motor's needs dictated by speed's or position's instructions (feedback). The different steps of the design of the 18F4550 based USB interface, the coder interface, and the power stage are presented in this thesis. Tests with many DC motors have shown the different advantages, mainly the flexibility of the USB.

Keys words: USB (universal serial bus), PIC microcontroller, chopper, PWM, DC Motor.

## SOMMAIRE :

### Chapitre I

Introduction .....	3
I.1 - Les avantages de l'USB .....	3
I.2 - Les versions et l'utilisation .....	3
I.3 - Les vitesses réelles .....	5
I.4 - Le port USB .....	6
I.5 -Le câble USB .....	6
I.6 - Les connecteurs USB .....	7
I.7- Les hubs .....	8
II - L'architecture de bus .....	9
II.1- La norme On-The-Go .....	10
II.2- Identification de la vitesse .....	10
II.3 - Les types de paquet USB .....	11
II.4 - Les modes de transfert .....	13
II.4.a - Transfert de contrôle .....	13
II.4.b - Transfert isochrone .....	13
II.4.c - Transfert par interruption .....	14
II.4.d - Transfert en bloc(Bulk) .....	14
II.5 - Gestion de la bande passante .....	14
II.6 - Les terminaisons (Endpoints) .....	15
II.7- Les descripteurs USB .....	16
II.7.a -Le Descripteur d'appareil .....	16
II.7.b -Le Descripteur de Configuration .....	16
II.7.c -Le Descripteur d'Interface .....	17
II.7.d- Le descripteur d'Endpoint .....	17
II.8 -Délai de propagation .....	17

### Chapitre II

I- Introduction .....	18
II- Caractéristiques générales du pic 18F4550 .....	18
II.1- USB .....	18

II.2- La Mémoire flash .....	18
II.2.1- Auto-Programmabilité .....	18
II.3- Jeu d'Instruction Prolongée .....	19
II.4- Module CCP amélioré .....	19
II.5- Convertisseur analogique-numérique 10-bit .....	19
Les ports E/S .....	19
III- UNIVERSAL SERIAL BUS (USB) .....	21
III.1- émetteur récepteur (Transceiver): .....	21
III.1 .a- Emetteur récepteur interne (Internal Transceiver) .....	22
III.1 .b- Emetteur récepteur externe (External Transceiver) .....	22
III.2- choix de la vitesse de transfert .....	24
III.2.1- Les Résistances pull-up internes .....	24
III.2.2- Les Résistances pull-up externes .....	24
IV- Les Interruptions du module USB .....	25
V- Aperçu sur l 'USB autour du PIC .....	27
V. 1- Structure en couches (LAYERED FRAMEWORK) .....	27
V.2- Les Trames .....	28
V.3- L'Alimentation Electrique .....	28
V.4- L'énumération du 18F4550 .....	28
VI- Conception et implémentation de l'USB autour du PIC .....	29
VI.1- Vue Générale .....	29
VI.2- CDC .....	30
VI.3- Les fonctions d' UART USB .....	31
VI.4- Caractéristiques de l'émulateur du RS-232 de Microchip .....	31
VI.5 - Le code d'application .....	32
VI.6- Identificateur de constructeur (VID) et Identificateur de produit (PID) .....	34
VI.7- Drivers pour Microsoft Windows 2000 ET Windows XP .....	34
<u>Chapitre 3 :</u>	
I- Introduction : .....	35
II- description du matériel : .....	37
II.1- le hacheur : .....	37
II.2- L'isolation optoélectronique : .....	38
II.3- La chaine d'acquisition de vitesse du moteur : .....	39
II.3.1- L'encodeur optique : .....	40

II.3.2- le compteur quadrature .....	42
Les registres de LS7266R1 : .....	42
III- L'interface utilisateur DELPHI : .....	43
III- 2.Les modes utilisés : .....	45
III- 2.1.Le mode Direct (en boucle ouverte) : .....	45
III- 2.2.Le mode d'asservissement de vitesse : .....	46
III- 2.3.Le mode asservissement de position : .....	47
III- 2.4.Le mode asservissement point à point : .....	48
III- 3.la base de données : .....	48
IV . Implémentation de la commande : .....	50
IV.1- La commande du hacheur : .....	50
IV.1.1 La modulation en largeur d'impulsion (MLI) : .....	50
IV.1.2 - Génération des signaux MLI avec PIC18F4550 : .....	51
IV.1.3- Configurations des sorties MLI du module CCP : .....	52
IV.2. Les routines d'acquisition de la position et la vitesse du moteur : .....	55
IV.3.La boucle de régulateur .....	57
IV.4.Le profil de vitesse : .....	60
IV.5.Protection du matérielle : .....	61
<u>Chapitre 4 :</u>	
I-Introduction : .....	63
I.1-Le mode vitesse : .....	63
Test 1 : régulateur proportionnel .....	63
Test 2 : régulateur proportionnel et intégrateur. ....	64
Test 3 : régulateur proportionnel, intégrateur et dérivateur. ....	65
I.2-Le mode Position : .....	66
Teste 4 : sans avec profil de vitesse. ....	66
Teste 5: avec profil de vitesse. ....	67
Test 6 : avec profil de vitesse et déplacement court. ....	68
Conclusion générale .....	69
Bibliographie .....	71
Annexe .....	74

## Liste des figures

### Chapitre 1 :

Figure I.1	Logos USB 1.1 et 2.0 .....	5
Figure I.2	Composition d'un câble USB full/high speed .....	7
Figure I.3	Principe du Codage NRZI .....	7
Figure I.4	L'organisation interne d'un Hub .....	8
Figure I.5	Topologie du bus USB .....	9
Figure I.6	La structure des paquets USB .....	11
Figure I.7	Paquet début de trame (SOF) .....	12
Figure I.8	Paquet SETUP, IN ou OUT .....	12
Figure I.9	Paquet DATA .....	13
Figure I.10	Paquet HANDSHAKE .....	13
Figure I.11	L'architecture des liaisons USB .....	15
Figure I.12	Temps de propagation d'une tram USB .....	17

### Chapitre 2 :

Figure II.1	Le PIC 18F4550 .....	21
Figure II.2	Une vue générale du module USB du 18F4550 .....	22
Figure II.3	Emetteur Récepteur Externe Typique Avec Isolation .....	23
Figure II.4	Les circuits externes pour full ou low speed .....	25
Figure II.5	Logique d'interruption d'USB .....	26
Figure II.6	Exemple d'une transaction d'USB et des événements d'interruption....	26
Figure II.7	Les Couches D'USB .....	27
Figure II.8	Emulation de communication RS-232 via le bus USB .....	30
Figure II.9	La forme de projet sous MPLAB .....	32

### Chapitre 3 :

Figure III.1	Représentation fonctionnelle du système globale .....	36
Figure III.2	Le microcontrôleur 18F4550 et la liaison USB .....	36
Figure III.3	L'étage de puissance .....	37



Figure III.4	L'étage de protection .....	39
Figure III.5	L'acquisition de la vitesse.....	40
Figure III.6	principe de fonctionnement d'un encodeur optique.....	40
Figure III.7	Les signaux visualisés par l'interface Delphi générés à partir d'un encodeur 500 impulsions / tour. ....	41
Figure III.8	les modes d'adressage des registres de LS7266 .....	43
Figure III.9	La fenêtre principale de l'interface utilisateur.....	44
Figure III.10	Le mode Direct .....	46
Figure III.11	Le mode Vitesse.....	46
Figure III.12	Le mode Position.....	47
Figure III.13	Le mode asservissement point à point.....	48
Figure III.14	La base de données visualisée par Delphi.....	49
Figure III.15	La base de données visualisée par Microsoft Excel .....	50
Figure III.16	Exemple d'un signal PWM avec un rapport cyclique de 10%.....	51
Figure III.17	Schéma bloc simplifié du module «MLI » .....	52
Figure III.18.a	Le Full-bridge FORWARD.....	53
Figure III.18.b	Le Full-bridge REVERSE.....	54
Figure III.19	Le changement de direction de mode Full-Bridge.....	54
Figure III.20	Organigramme de l'acquisition de vitesse et la tâche USB.....	56
Figure III.21	les composants d'un PID.....	58
Figure III.22	l'organigramme de La boucle de PID.....	59
Figure III.23	graphe de variation de la vitesse.....	60
Figure III.24	L'organigramme de profil de vitesse.....	62
<u>Chapitre 4 :</u>		
Figure IV.1 :	asservissement de vitesse avec régulateur proportionnel.....	63
Figure IV.2 :	asservissement de vitesse avec régulateur proportionnel et intégrateur .....	64
Figure IV.3 :	asservissement de vitesse avec régulateur PID.....	65
Figure IV.4 :	asservissement de position sans profil de vitesse. ....	66
Figure IV.5 :	asservissement de position avec profil de vitesse.....	67
Figure IV.6 :	asservissement de position sans profil de vitesse et déplacement court .....	68

## **Liste des tableaux**

### Chapitre 1 :

Tableau I.1	Comparaison de l'USB avec des autres bus d'ordinateur	4
Tableau I.2	paramètres électriques des câbles USB	6
Tableau I.3	Les types des connecteurs USB	8
Tableau I.4	Les différents PID des paquets	12

### Chapitre 2 :

Tableau II.1	Caractéristique de 18F4550	20
Tableau II.2	Les combinaisons des états de ces signaux et de leur traduction	24
Tableau II.3	La version 1,0 des firmwares de l'émulateur du RS-232 de Microchip	31

RÉALISATION D'UNE  
INTERFACE USB  
AUTOUR D'UN  
MICROCONTRÔLEUR  
(PIC 18F4550)  
POUR LA  
COMMANDE D'UN  
MOTEUR À COURANT  
CONTINU

## **Introduction générale**

L'informatique est un domaine fraîchement développé, même s'il trouve ses origines dans l'antiquité avec la cryptographie ou dans la machine à calculer de Blaise Pascal, au XVIIe siècle. Ce n'est qu'à la fin de la seconde guerre mondiale qu'elle a été reconnue comme une discipline à part entière et a développé des méthodes, puis une méthodologie qui lui étaient propres. L'ère des ordinateurs modernes commença avec les développements de l'électronique pendant la seconde guerre mondiale, ouvrant la porte à la réalisation concrète de machines opérationnelles. Les systèmes à base d'ordinateurs sont aujourd'hui utilisés par non seulement les électroniciens et informaticiens mais par tout le monde et dans tous les domaines. Ceci a été rendu possible par l'utilisation d'interfaçages, hardware ou software, et les différents bus et connecteurs.

Généralement, quand on parle de périphérique permettant à un ordinateur de communiquer avec le monde extérieur, on pense tout de suite à un clavier, un écran, une souris, une imprimante, etc.... En réalité, dans la majeure partie des applications où il est nécessaire de contrôler des systèmes électroniques, de tels organes ne sont pas évidemment suffisants, mais il faut utiliser des cartes spécifiques constituant des interfaces vers l'extérieur.

Ces interfaces doivent relier les circuits électroniques à l'ordinateur et permettre l'échange des données entre les deux entités. Dans la plupart des cas, les cartes sont constituées d'un microcalculateur (microcontrôleur, FPGA,...) avec certain nombre de ressources entrées/soties numériques et analogiques, ces ressources codées et transmis à un logiciel (interface utilisateur) présent dans l'ordinateur. C'est dans cet objectif qu'elle nous a été confiée la réalisation d'une carte à base d'un microcontrôleur PIC18F4550 pour le pilotage d'un moteur à courant continu via le bus USB.

Le bus USB a pris son envol grand public avec l'apparition de Windows98. Les nouveaux périphériques qui apparaissent sur le marché actuel sont le plus souvent de type USB et les ports séries et parallèles commencent à désertir progressivement les cartes mères.

Pour pouvoir continuer à développer des applications reliées aux ordinateurs, il est préférable de maîtriser cette nouvelle technologie USB.

L'objectif de ce travail consiste à développer une carte à base d'un microcontrôleur PIC18F4550 pour le pilotage d'un moteur à courant continu via le port USB. La carte microcontrôleur recevra depuis l'ordinateur et via la liaison USB les informations de vitesse consigne, position consigne, marche/ arrêt, le sens de rotation etc.... Un étage de puissance prendra en charge l'interface avec le moteur à courant continu. L'ordinateur recevra depuis la carte microcontrôleur la vitesse réelle du moteur, la position, l'état de fonctionnement du moteur (marche/arrêt), le sens actuel de rotation et le rapport cyclique appliqué.

Durant ce projet, on arrive à :

1. Réaliser une carte prototype de commande d'un moteur DC. Cette carte a les fonctions de base suivante,
  - Reçoit les données de consigne de position.
  - A une interface codeur pour l'acquisition de la vitesse du moteur.
  - Pilote un moteur à courant continu.
1. En second, réaliser les fonctions logiques suivantes :
  - Interface USB.
  - Gestion du protocole de communication.
  - Boucle d'asservissement.
  - Sécurité d'asservissement.
  - Se connecter à un étage de puissance.
2. Création d'une interface DELPHI qui permet à l'utilisateur de gérer les différents scénarios de test de la carte avec ses éventuels modes de fonctionnement.
  - Commande en boucle ouverte.
  - Commande point à point avec profil de vitesse.
  - Commande sans profil (asservissement au point de consigne).
  - Pilotage E/S.
3. Et comme notre carte est destinée aux travaux pratiques, on a ajouté des fonctions supplémentaires (base de données, graphes,...) qui aident pour l'analyse des différentes expériences.

---

# CHAPITRE I

## GÉNÉRALITÉS

### SUR

## LA NORME USB

---

## **Chapitre I :**

# **Généralités Sur La norme USB**

### **Introduction :**

L'USB (Universal Serial Bus) a été conçu afin de remplacer les nombreux ports externes d'ordinateur lents et incompatibles, notamment les ports série et parallèle. L'USB est aujourd'hui présent sur tous les ordinateurs et est généralement utilisé pour brancher les imprimantes, les scanners, les modems et de nombreux appareils de stockage des données, dont les clés USB.

Le bus USB a été conçu en 1994 par le consortium de sept partenaires industriels qui sont : Compaq, DEC, IBM, Intel, Microsoft, NEC et Northern Telecom. C'est eux qui ont commencé à concevoir la norme USB.

L'architecture qui a été retenue pour ce type de port est en série pour deux raisons principales :

- l'architecture série permet d'utiliser une cadence d'horloge beaucoup plus élevée qu'une interface parallèle, car celle-ci ne supporte pas des fréquences trop élevées (dans une architecture à haut débit, les bits circulant sur chaque fil arrivent avec des décalages, provoquant des erreurs) ;
- les câbles série coûtent beaucoup moins cher que les câbles parallèles.

### **I.1 - Les avantages de l'USB :**

Les avantages de l'USB sont nombreux :

- vitesse beaucoup plus élevée que les ports série classiques ou parallèle.
- un périphérique USB peut se connecter sur n'importe quel port USB.
- les périphériques peuvent être alimentés via le port USB même (500mA/5 volts maximum).
- faible coût de l'interface.
- on peut connecter jusqu'à 127 périphériques sur un PC, soit directement, soit via des hubs.
- L'USB support le Hot Plug & Play : on peut brancher et débrancher les périphériques quand on veut, l'ordinateur charge automatiquement les pilotes sans avoir besoin de redémarrer l'ordinateur.
- support des fonctions de mise en veille par l'ordinateur.
- fiabilité et sécurité (détection et correction d'erreurs), trois vitesses possibles et quatre modes de transferts [S4].

### **I.2 - Les versions et l'utilisation :**

La version 1.1 du l'USB permet de communiquer dans deux modes : lent (1,5 Mbit/s) ou rapide (12 Mbit/s).

- Le mode lent ("low speed") permet de connecter des périphériques qui ont besoin de transférer peu de données, comme les claviers, les souris et les manettes de jeux.
- Le mode rapide ("full speed") est utilisé pour connecter des imprimantes, scanners, disques durs, graveurs de CD et autres périphériques ayant besoin de plus de rapidité.

**Tableau I.1:** Comparaison de l'USB avec des autres bus d'ordinateur.

Interface	Nombre des dispositifs (maximum)	La longueur maximale du câble (mètre)	La vitesse maximum (bits/sec.)
USB	127	5-30	1,5-12-480M
RS-232	2	15-30	20K
RS-485	32	120	10M
IrDA	2	2	115K
Microwire	8	3	2M
SPI	8	3	2,1M
I2C	40	5,5	3,4M
IEEE- 1394 (Fire Wire)	64	4,6	400M (jusqu' à 3.2G avec IEEE-1394b)
IEEE-488 (GPIB)	15	18,28	8M
Ethernet	1024	490	10M/100M/1G
MIDI	2	15,2	31,5k
P/Parallèle	2	3-9	8M
BUS CAN	2 à 20	40 à 1000	125 K
	2 à 30		1 M

La nouvelle version de ce bus, USB 2.0, comporte un troisième mode ("high speed") permettant de communiquer à 480 Mbit/s. Il est utilisé par les périphériques rapides : disques durs, graveurs, vidéo etc.



La compatibilité entre les périphériques USB 1.1 et 2.0 est assurée. Toutefois l'utilisation d'un périphérique USB 2.0 sur un port USB 1.1, limitera le débit maximum à 12 Mbit/s. De plus, le système d'exploitation est susceptible d'afficher un message expliquant que le débit sera bridé.

L'USB est un bus rapide, mais il n'est pas conçu pour faire tout. L'USB high speed est compétitif avec les interfaces IEEE-1394 (Firewire) 400 Mbits/s, mais IEEE-1394b est devenu plus rapide, à 3,2 Gbits/s.

L'USB a été conçu essentiellement comme un bus de bureau, la longueur du câble USB est de 5 mètres. D'autres interfaces, telles que RS-232, RS-485, et Ethernet, permettent des câbles beaucoup plus longs. Nous pouvons allonger la longueur d'un lien USB jusqu'à 30 mètres en utilisant 6 segments de câble de 5 mètres qui connectent cinq hubs et un périphérique [7].

Les périphériques certifiés USB portent le logo suivant :



Figure I.1: Logos USB 1.1 et 2.0

### I.3 - Les vitesses réelles :

Les vitesses (1,5 Mbit/s - 12Mbit/s - 480 Mbit/s) sont des vitesses approximatives. Ce sont en fait les vitesses que peuvent supporter les différents bus USB. Le taux de transfert de données réel est plus faible.

En fait le bus doit faire passer, outre les données, les bits de status, de contrôles, et les bits d'erreurs. Sans oublier que plusieurs périphériques peuvent se partager le bus.

Pour la vitesse Low Speed le débit réel est de 800oct/s en mode Interrupt. En effet la norme permet une interruption toute les 10ms, dans les meilleurs des cas, car elle est réglable de 10 à 255ms. Donc avec 100 interruptions/s et une taille du paquet de 8 octets pour cette vitesse, le débit réel sera de 800octets/s.

Pour la vitesse Full Speed, en mode Interrupt la norme permet une interruption au mieux toute les 1ms (réglable de 1 à 255ms), c'est à dire 1000 interruptions/s, la trame pouvant s'étendre jusqu'à 64 octets, on obtient donc un débit de 64Ko/s. En mode Bulk on arrive à un débit réel d'environ 1Mo/s. La bande passante est divisée dans le cas ou il y a plusieurs périphériques qui travaillent simultanément.

Pour la vitesse High Speed le taux de transfert réel est de 53Mbit/s.

Il faut bien préciser que les caractéristiques énoncées ci-dessus sont les caractéristiques de l'USB même. Si l'application développée fonctionne avec l'OS Windows il faut encore rajouter d'autres contraintes comme par exemple l'intervalle entre les différentes interruptions qui est de l'ordre de 32ms au minimum, donc le débit en mode interrupt du Full Speed va être encore inférieur [6].

## I.4 - Le port USB :

Les ports d'USB diffèrent beaucoup d'autres ports parce que tous les ports sur le bus partagent une voie d'accès simple à l'ordinateur. Par exemple avec l'interface série RS232, chaque port est indépendant des autres. Si nous avons deux ports RS232, chaque port a son propre chemin de données, et chaque câble ne porte que ses propres données. Les deux ports peuvent envoyer et recevoir des données en même temps.

L'USB utilise une approche différente. Chaque contrôleur d'hôte supporte un bus unique, ou un chemin de données. Chaque connecteur sur le bus représente un port USB, mais à la différence de RS232, tous les périphériques partagent le bus. Ainsi quoiqu'il y ait des ports multiples, chacun avec son propre connecteur et câble, il y a seulement un chemin de données.

Seulement le contrôleur d'hôte ou un seul périphérique, peut transmettre à la fois. Un ordinateur peut supporter plusieurs contrôleurs d'hôte d'USB [1].

## I.5 -Le câble USB

Le câble USB possède deux fils d'alimentation (5V et GND) et deux fils destinés au transfert de données (D+ et D-).

On utilise des couleurs standards pour les fils intérieurs des câbles USB. La normalisation précise les différents paramètres électriques pour les câbles.

**Tableau I.2 : paramètres électriques des câbles USB**

Couleurs des câbles	Fonction	Diamètre Nominal de fil
Rouge	V <sub>Bus</sub> (5 volts)	0.381 à 0.890 mm
Blanc	D-	0.381 mm
Vert	D+	0.381 mm
Noir	Masse	0.381 à 0.890 mm

La longueur maximale autorisée par la norme est de 3m pour un périphérique Low speed (un câble non blindé) et de 5m pour un câble blindé dans le cas d'un périphérique Full ou high speed.

Pour la version Low Speed le blindage n'est pas obligatoire (ce qui assure une plus grande souplesse de manipulation en particulier pour une liaison souris).

Les deux fils (D+ et D-) utilisent le principe de la transmission différentielle afin de garantir une certaine immunité aux bruits parasites de l'environnement physique du périphérique ou de son câble [7].

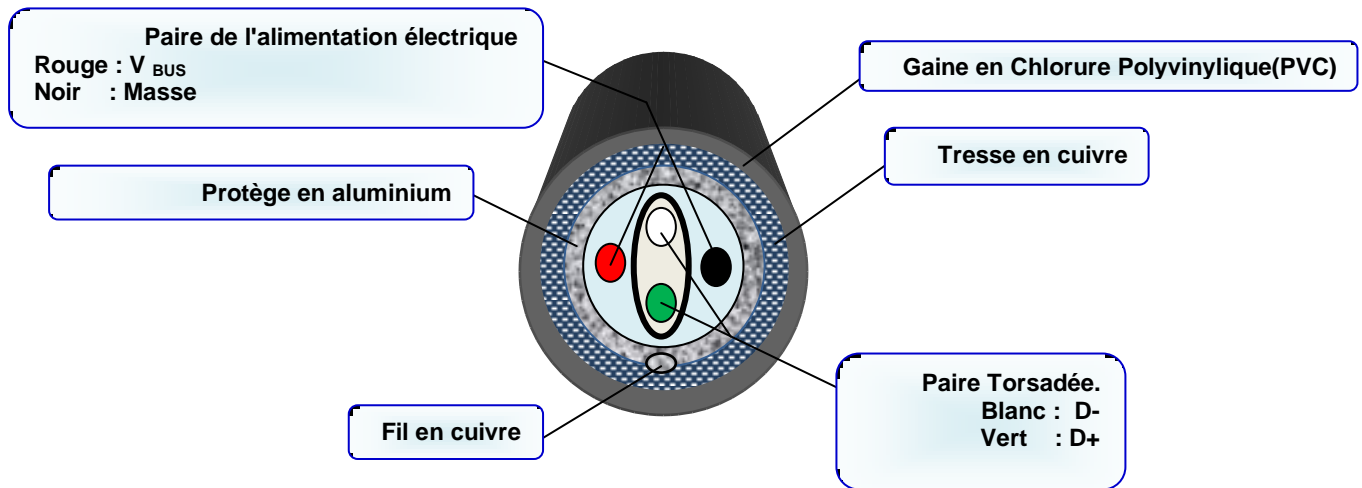


Figure I.2: Composition d'un câble USB full/high speed

La signalisation différentielle symétrique est une méthode de transmission de signaux sur une paire torsadée. Elle consiste à envoyer sur un fil le signal et sur l'autre le signal inverse. On reconstitue le signal à l'arrivée en effectuant la différence des signaux, même s'il y a une perturbation électromagnétique qui dégrade le signal, la différence est inchangée.

Le codage des données se fait selon la méthode NRZI (Non Retour à Zéro Inversé).

Un "1" est représenté par l'absence de changement d'état du bus, un "0" est représenté par un changement d'état du bus.

Ce type de codage est uniquement utilisé pour le transport à travers le cordon USB. Il doit être décodé lors de la réception pour pouvoir retraiter les données.

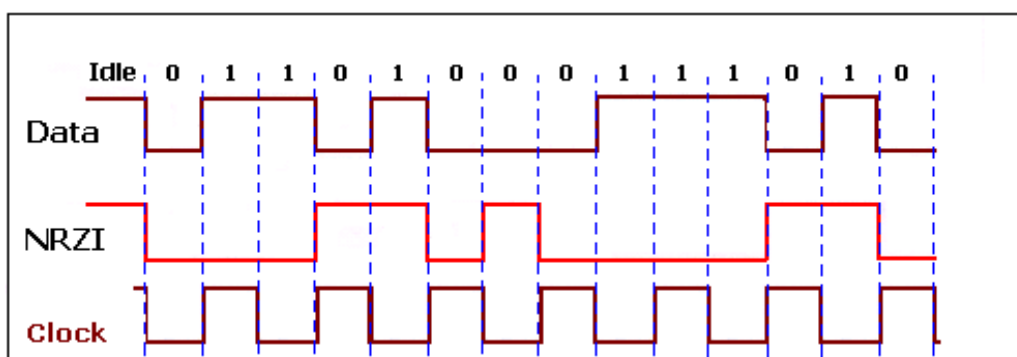








Figure : I.3 Principe du Codage NRZI

## I.6 - Les connecteurs USB :

Le câble USB est composé de deux fiches bien différentes :

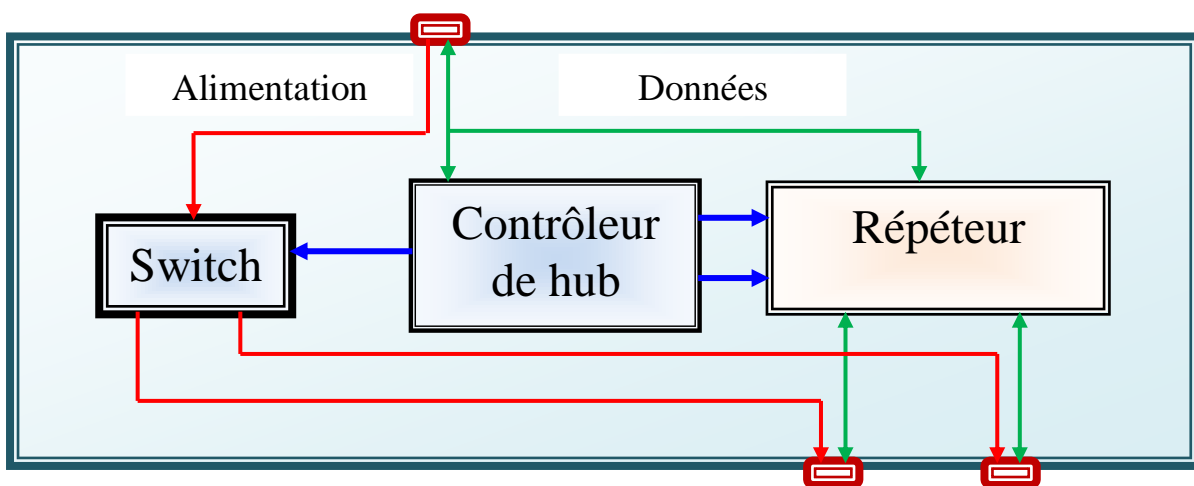
- Une fiche en amont appelée connecteur USB de type A, (branché à l'ordinateur).
- Une fiche en aval qui peut se retrouver en deux versions : Connecteur de type B et un mini connecteur de type B. Ce dernier est réservé aux dispositifs de très faibles dimensions (ou de grande intégration) tels les appareils photo numériques.

**Tableau I.3:** Les types des connecteurs USB

Les connecteurs type A	Les connecteurs type B	Les mini connecteurs type B
		
		

### I.7- Les hubs

Généralement, un ordinateur possède de 2 à 4 ports USB. Mais, avec une imprimante USB, une webcam USB, une souris USB, un scanner USB, ... on arrive vite manquer de ports. La solution est l'ajout d'un hub USB. Ce hub USB permet d'accroître le nombre de ports USB de notre ordinateur. Le hub va permettre de connecter au-moins 4 périphériques. Et on peut connecter un hub à un autre hub, etc.



**Figure I.4:** L'organisation interne d'un Hub

Un hub va jouer le rôle de multiplexeur (connexion de plusieurs périphériques à un même

câble) mais aussi de répéteur, d'amplificateur, de contrôleur du signal et de fournisseur de courant.

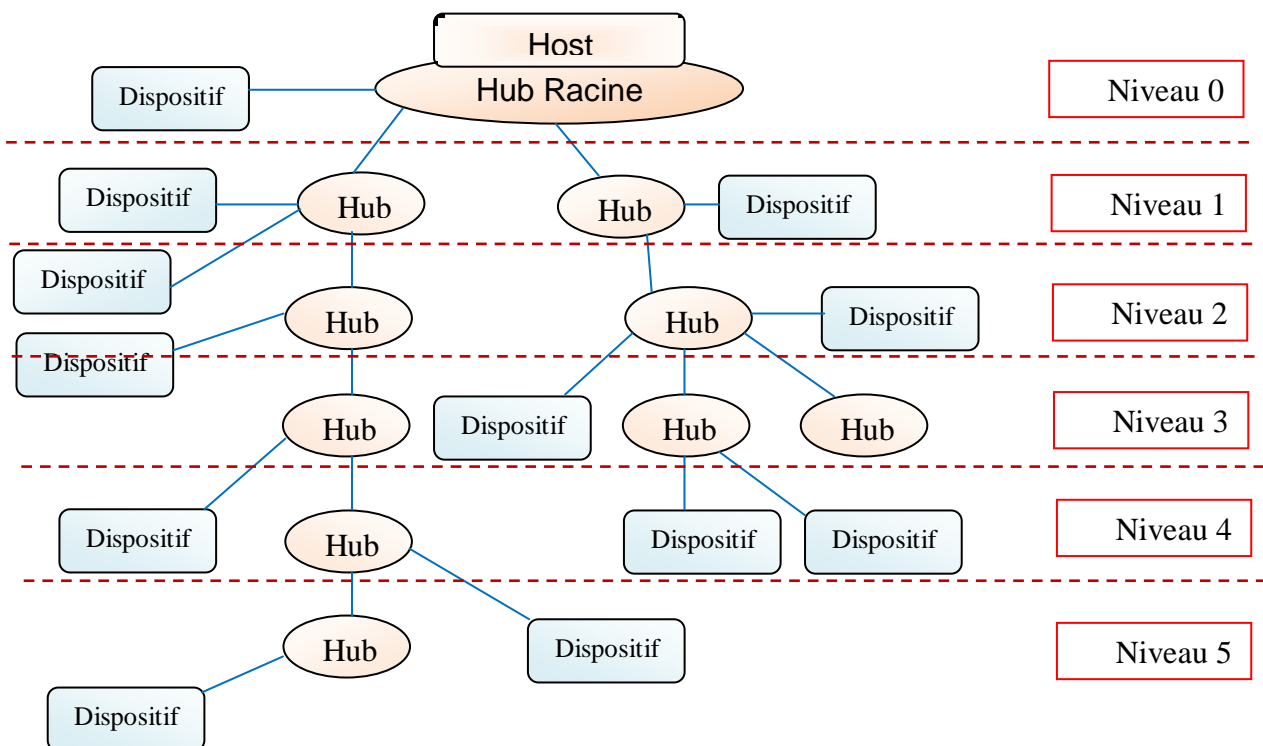
Comme les périphériques peuvent prendre l'alimentation dans le port USB, le hub se doit de pouvoir jouer le même rôle. Bien sûr, les périphériques gros consommateurs d'énergie ont leur propre alimentation électrique. Mais les claviers, souris, webcam, etc. prennent généralement leur électricité dans le port.

## II - L'architecture de bus :

Le bus est entièrement contrôlé par un hôte unique, généralement le PC.

La norme USB permet le chaînage des périphériques, en utilisant une topologie en bus ou en étoile. Les périphériques peuvent alors être soit connectés les uns à la suite des autres, soit ramifiés. La ramification se fait à l'aide des hubs.

La norme permet plusieurs niveaux de hubs sur lesquels se connectent les appareils au nombre maximum de 127. Le hub racine est implémenté dans l'hôte, il peut avoir plusieurs ports. Il peut y avoir en cascade jusqu'à 5 hubs en plus du hub racine. Chaque hub contient un contrôleur qui surveille les différents ports et rends des comptes à l'hôte. Celui-ci interroge les contrôleurs de hubs afin de connaître les connexions et déconnexions d'appareils.



**Figure I.5:** Topologie du bus USB

A la connexion d'un appareil, l'hôte l'identifie et lui assigne une adresse unique : c'est la phase d'énumération. Lors de la connexion d'un hub, il y a énumération de tous les

appareils en aval. L'hôte attribue également la bande passante en fonction des types de transfert requis par les appareils.

Puisque l'adresse est codée sur 7 bits, 128 périphériques ( $2^7$ ) peuvent être connectés simultanément à un port de ce type. Il convient en réalité de ramener ce chiffre à 127 car l'adresse 0 est une adresse réservée.

L'hôte (l'ordinateur) émet un signal de début de séquence chaque une milliseconde (ms), c'est l'intervalle de temps pendant lequel il va donner simultanément la « parole » à chaque périphérique.

Lorsque l'hôte désire communiquer avec un périphérique, il émet un jeton (un paquet de données, contenant l'adresse du périphérique, codé sur 7 bits) désignant un périphérique, c'est donc l'hôte qui décide du « dialogue » avec les périphériques. Si le périphérique reconnaît son adresse dans le jeton, il envoie un paquet de données (de 8 à 255 octets) en réponse, sinon il fait suivre le paquet aux autres périphériques connectés [4].

## **II.1- La norme On-The-Go:**

La norme USB 2.0 s'est enrichie d'une fonctionnalité appelée On-The-Go (OTG) pour pouvoir effectuer des échanges de données point à point entre deux périphériques sans avoir à passer par un hôte (généralement un ordinateur personnel).

Un périphérique OTG peut se connecter à un autre périphérique OTG, à un périphérique (non OTG) ou à un hôte.

Dans le cas d'une connexion OTG-OTG, c'est la position du connecteur du câble sur la prise mini A ou mini B, à chaque extrémité, qui va permettre de déclarer lequel des deux périphériques OTG va être l'hôte. Car les câbles USB sont directionnels. Nous ne pouvons les brancher que dans un sens, A sur l'hôte, B sur l'invité. Grâce au réceptacle Mini AB, l'unité DRD (Dual-Role Device) sait quel rôle elle doit jouer : hôte, si elle est raccordée en A, invité si c'est en B. Ensuite, il peut se produire un renversement des rôles suite à une étape de négociation entre les deux systèmes OTG, un sous-protocole nommé HNP (Host Negotiation Protocol) se charge d'établir un dialogue à l'issue duquel l'une devient hôte et l'autre invitée.

Les applications de cette technologie sont par exemple la connexion directe d'un appareil photo avec une imprimante, la connexion d'un téléphone mobile avec un lecteur MP3, etc.

## **II.2- Identification de la vitesse :**

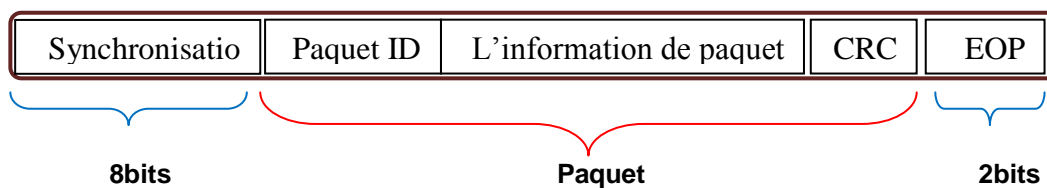
Un appareil USB doit indiquer sa vitesse en mettant soit D+ ou D- à 3,3V. Un appareil full speed, utilisera une résistance de rappel rattachée à D+ et un appareil low speed, utilisera une résistance de rappel rattaché à D-. Ces résistances de rappel à l'extrémité de l'appareil seraient aussi utilisées par l'hôte ou Hub pour détecter la présence d'un appareil connecté à son port. Sans résistance de rappel, l'USB suppose qu'il n'y a rien de connecté au BUS. Certains appareils possèdent cette résistance intégrée sur le silicium, pouvant être connecté ou non sous commande micro programmée, d'autres exigent une résistance externe.

Les appareils high speed démarreront dès qu'ils seront connectés en tant qu'appareils Full speed (1,5 kOhms à 3,3V). Une fois qu'il sera attaché il émettra un paquet à haute vitesse pendant la réinitialisation et établira une connexion high speed si le Hub le supporte [31].

### II.3 - Les types de paquet USB :

Toutes les communications sur le bus sont transférées dans un intervalle de temps régulier de 1ms appelé trame de bits (ou « Frame »). Chaque trame contient des transactions engendrées par le driver USB de façon à exécuter plusieurs tâches. Chaque transaction est composée d'une séquence de paquets USB.

La structure des paquets étant identique, il faut préciser que le format est différent selon la nature du paquet.



**Figure I.6:** La structure des paquets USB

**Sync :** Tous les paquets doivent commencer avec un champ Sync. Le champ Sync fait de 8 bits de long pour low et full speed ou 32 bits pour high speed, est utilisé pour synchroniser l'horloge du récepteur avec celle de l'émetteur / récepteur.

**PID :** signifie Paquet ID. Ce champ est utilisé pour identifier le type de paquet qui est envoyé.

**CRC :** Les Contrôles à Redondance Cyclique sont exécutés sur les données à l'intérieur du paquet de charge utile.

Tous les paquets Tokens ont un CRC de 5 bits tandis que les paquets de données ont un CRC de 16 bits.

**EOP :** Fin de Paquet. Signalé par une sortie unique zéro (SE0) pendant une durée approximative de 2 bits suivie par un état " J " d'une durée de 1 bit.

Il existe 4 catégories de paquets que l'on peut classifier selon leur utilisation dans TOKEN, DATA, HANDSHAKE et SPECIAL. Tous les paquets sont diffusés vers tous les Dispositifs connectés au bus, mais seulement le dispositif concerné répondra au contrôleur Hôte.

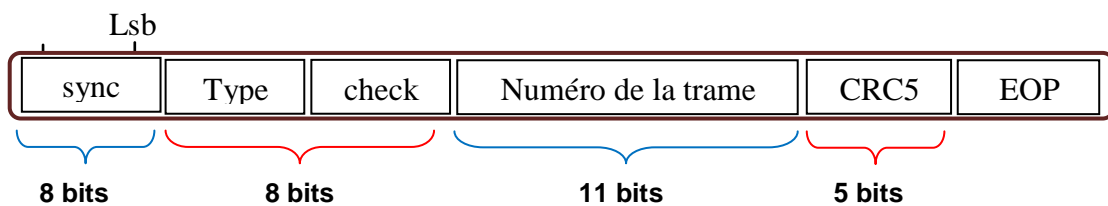
- 1 - Les paquets de type TOKEN (jeton) identifient la transaction qui vient d'être initialisée.
- 2 - Les paquets de type DATA contiennent la charge utile.
  - La taille maximale de données (données utiles) pour les périphériques Low Speed est de 8 octets.
  - La taille maximale de données (données utiles) pour les périphériques Full Speed est de 64 octets.

- La taille maximale de données (données utiles) pour les périphériques High Speed est de 1024 octets.
- 3 - Les paquets HANDSHAKE sont utilisés pour confirmer ou infirmer le bon déroulement du transfert et pour signaler la présence d'erreurs.
- 4 - Le seul paquet SPECIAL utilisé est le paquet de préambule (preamble) qui sert à notifier tous les hubs qu'il y aura une transaction lente qui va se dérouler.

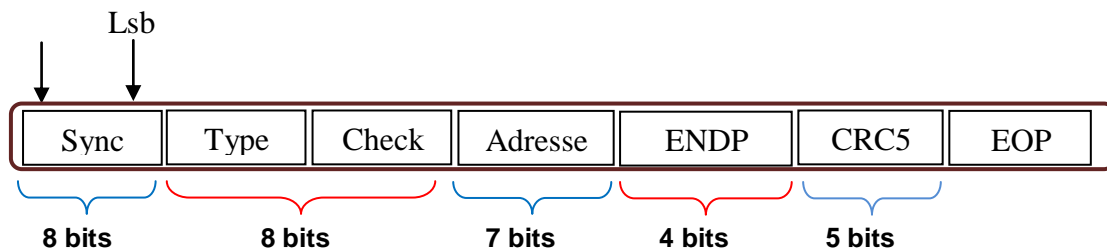
Le tableau suivant montre les codes binaires (PID) pour les quatre types de paquets.

**Tableau I.4:** Les différents PID des paquets

Catégorie de paquet	Type de paquet	Code Binaire(PID)
TOKEN	Start of Frame - SOF -	0101
	Setup Paquet - SETUP -	1101
	Input Packet - IN -	0001
	Output Packet - OUT -	1001
DATA	Data0	0011
	Data1	1011
HANDSHAKE	Acknowledge - ACK -	0010
	Negative Ack - NAK -	1010
	EndPoint Stall - STALL -	1110
	No Response Yet - NYET -	0110
SPECIAL	Preamble - PRE -	1100



**Figure I.7:** Paquet début de trame (SOF)



**Figure I.8:** Paquet SETUP, IN ou OUT



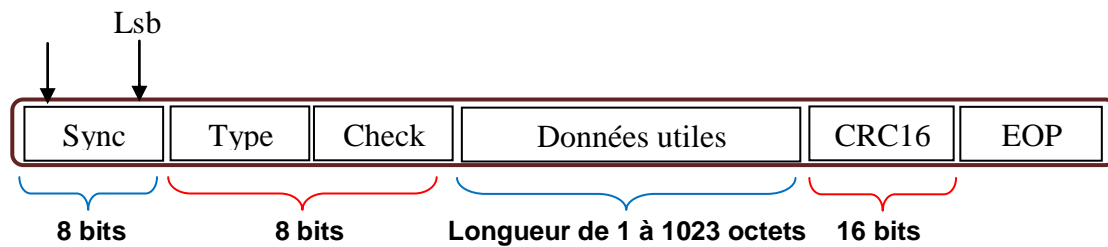


Figure I.9: Paquet DATA

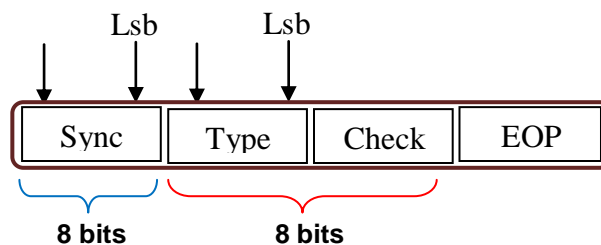


Figure I.10: Paquet HANDSHAKE

## II.4 - Les modes de transfert :

Les différents paquets sont organisés dans un certain ordre ; c'est ce qu'on appelle le type de transfert.

Il existe 4 modes de transfert en USB, chaque mode de transfert fourni au concepteur des compromis dans les domaines de la détection d'erreur et de la reprise, du temps d'attente garanti et de la bande passante [27].

### II.4.a - Transfert de contrôle :

Destiné à la configuration et à la commande d'un appareil. Il se compose d'une transaction de "Setup", des transactions de données éventuelles dans le sens indiqué par le Setup et d'une transaction de statut. Le système USB utilise le canal par défaut pour configurer l'appareil par des transferts de contrôle. Ces transactions ont un format défini par la norme. La charge utile des paquets de données est de 8, 16, 32 ou 64 octets en full speed et de 8 octets en low speed. La taille maximale est définie par le nombre d'Endpoint.

Un système de retransmission en cas d'erreur assure la fiabilité du transfert.

### II.4.b - Transfert isochrone:

C'est un transfert à débit constant tolérant aux erreurs destinées aux flux importants de données tels l'audio ou la vidéo. Le débit dans le canal est garanti, aucun format n'est imposé aux données. La charge utile des paquets peut atteindre 1024 en high speed, Ce mode ne supporte pas le low speed. Les erreurs sont signalées mais il n'y a pas de retransmission. La synchronisation peut se faire, entre autre, en détectant le paquet "SOF" qui est placé au début de chaque trame.

Les transferts Isochrones fournissent.

- Un accès garanti à la bande passante USB.
- Un temps d'attente limité.
- Des flux de données unidirectionnels.
- La détection d'erreur via le CRC, mais sans reprise ou garantie de livraison.
- Seulement pour les modes full et high speed.

#### **II.4.c - Transfert par interruption :**

Il est destiné aux appareils transmettant peu de données mais dans un délai garanti. Il ne s'agit pas vraiment d'interruptions mais d'interrogations de l'appareil par l'hôte à une cadence fixe et garantie. Aucun format n'est imposé aux données. La charge utile des paquets de données est identique à celle des transferts de contrôle. En cas d'erreur il y a retransmission lors de l'interrogation suivante.

Les transferts par interruption fournissent.

- Temps de retard (ou Latence) garanti.
- Ligne de flux – Unidirectionnel.
- Détection d'erreur et nouvel essai sur période suivante.

#### **II.4.d - Transfert en bloc(Bulk) :**

Il est destiné aux gros volumes de données sans contraintes temporelles, par exemple pour une imprimante. Le système peut retarder un transfert en bloc jusqu'à disposer de suffisamment de bande passante. Aucun format n'est imposé aux données. La charge utile des paquets de données est de 8, 16, 32 ou 64 octets. Les transferts en bloc supportent la correction des erreurs ; ils sont full speed ou en high speed.

Les transferts en Bloc fournissent.

- Détection d'erreurs via le CRC, avec la garantie de livraison.
- Utilisés pour de grandes quantités de données.
- Pas de garantie de bande passante ou du temps d'attente minimum.
- Des flux de données unidirectionnels.
- Seulement pour les modes full et high speed.

#### **II.5 - Gestion de la bande passante :**

L'hôte est responsable de la bande passante du Bus.

La spécification place des limites sur le Bus, l'autorisant à allouer plus de 90% pour des transferts périodiques (Interruption et Isochrone) sur un Bus Full speed. Sur des Bus high speed, cette limitation se réduit à moins de 80% d'une micro-trame qui peut être allouée pour des transferts périodiques.

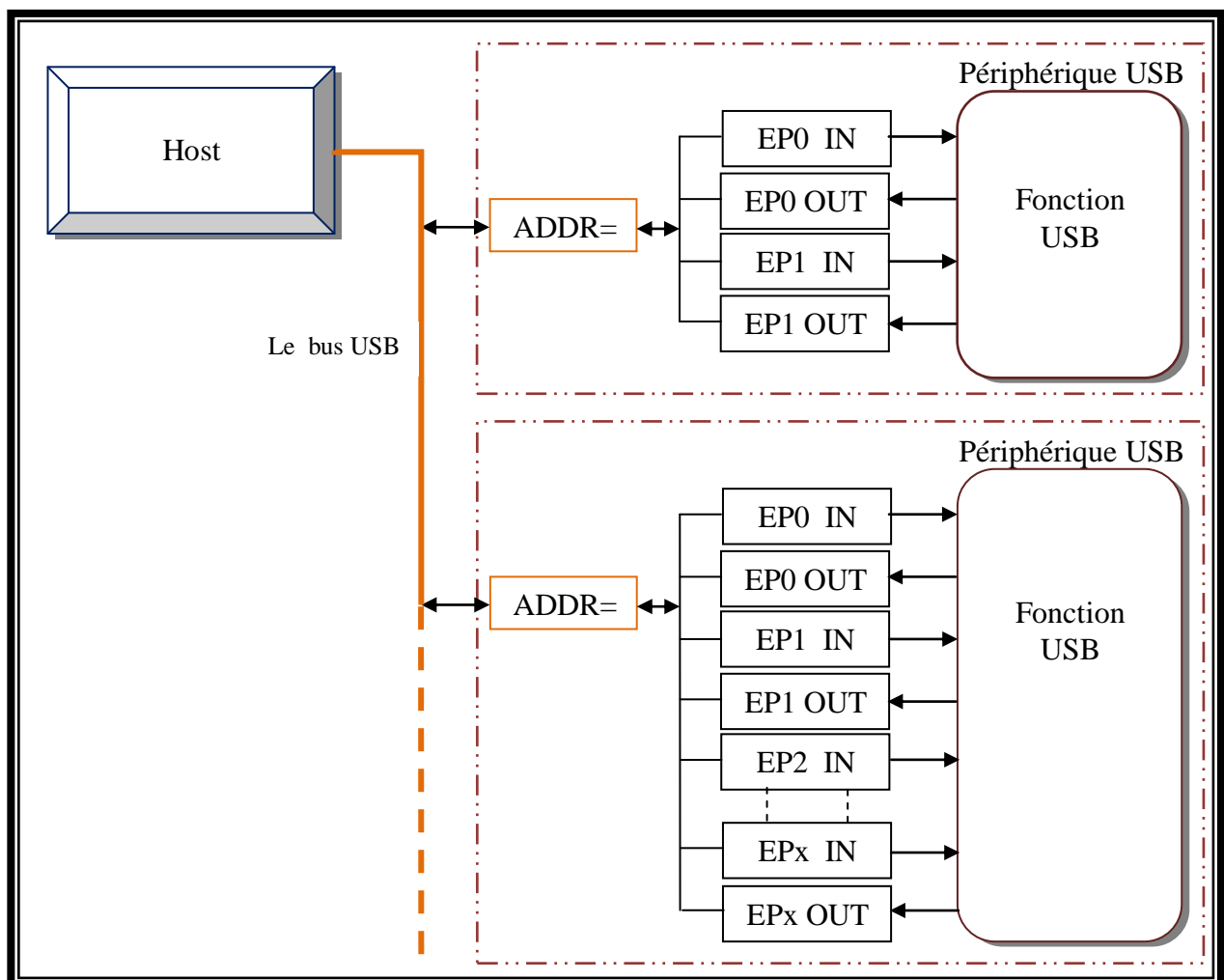
Aussi vous pouvez assez rapidement voir que si vous avez un Bus hautement saturé avec des Transferts périodiques, les 10% restants sont laissés pour les transferts de contrôle et une fois qu'ils ont été attribués, les transferts en Bloc prendront ce qui reste.

## II.6 - Les terminaisons (Endpoints) :

Comme l'indique la figure I.11, chaque périphérique USB est décomposé en plusieurs sous blocs, possédant chacun un rôle différent dans la communication. Il s'agit ici de décrire de manière générale cette architecture, et non de développer complètement le fonctionnement précis de chaque étage.

Nous pouvons distinguer 3 sous blocs principaux :

- La partie qui décode l'adresse émise par l'hôte dans le paquet Token. Cette entité permet au périphérique de savoir que c'est bien à lui que l'hôte s'adresse.
- La partie Endpoint.
- La partie réalisant la fonction USB proprement dit.



**Figure I.11 :** L'architecture des liaisons USB.

Les Endpoints peuvent être vues comme des intermédiaires, des tampons entre le bus et la fonction USB. En effet, il n'est pas possible pour le bus d'écrire directement dans la fonction, comme il n'est pas possible pour la fonction d'écrire directement sur le bus. Les données sont

donc stockées temporairement (jusqu'à ce que l'hôte ou le périphérique les lisent) dans les Endpoints. C'est donc pour cette raison que dans le paquet Token, l'hôte précise l'endpoint à laquelle il veut s'adresser.

On peut remarquer qu'une même fonction USB peut utiliser plusieurs Endpoints. Dans la spécification USB 1.1, le nombre de paires d'Endpoints est limité à 2, c'est-à-dire que les communications peuvent se faire via EP0 In, EP0 Out, EP1 In et EP1 Out. La paire de terminaisons utilisées par défaut par l'hôte pour dialoguer avec le périphérique est EP0.

Un dispositif USB 2.0 peut supporter jusqu'à 15 Endpoints.

## **II.7- Les descripteurs USB :**

On peut définir les descripteurs comme étant des blocs d'informations pré formatés. Tous les composants USB doivent obligatoirement posséder les descripteurs standards.

Tous les transferts d'informations durant la phase d'énumération se font suivant le type Control. Il va de soit que tout composant USB doit pouvoir être capable de supporter ce type de transfert.

Il existe sur le marché de nombreux périphériques USB. Il a fallu, lors de la création de la norme USB, trouver un dispositif pour reconnaître chaque composant USB. Cela était indispensable puisque l'USB devait être un dispositif plug & play. Lors du branchement du périphérique, le « host » autrement dit plus communément l'ordinateur, doit reconnaître tous les périphériques qui lui sont branchés.

Les descripteurs les plus courants sont : les descripteurs d'Appareils, de Configurations, d'Interfaces et d'Endpoint.

### **II.7.a -Le Descripteur d'appareil :**

Ce type de descripteur donne les informations générales. C'est le premier descripteur que vient lire le host. Comme mentionné précédemment, il y a une très grande diversité de composants USB, il y a des propriétés communes à chaque dispositif d'USB comme par exemple le numéro de spécification USB qui est présent dans toutes les configurations, de même pour les numéros d'identifiant de produit et de vendeur. (Product ID et Vendor ID).

Un ordinateur peut, uniquement avec ces informations contenues dans le descripteur d'appareil, reconnaître un composant USB. Il donne également des renseignements sur le PIPE de communication par défaut qui est utilisé pour la configuration.

### **II.7.b -Le Descripteur de Configuration :**

Un appareil USB peut avoir plusieurs configurations différentes alors que la majorité des appareils sont simples et n'en ont qu'une. Le Descripteur de Configuration précise la façon dont l'appareil est alimenté, quelle est sa consommation électrique maximale, le nombre d'interfaces qu'il possède. Il est donc possible d'avoir 2 configurations, quand il est alimenté par le bus et une autre quand il est alimenté par le secteur. Comme ceci est un "en tête " de descripteur d'interface, il est aussi possible d'avoir une configuration utilisant un mode de transfert différent de celui d'une autre configuration.

### II.7.c -Le Descripteur d'Interface :

Une interface peut être considérée comme un ensemble d'Endpoint .

Un Endpoint est en quelque sorte l'extrémité d'un « PIPE ». Un « PIPE » est une sorte de tuyau par lequel transitent les données vers le host. Un descripteur d'interface communique une information unique à tous ses Endpoints.

Si une configuration est choisie, toutes ses interfaces sont actives et par conséquent, aucun Endpoint ne peut être relié à des interfaces différentes sous la même configuration. Les interfaces sous différentes configurations peuvent partager des Endpoints.

### II.7.d- Le descripteur d'Endpoint :

Un descripteur d'Endpoint indique la direction du transfert (IN ou OUT), ses types de transfert (ISOCRONOUS, BULK, INTERRUPTION ou CONTROL), ainsi que d'autres Informations.

En fait, l'ordinateur « le host » communique uniquement avec ces Endpoints.

Tous les transferts de paquet de données transitant sur le bus proviennent d'un Endpoint ou sont envoyés à un Endpoint. Généralement les Endpoints correspondent aux Entrées-Sorties ou au registre du dispositif USB.

Le nombre maximum d'Endpoint est différent selon que l'on utilise un USB Low Speed ou un USB High speed. Un dispositif USB High Speed peut supporter jusqu'à 15 Endpoints tandis qu'un dispositif USB Low Speed ne peut supporter que 3 Endpoints.

### II.8 -Délai de propagation :

La longueur maximale exploitable est conditionnée par les temps de propagation le long du câble (30ns par segment), et à travers les hubs (40ns par HUB), et les temps de réponse de la fonction adressée (700ns maxi). Sachant que le temps de propagation (aller-retour) entre le host et la dite fonction ne doit pas dépasser 1300ns.

En outre, lorsqu'on dispose plusieurs HUBs en cascade, il ne faut pas sous estimé les contraintes d'alimentation et la chute de tension le long du trajet : s'il s'agit d'un périphérique alimenté via le bus USB il est clair qu'il doit pouvoir fonctionner avec une tension notablement inférieure à 5V.

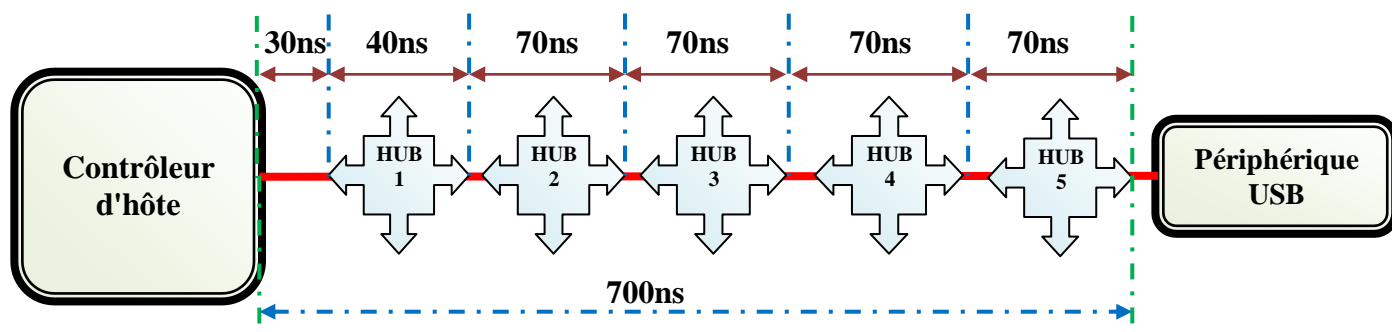


Figure I.12 : Temps de propagation d'une trame USB

# CHAPITRE II

ETUDE DE L'USB

AUTOUR DU

MICROCONTRÔLEUR

PIC 18F4550

## **Chapitre II :**

### **Etude de l'USB autour du microcontrôleur PIC 18F4550**

#### **I- Introduction :**

Le PIC 18F4550, commercialisé à partir de l'année 2004, a une gamme de dispositifs qui consomme très peu ce qui réduit de manière significative la consommation d'énergie lors du fonctionnement. C'est un microcontrôleur de technologie nanoWatt. La vitesse de ce PIC peut aller jusqu'à 48 MHz.

#### **II- Caractéristiques générales du pic 18F4550**

##### **II.1- USB :**

Le PIC18F4550 inclut un module USB. C'est un module de communications qui est conforme avec USB Specification Revision2.0. Le module soutient les deux vitesses de communication low-speed et full-speed pour soutenir tous les types de transfert de données. Il inclut également son propre émetteur récepteur avec un régulateur 3.3v et il a la possibilité d'utilisation d'un émetteur récepteur et d'un régulateur de tension externes [5].

##### **II.2- La Mémoire flash :**

Un espace mémoire de 32 K octets est disponible. Les cellules de la mémoire de programme et la mémoire EEPROM sont évaluées pour durer beaucoup de milliers de cycles d'efface/écrire – jusqu' à 100.000 fois pour la mémoire de programme et 1.000.000 fois pour EEPROM. La conservation des données en dehors d'actualisation est estimée pour être supérieure à 40 ans.

##### **II.2.1- Auto-Programmabilité :**

Ces dispositifs peuvent charger un programme à leurs propres espaces mémoire de programme sous la commande interne de logiciel.

En utilisant un programme bootloader, situé dans le bloc de démarrage (Boot Block) qui est protégé et placé au dessus de la mémoire de programme. Il devient possible de créer une application qui peut se mettre à jour dans son environnement de travail.

### **II.3- Jeu d'Instruction Prolongée :**

Un jeu d'instruction de 75 instructions avec une prolongation facultative pour le PIC18F4550, ce qui ajoute 8 nouvelles instructions. Cette prolongation facilite la configuration du PIC et est conçue spécifiquement pour optimiser le code d'application à l'origine développé dans un langage haut niveau tel que le C.

### **II.4- Module CCP amélioré :**

C'est un module qui nous a permis, parmi ses options, de faire fonctionner le mode MLI (PWM).

En mode MLI, le module fournit 1, 2 ou 4 sorties modulées pour faire contrôler les hacheurs en half-bridge et full-bridge. Ce module sera utilisé pour la commande du hacheur.

### **II.5- Convertisseur analogique-numérique 10-bit :**

Le PIC18F4550 possède un convertisseur 10 bits à 13 canaux A/D avec un temps d'acquisition programmable.

### **Les ports E/S :**

LE 18F4550 a 5 ports bidirectionnels (port A, port B, port C, port D, port E).

Quelques broches des ports d'E/S sont multiplexées avec des fonctions alternatives.

RE3 est multiplexé avec MCLR et il est disponible seulement lorsque les resets de MCLR sont désactivés.

OSC1/CLKI et OSC2/CLKO sont disponibles seulement dans des modes "select oscillator" et quand ces pins ne sont pas utilisés en tant que E/S numérique.



Tableau II.1 : Caractéristique de 18F4550.

caractéristique	PIC 18F4550
Fréquence d'opération	DC - 48MHZ
Mémoire de programme (Octets)	32768
Mémoire de programme (instructions)	16384
Mémoire de données (Octets)	2048
Mémoire de données EEPROM	256
Les sources d'interruption	20
Ports E/S	Port A, B, C, D, E
Timers	4
Modules CCP	1
Modules CCP amélioré	1
Communication série	MSSP Enhanced USART
Module USB	1
SSP	OUI
Convertisseur A/N de 10 bits	13 canaux d'entrée
Comparateurs	2
Reset	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	oui
Programmable Brown-out Reset	oui
Jeu d'instruction	Jeu d'instruction prolongé de 83 instructions
Packages	40 pin PDIP 44 pin QFN 44 pin TQFP

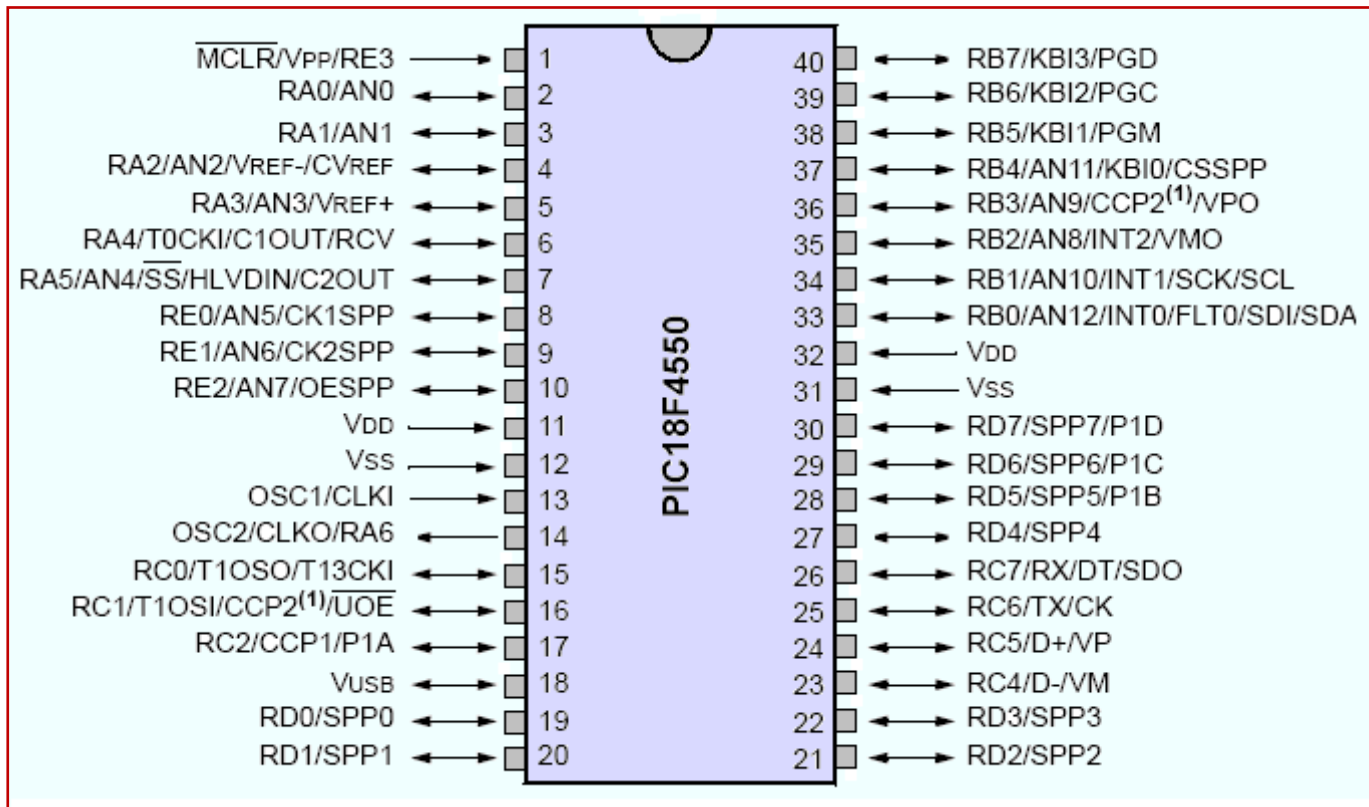


Figure II.1: Le PIC 18F4550

### III- UNIVERSAL SERIAL BUS (USB):

Cette section décrit les détails de ce périphérique USB.

La famille des PIC18FX455/x550 contient un Moteur d'interface série (SIE) compatible à full-speed et low-speed qui permet la mise en œuvre des communications rapides entre l'ordinateur et le microcontrôleur.

#### III.1- Emetteur - Récepteur (Transceiver) :

Le SIE peut être connecté directement à l'USB en utilisant l'émetteur récepteur interne, ou indirectement en utilisant l'émetteur récepteur externe. Un régulateur 3.3v interne est également disponible pour actionner l'émetteur récepteur interne dans des applications 5V.

La figure II.2 présente une vue générale sur le périphérique USB et ses dispositifs [5].

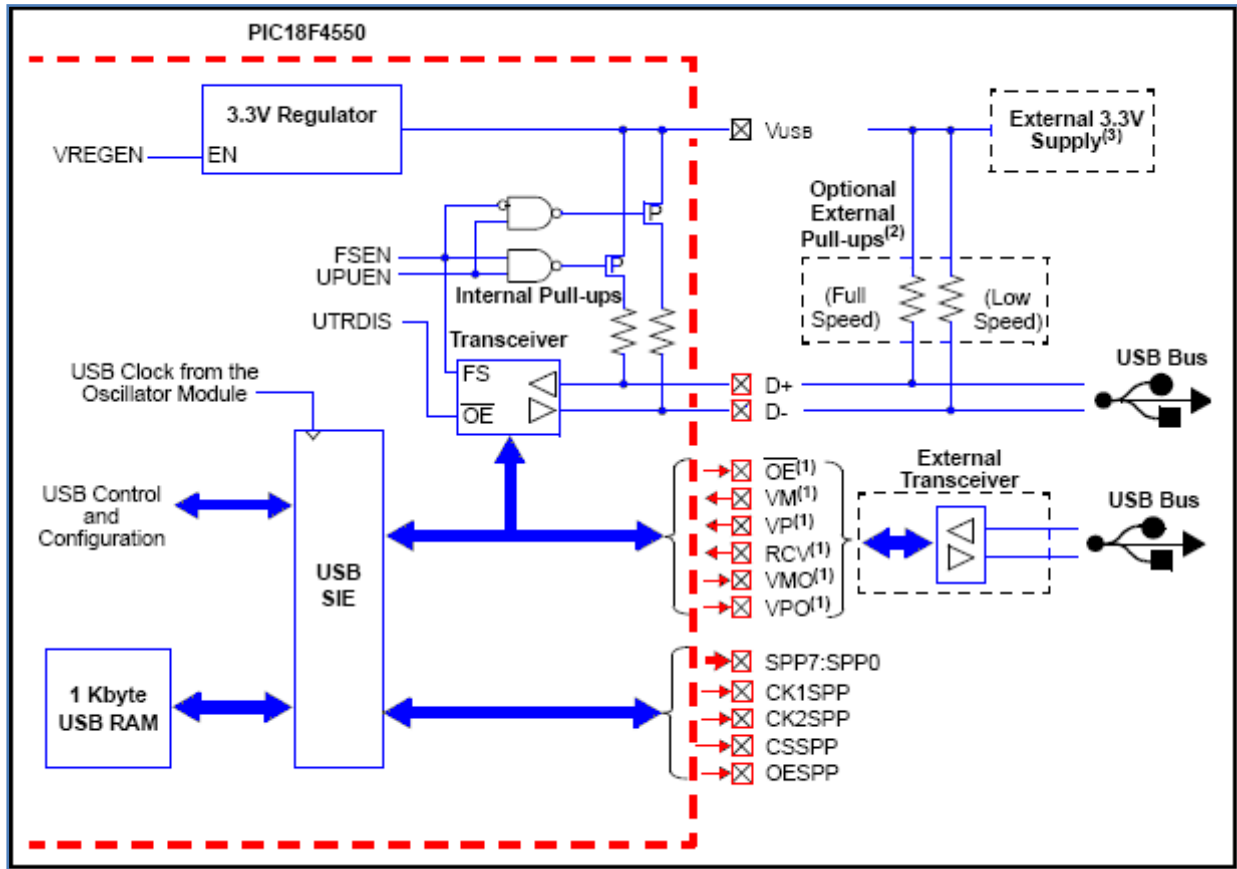


Figure II.2 : Une vue générale du module USB du 18F4550.

### III.1 .a- Émetteur récepteur interne (Internal Transceiver):

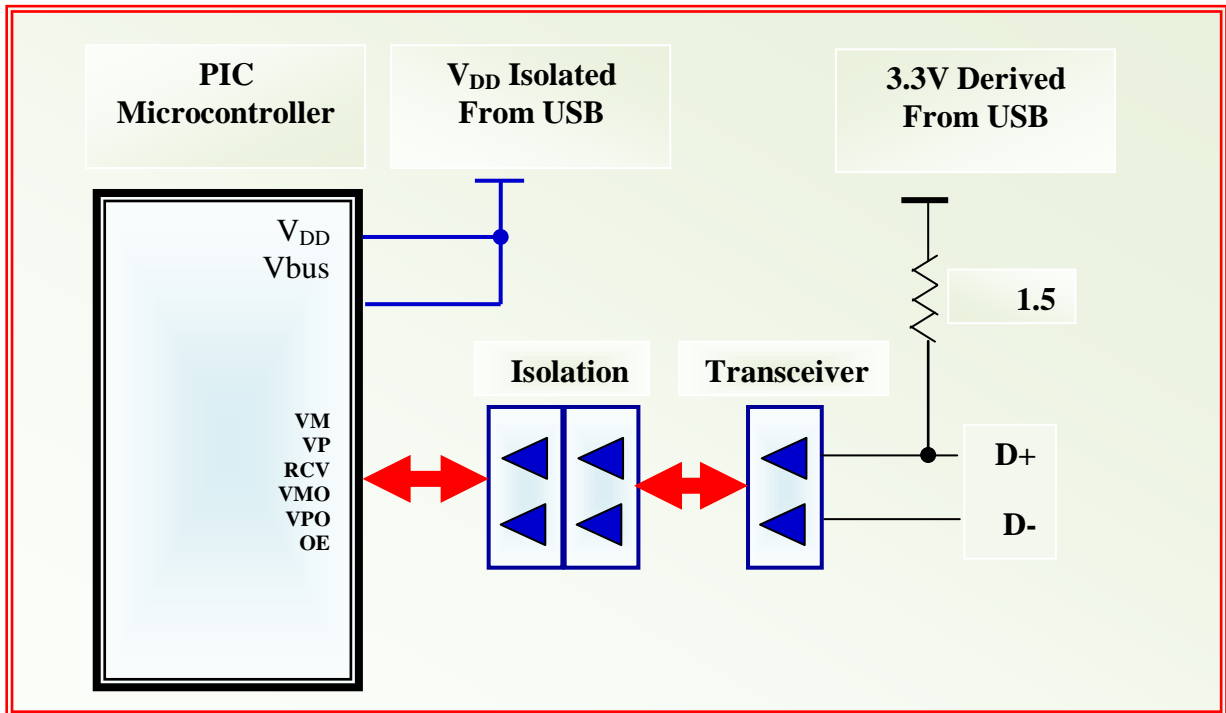
Le périphérique USB possède un émetteur récepteur intégré, conforme aux vitesses d'USB 2.0 (full et low speed), intérieurement relié au SIE. Les spécifications d'USB exigent l'opération sous 3.3v pour les communications. Cependant, le reste de la puce peut être alimenté avec une tension plus élevée. Ainsi, l'alimentation de l'émetteur récepteur est assurée via une source séparée (VUSB).

### III.1 .b- Émetteur récepteur externe (External Transceiver):

Ce module est conçu pour supporter l'usage en émetteur récepteur externe (hors puce). L'émetteur récepteur externe est destiné pour les applications où les conditions physiques dictent l'emplacement de l'émetteur récepteur pour être à l'extérieur du SIE.

Par exemple, les applications qui exigent l'isolation dans l'USB pourraient utiliser un émetteur récepteur externe par certaine isolation au SIE du microcontrôleur.

La configuration suivante montre un schéma simplifié pour une configuration full speed en utilisant un émetteur récepteur externe avec isolation.



**Figure II.3 :** Emetteur Récepteur Externe Typique Avec Isolation.

Il y a 6 signaux pour communiquer et contrôler un émetteur récepteur externe :

- ▲ VM : l'entrée qui vient de la ligne asymétrique de D<sup>-</sup>
- ▲ VP : l'entrée qui vient de la ligne asymétrique de D<sup>+</sup>
- ▲ RCV : l'entrée qui vient du récepteur différentiel
- ▲ VMO : sortie vers le gestionnaire de ligne différentiel
- ▲ VPO : sortie vers le gestionnaire de ligne différentiel
- ▲  $\overline{OE}$  : Validation de La sortie.

Les signaux de VPO et de VMO sont des sorties du SIE à l'émetteur récepteur externe.

Le signal RCV est la sortie de l'émetteur récepteur externe au SIE. Il représente les signaux différentiels du bus série translatés dans un train d'impulsion simple.

Les signaux de VM et de VP sont utilisés comme moyen pour enregistrer des conditions sur le bus séquentiel au SIE qui ne peut pas être capturé avec le signal du récepteur.

Les combinaisons des états de ces signaux et de leur traduction sont énumérées dans le tableau II.2.

**Tableau II.2 :** Les combinaisons des états de ces signaux et de leur traduction. [5]

VP	VM	L'état de bus	VPO	VMO	L'état de bus
0	0	Signal –Ended Zero	0	0	Signal –Ended Zero
0	1	Low speed	0	1	Differential « 0 »
1	0	High speed	1	0	Differential « 1 »
1	1	error	1	1	Illegal Condition
Les Entrées de l'émetteur récepteur			Les Sorties de l'émetteur récepteur		

Le moniteur d'USB OE fournit l'indication pour savoir si le SIE écoute le bus ou le pilotage de bus est actif. Ceci est permis par défaut quand on utilise un émetteur récepteur externe ou quand  $UCFG\langle 6 \rangle = 1$ .

### III.2- choix de la vitesse de transfert :

Le choix de la vitesse de transfert des données (low-speed ou full speed) se fait par le choix des résistances de pull-up.

Il y a deux types de résistance de pull-up :

- ▲ -Les Résistances pull-up internes.
- ▲ -Les Résistances pull-up externes.

#### III.2.1- Les Résistances pull-up internes :

Le PIC18F4550 a des résistances pull-up intégrées conçues pour le choix de la vitesse des transferts des données (low-speed ou full speed).

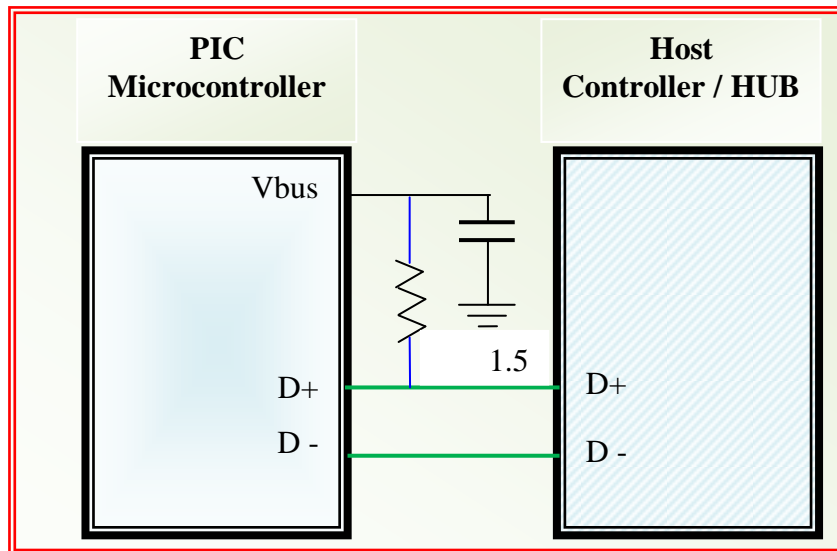
Le bit d'UPUEN ( $UCFG\langle 4 \rangle$ ) valide le pull-up interne.

#### III.2.2- Les Résistances pull-up externes :

La broche VUSB peut être utilisée pour tirer vers le haut la tension D+ ou D-.

La résistance pull-up doit être 1,5 k ( $\pm 5\%$ ) selon les exigences des caractéristiques d'USB.

La figure II.4 montre un exemple.



**Figure II.4 :** Les circuits externes pour full ou low speed

La configuration ci-dessus montre la connexion typique pour une configuration full speed en utilisant le régulateur interne et une résistance de pull-up externe.

#### IV- Les Interruptions du module USB:

Le module USB peut produire des modes d'interruption multiples.

Pour faciliter toutes ces sources d'interruption, le module USB est équipé de sa propre structure de logique d'interruption, semblable à celui du microcontrôleur.

Des interruptions d'USB sont permises avec un ensemble de compteurs d'instructions et désactivées avec un ensemble séparé d'indicateur de registre.

Toutes les sources sont dirigées vers une demande simple d'interruption d'USB, USBIF (PIR2<5 >), dans la logique de l'interruption du microcontrôleur.

Le schéma de la figure II.5 montre la logique d'interruption pour le module USB.

Il y a deux couches de registres d'interruption dans le module USB.

- ☞ Le niveau supérieur se compose des modes d'interruptions globales d'USB; celles-ci sont validées dans les registres UIE et UIR.
- ☞ Le deuxième niveau se compose des cas d'erreur d'USB, qui sont validées et marquées dans les registres UEIR et UEIE.

Une interruption quelconque déclenche un indicateur d'interruption d'erreur d'USB (UERRIF) au niveau supérieur.

Le schéma de la figure II.6 montre quelques événements communs dans une trame USB et leurs interruptions correspondantes.

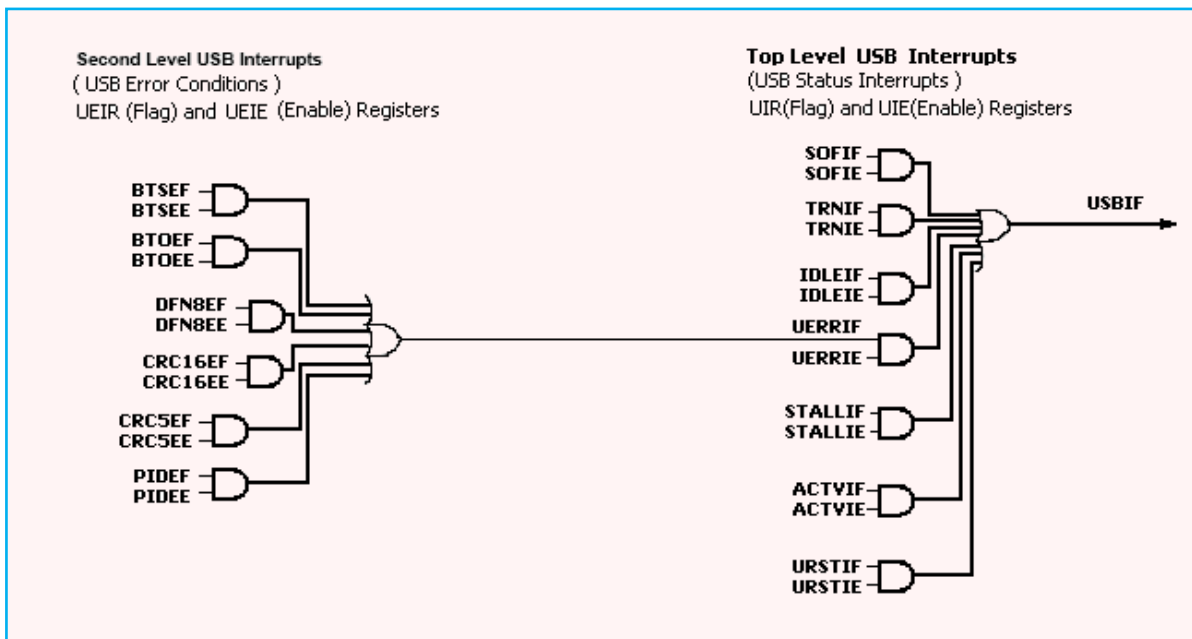


Figure II.5: Logique d'interruption d'USB

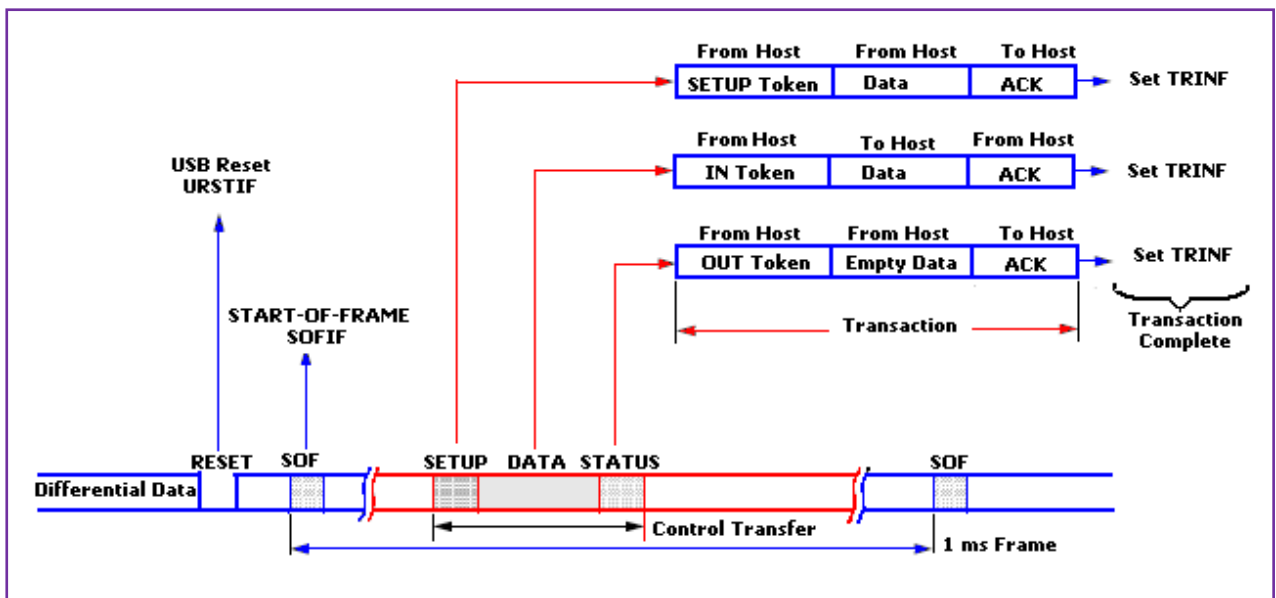


Figure II.6 : Exemple d'une transaction d'USB et des événements d'interruption

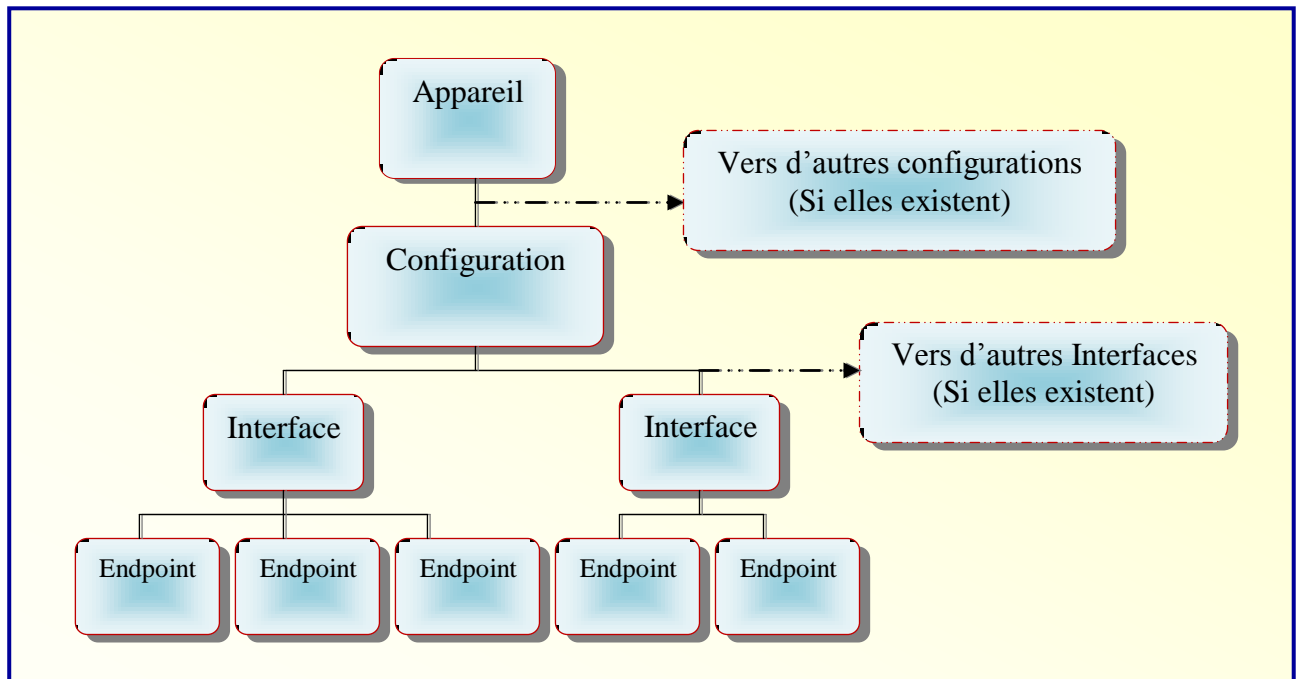
Note : Le transfert montré ici est seulement un exemple montrant les événements qui peuvent se produire pour chaque transaction.

## V- Aperçu sur l'USB autour du PIC :

Cette section présente une partie des concepts de base de l'USB et des informations utiles nécessaires pour concevoir un dispositif USB.

### V. 1- Structure en couches (LAYERED FRAMEWORK):

La fonctionnalité du dispositif USB est structurée dans une structure en couches représentées graphiquement sur le schéma de la figure II.7.



**Figure II.7:** Les Couches USB

Chaque niveau est associé à un niveau fonctionnel dans l'USB.

La couche la plus élevée est la couche configuration.

Un dispositif peut avoir des configurations multiples.

Par exemple, le PIC peut avoir une alimentation électrique basée sur le mode Self-Power seulement ou basé sur le mode d'alimentation électrique du bus.

Pour chaque configuration, il peut y avoir des interfaces multiples.

Chaque interface peut supporter un mode particulier de cette configuration.

Au-dessous de l'interface, c'est l'Endpoint. Les données sont directement déplacées à ce niveau. Il peut y avoir jusqu'à 16 endpoints bidirectionnels.



L'Endpoint 0 est toujours un Endpoint de commande et par défaut, quand le dispositif est sur le bus, l'Endpoint 0 doit être disponible pour configurer le PIC [5].

## V.2- Les Trames :

Les informations transférées sur le bus sont groupées dans des trames de 1 ms. Chaque trame peut contenir beaucoup de transactions à des divers dispositifs et Endpoints.

## V.3- L'Alimentation Electrique :

L'alimentation électrique est fournie par le bus USB. Les PICs peuvent être autoalimentés ou alimentés via le bus USB.

Les dispositifs autoalimentés tirent l'alimentation électrique d'une source extérieure, alors que les autres dispositifs utilisent l'alimentation électrique assurée à partir du bus.

La norme USB limite l'alimentation électrique prise du bus. Chaque dispositif est alimenté de 100 mA à 5V approximativement (une charge d'unité). L'alimentation électrique supplémentaire peut être demandée à partir d'un maximum de 500 mA

La norme USB définit également un mode de suspension. Dans cette situation, le courant doit être limité à 500  $\mu$  A.

Un dispositif doit écrire un état de suspension après 3ms de l'inactivité (c.-à-d., aucune marque de SOF pour 3ms) [5] [17] .

## V.4- L'énumération du PIC 18F4550:

Quand le PIC est branché au bus, l'ordinateur écrit un processus d'énumération afin d'essayer d'identifier le PIC. L'ordinateur interroge le PIC, recueillant l'information telle que la puissance consommée, les débits et les tailles des données, le protocole et toute autre information descriptive. C'est les descripteurs qui contiennent cette information.

Un processus d'énumération typique est comme suit :

- 1 - USB reset : L'ordinateur Remet à l'état initial le PIC. Ainsi, le dispositif n'est pas configuré et n'a pas une adresse (adresse 0).
- 2 - Obtention du Descripteur du PIC (Device Descriptor) : l'ordinateur demande une petite partie du Device Descriptor.
- 3 - USB reset : l'ordinateur Remet encore le PIC à l'état initial.
- 4 - Placement de l'adresse : l'ordinateur assigne une adresse au PIC.

- 5 - Obtention du Descripteur du PIC (Device Descriptor) : l'ordinateur recherche le Device Descriptor, recueillant l'information telle que le constructeur, type de PIC, taille maximale du paquet de commande.
- 6 - Obtention des Descripteurs de Configuration.
- 7 - Obtention de tous les autres descripteurs. (Les descripteurs d'Interface et d'Endpoint)
- 8 - Placement d'une configuration.

Il est noté que :

- Le processus exact d'énumération dépend du host.
- Lors de l'installation d'un nouveau périphérique USB, les descripteurs fournissent les informations concernant le PID (Product ID) et le VID (Vendor ID). C'est grâce à ces deux valeurs que l'ordinateur peut reconnaître l'identité du composant. Comme mentionné précédemment la réglementation des VID est très stricte. Chaque fabricant possède un VID et c'est grâce à cette valeur codée sur 16 bits que l'on peut retrouver le fabricant du composant. Chaque fabricant ayant plusieurs produits à leurs actifs, et ils sont différenciés avec le PID codé également sur 16bits.

L'allocation des PID, contrairement aux VID, est faite par le constructeur du dispositif.

## **VI- Conception et implémentation de l'USB autour du PIC :**

Pour communiquer avec le port USB on utilise une méthode développée par Microchip en 2004 (**RS-232 Emulation over USB, Migrating Applications to USB from RS-232**) [9].

L'avantage de cette méthode est que l'application Windows verra la connexion USB comme connexion port COM et ainsi, elle n'exige aucun changement au logiciel existant.

Un autre avantage est que cette méthode utilise un driver Windows compatible avec le Windows 98 et les versions postérieures, rendant ainsi le développement d'un nouveau driver inutile [9].

### **VI.1- Vue Générale :**

L'application Windows voit la connexion USB physique en tant que port COM et communique avec le dispositif branché via le câble USB.

Lorsqu'on branche notre carte à l'USB, l'application Windows peut voir la connexion USB comme un port COM virtuelle, et elle peut communiquer avec la carte par l'intermédiaire des services fournis par les deux drivers de Windows, usbser.sys et ccport.sys.

Les zones qui exigent des changements sont le matériel et le code d'application.

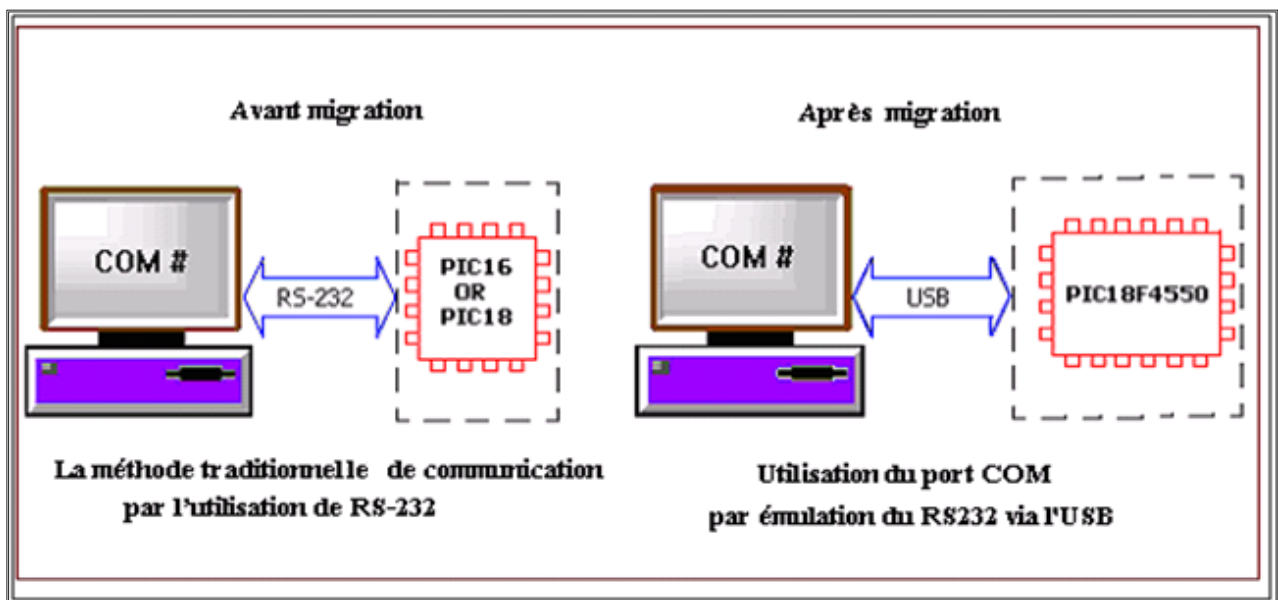
Pour le matériel, un microcontrôleur avec un périphérique USB full-speed est exigé pour mettre en application cette solution. Le PIC18F4550 est un bon choix.

Les modifications aux codes d'applications (RS-232) existant sont minimales, seulement des petits changements sont nécessaires pour appeler les nouvelles fonctions de « USB UART » fournies en tant qu'élément du cadre d'USB écrit en C.

Cette méthode de Microchip fournit les avantages suivants :

- ✍ Elle a peu ou pas d'impact sur l'application logiciel de l'ordinateur.
- ✍ Elle exige des changements minimaux aux codes d'applications (RS-232) si ils existent.
- ✍ Elle minimise la durée du cycle de développement.
- ✍ Elle élimine la nécessité de supporter et mettre à jour un driver de Windows qui est une tâche très exigeante.

Puisque le protocole USB traite déjà les détails de transmission inférieure, le concept de la vitesse, le bit de parité et la commande de flux de la norme RS-232 deviennent abstraits.



**Figure II.8:** Emulation de communication RS-232 via le bus USB.

## VI.2- CDC:

La spécification CDC (Communication Device Class) définit beaucoup de modèles de transmission. Toutes les références à la spécification CDC dans ce travail se rapportent à la version 1.1. Le driver de Microsoft Windows, usbser.sys se conforme à cette spécification [9].

Par conséquent, le dispositif utilisé doit également être conçu pour se conformer à cette spécification afin d'utiliser ce driver existant de Windows.

En résumé, deux interfaces d'USB sont exigées :

La première est l'interface de classe de transmission, en utilisant un IN Endpoint d'interruption. Cette interface est utilisée pour informer le hôte USB sur le mode actuel de la connexion du dispositif USB.

La seconde est l'interface de classe des données, en utilisant OUT Bulk endpoint et un IN Bulk Endpoint. Cette interface est utilisée pour transférer les octets de données brutes qui seraient normalement transférées sur la vraie interface RS-232. [9]

### VI.3- Les fonctions d' UART USB:

Toutes les références aux fonctions d'USB UART dans ce document se rapportent à des fonctions fournies dans la version 1.0 du driver de CDC firmware. Ce driver est fourni en tant qu'élément du Microchip USB firmware [9].

Les fonctions fournies dans cette bibliothèque sont très semblables dans la fonctionnalité à celles fournies dans la bibliothèque du MPLAB C18 USART library.

Des fonctions spécifiques sont énumérées dans le tableau suivant :

**Tableau II.3:** La version 1,0 des firmwares de l'émulateur du RS-232 de Microchip.

<u>Fonctions</u>	<u>Description</u>
<b>putrsUSBUSART</b>	Pour écrire une chaîne de caractères de mémoire de programme à l'USB.
<b>putsUSBUSART</b>	Pour écrire une chaîne de caractères de mémoire de données à l'USB.
<b>mUSBUSARTTxRom</b>	Pour écrire une chaîne de caractères de longueur connue de mémoire de programme à l'USB.
<b>mUSBUSARTTxRam</b>	Pour écrire une chaîne de caractères de longueur connue de mémoire de données à l'USB.
<b>mUSBUSARTIsTxTrfReady</b>	Pour tester si le driver est prêt à recevoir une nouvelle chaîne de caractères pour écrire à l'USB ?
<b>getsUSBUSART</b>	Pour lire une chaîne de caractères de l'USB.
<b>mCDCGetRxLength</b>	Pour Lire la longueur de la dernière chaîne de caractères lue de l'USB.

### VI.4- Caractéristiques de l'émulateur du RS-232 de Microchip :

- Un code footprint relativement petit de 3 Koctets pour la bibliothèque de firmware

- Utilisation de mémoire de données d'approximativement 50 octets (non compris la mémoire tampon de données)
- Vitesse maximum d'environ 640 Kbits/s .
- Le flux de données est commandé entièrement par le protocole USB.
- N'exige pas de drivers supplémentaires ; tous les fichiers nécessaires, y compris les fichiers .inf sont inclus pour le Windows XP et le Windows 2000,

### VI.5 - Le code d'application :

Le code d'application est écrit en langage C avec l'utilisation de compilateur MCC18 de Microchip et MPLAB IDE.

La forme de projet sous MPLAB est illustrée sur la figure II.9.

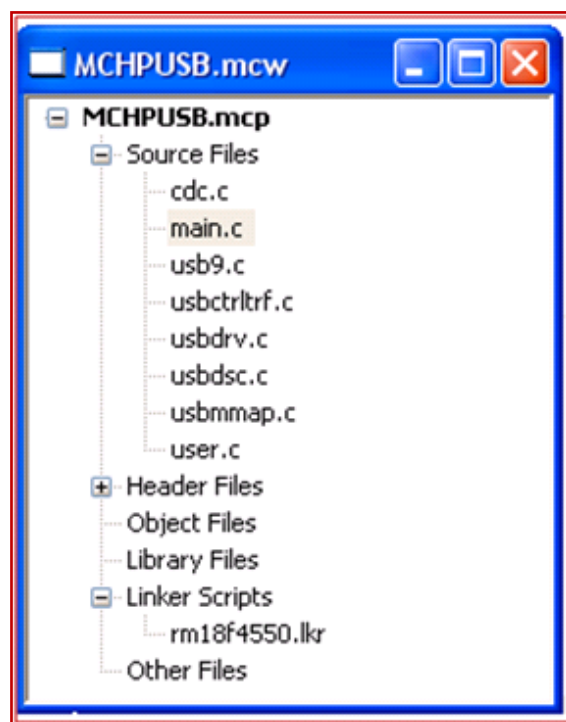


Figure II.9: La forme de projet sous MPLAB

Chaque fichier dans ce projet a son rôle pour communiquer avec l'ordinateur et contrôler l'étage de puissance afin de piloter le moteur à courant continu via le bus USB.

Dans ce qui suit le rôle de chaque fichier est expliqué.

#### **Main.c :**

C'est le fichier qui fait l'initialisation du système et appelle les fonctions principales pour adapter la liaison USB avec l'ordinateur.

#### **cdc.c :**

Dans ce fichier on trouve les routines qui contrôlent les paquets des données d'installation pour voir si elles sont bien traitées.

On trouve aussi la routine qui est responsable de l'encapsulation des données pour qu'elles soient conformes à la forme des paquets de la spécification USB.

On trouve aussi les fonctions spécifiques du tableau II.3.

### **Usb9.c :**

Dans ce fichier on trouve les fonctions qui réinitialisent les registres du microcontrôleur quand il est opérationnel et configurent les endpoints durant et après la phase de l'énumération.

### **Usbctrlrf.c :**

Ce sous-programme est un expéditeur de tâches à 3 étapes.

- ▲ initialise le contrôle de transfert.
- ▲ invite chacun des modules (USB9, HID, CDC, MSD, ...) qui peut servir la demande d'installation à partir du centre serveur.
- ▲ Une fois chacun des modules a eu une chance de vérifier s'il est le responsable de servir la requête, alors il contrôle la direction du transfert pour déterminer comment préparer EP0 pour le transfert de contrôle.

Un transfert de contrôle se compose de beaucoup de transactions USB.

En transférant les données par des transactions multiples, il est important de maintenir la source de données, la destination de données, et les données de comptage. Ces trois paramètres sont enregistrés dans pSrc, pDst, et wCount. Un indicateur est utilisé pour noter si la source de données est la ROM ou RAM.

### **usbdrv.c :**

Dans ce fichier on trouve les routines qui ont les fonctions suivantes :

- active le module USB.
- désactive le module USB.

On trouve aussi la routine qui contrôle toutes les interruptions d'USB pendant les états suivants :

DETACHED->ATTACHED -> POWERED -> DEFAULT ->ADDRESS\_PENDING ->ADDRESSED -> CONFIGURED -> READY.

### **usbds.c :**

Ce fichier contient les descripteurs d'USB. Il est utilisé avec le fichier usbds.h. Quand un descripteur est ajouté ou retiré du descripteur principal de configuration, c.-à-d. CFG01, l'utilisateur doit également changer la structure du descripteur définie dans le fichier usbds.h.

La structure sizeof (CFG01) est utilisée pour calculer la taille du descripteur.

Un descripteur typique de configuration se compose de :

- Au moins un descripteur de configuration (USB\_cfg\_dsc)
- Un ou plusieurs descripteurs d'interface (USB\_intf\_dsc)
- Un ou plusieurs descripteurs d'Endpoint (USB\_ep\_dsc)

### **usbmmmap.c :( USB Memory Map)**

Ce fichier est le gestionnaire de mémoire USB; il sert à programmer l'allocation de mémoire aux Endpoint d'USB.

Chaque Endpoint exige d'avoir un ensemble des registres du descripteur de mémoire tampon (BDT).

Un BDT est 4-byte et a un emplacement spécifique de RAM pour chaque endpoint.

Le BDT pour l'endpoint 0 out est situé à l'adresse 0x400 à 0x403 .

Le BDT pour l'endpoint 0 in est situé à l'adresse 0x404 à 0x407.

Le BDT pour l'endpoint 1 out est situé à l'adresse 0x408 à 0x40B.

Et ainsi de suite.

### **user.c :**

C'est le fichier où on peut placer nos routines de commande. Les routines de ce fichier seront expliquées ultérieurement.

## **VI.6- Identificateur de constructeur (VID) et Identificateur de produit (PID) :**

Dans notre projet on utilise le VID de MICROCHIP qui est (04D8), et on peut utiliser n'importe quel Pid , à condition qu'on met les mêmes pid et vid dans le fichier usbds.c et le fichier (.inf) qui est utilisé pour l'installation de notre carte .

## **VI.7- Drivers pour Microsoft Windows 2000 et Windows XP:**

Microsoft Windows n'a pas un fichier standard ( .inf) pour le driver CDC.

Les drivers sont, cependant, une partie de l'installation de Windows. La seule chose nécessaire à faire est de fournir un fichier (.inf) quand un dispositif CDC est connecté à un système Windows pour la première fois. Ce fichier INF contient le VID et le PID qui doivent être les mêmes avec ceux indiqués dans usbds.c [9].

---

# CHAPITRE III

## RÉALISATION DE LA COMMANDE DU MOTEUR À COURANT CONTINU



## Chapitre III :

### Réalisation de la commande du moteur à courant continu

#### I- Introduction :

Malgré leur grand succès durant ces dernières décennies, les ports séries et parallèles désertent progressivement les cartes mères, rendant certain applications inutiles.

Pour remédier à ce problème, il faut commencer à réfléchir à remplacer ou à adapter ces applications en utilisant le port USB, vu les avantages, cités auparavant, de ce dernier.

Le présent chapitre a en effet comme objectif la présentation d'une application du port USB dans la commande d'un moteur à courant continu.

A cet effet, une carte de commande à base du microcontrôleur PIC18F4550 associée à un étage de puissance, convertisseur DC-DC, ont été réalisés.

Le convertisseur DC-DC, hacheur en pont, est alimenté par une source continue externe, provenant d'une batterie ou d'un panneau solaire par exemple. Il est commandé par ordinateur via le port USB et la carte de commande selon une consigne de vitesse, de position(en boucle fermée), ou une tension désirée(en boucle ouverte).

La carte de commande reçoit les consignes via l'USB et détecte la vitesse du moteur par un capteur incrémental optique associée à une routine de comptage qui pilote le compteur quadrature LS7266.

La carte microcontrôleur recevra depuis le PC et via la liaison USB les informations de Position consigne, des paramètres d'asservissement, des paramètres de sécurité,....

Pour faire un affichage de toutes ses données sur l'interface utilisateur au niveau de l'ordinateur, La carte de commande envoie des données sur l'état de marche du moteur vers l'ordinateur via le câble USB.

Rappelons que ce montage n'est valable que pour un seul canal de commande.

Pour la boucle d'asservissement de vitesse ou de position on utilise le régulateur (PID), et pour améliorer les performances de notre boucle d'asservissement de position, un profil de vitesse est implémenté afin d'éviter le démarrage et le freinage brusque qui peuvent détruire notre mécanisme.

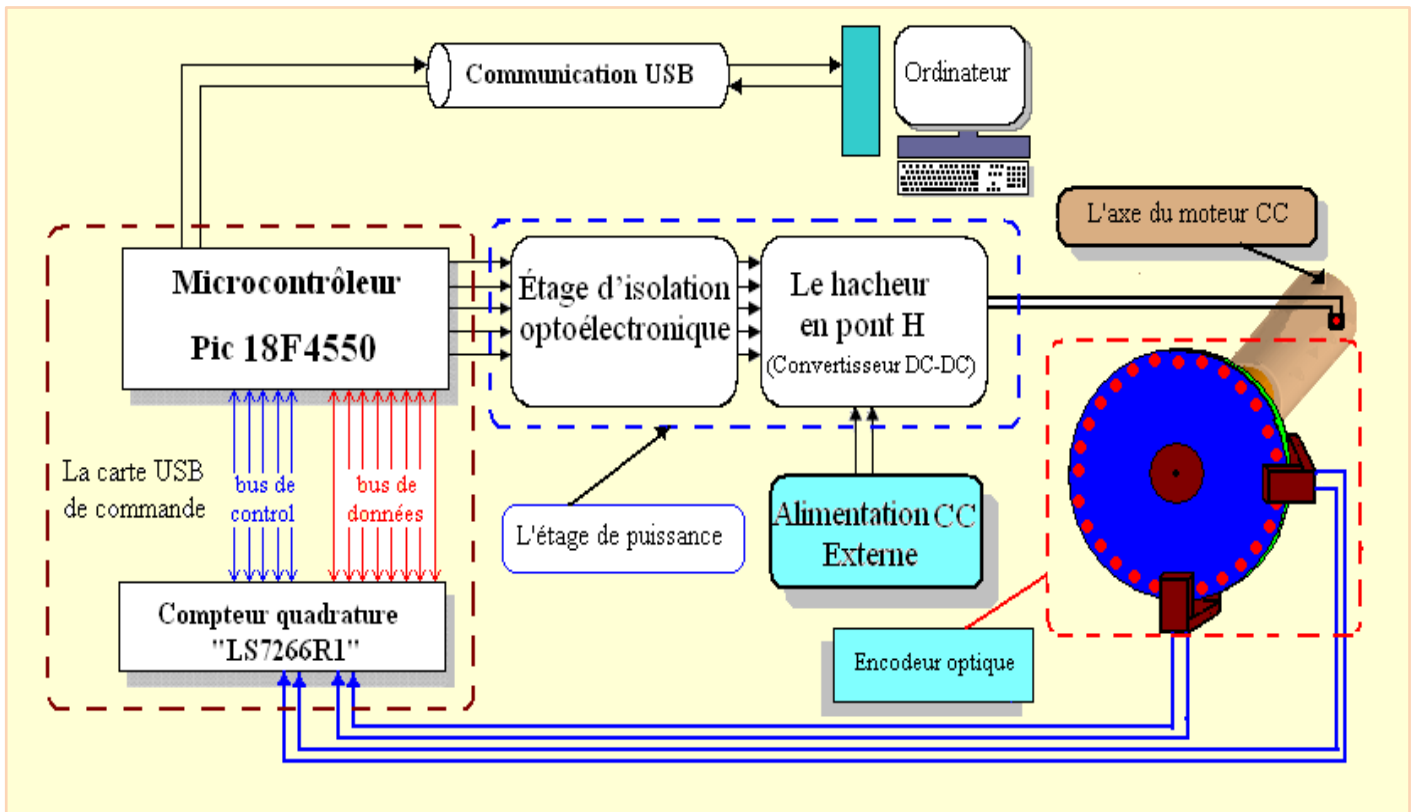


Figure III.1 : Représentation fonctionnelle du système globale

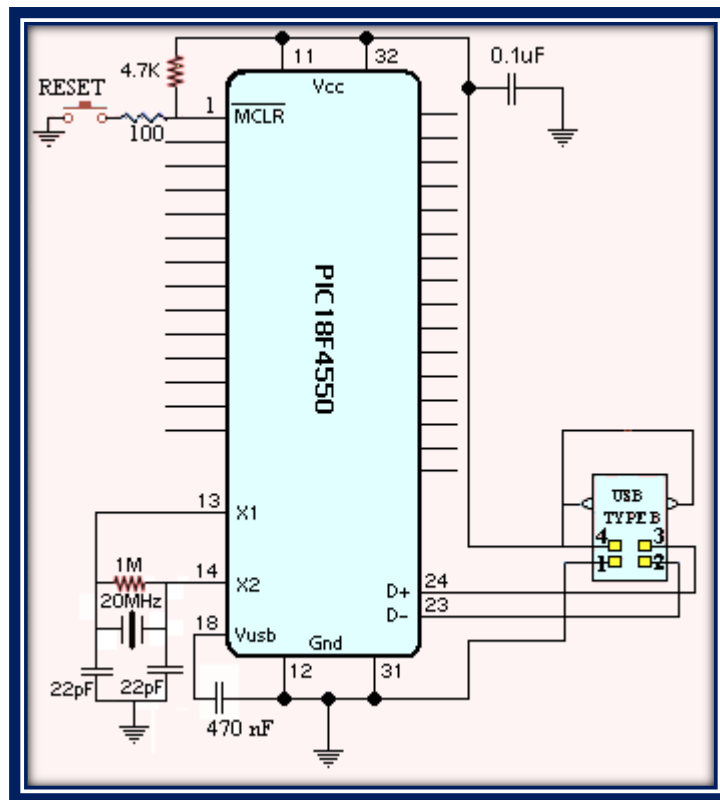


Figure III.2 : Le microcontrôleur 18F4550 et la liaison USB

## II- description du matériel :

### II.1- le hacheur :

L'étage de puissance est un hacheur Buck (figure III.3), où la tension d'entrée est prise d'une source continue externe et transférée vers le moteur selon le rapport cyclique choisi par l'utilisateur ou les besoins du moteur dictés par les consignes de vitesse et de position.

Les moteurs à courant continu sont utilisés pour le déplacement des robots, dans les machines, les bras mécaniques etc.... Ils peuvent être de différentes tailles, puissances, couples et prix.

On peut commander le moteur par l'intermédiaire d'un seul transistor que l'on fait conduire et que l'on bloque alternativement. Cela revient à faire varier le rapport cyclique du signal et donc à faire varier le courant moyen dans le moteur. L'inconvénient est que le moteur ne tourne que dans un sens.

L'idéal est de pouvoir faire une inversion du courant dans le bobinage du moteur. Il faut pour cela réaliser un montage quatre quadrants que l'on appelle en pont (en H), et qui met en œuvre 4 transistors.

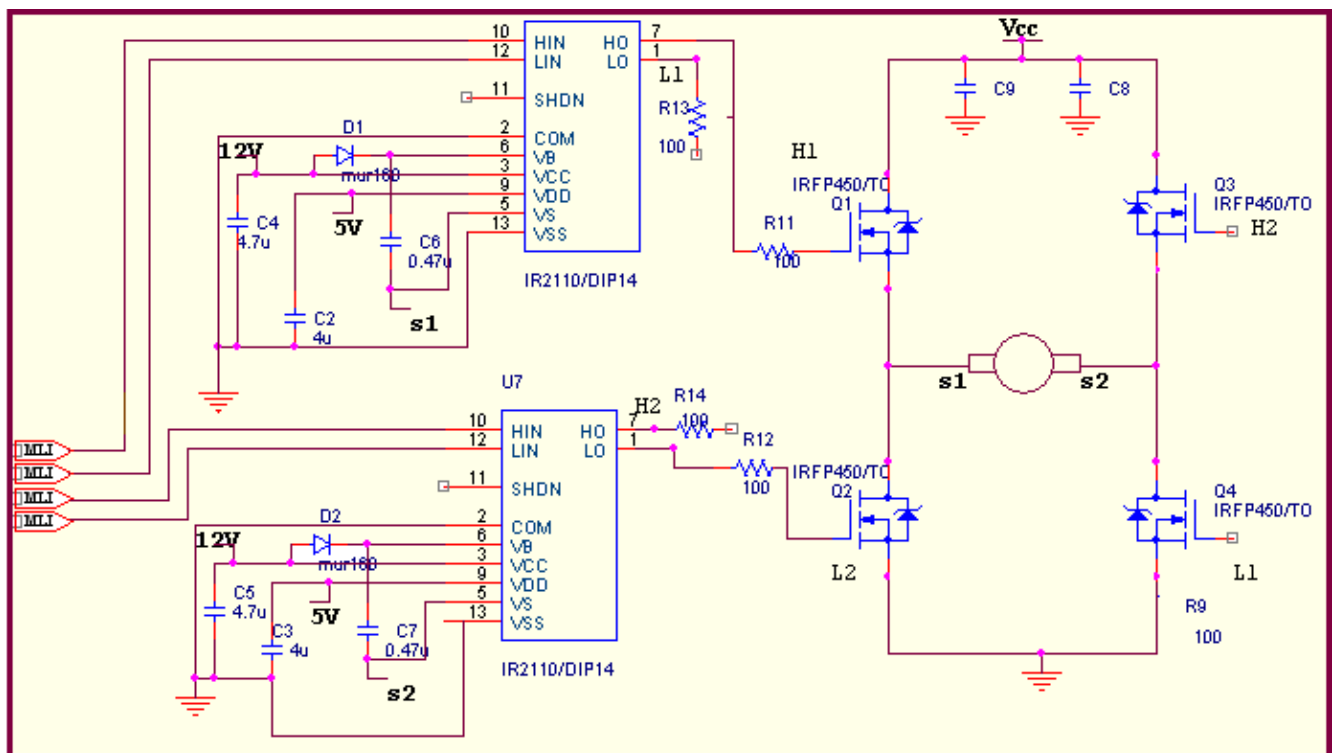


Figure III.3 : L'étage de puissance

On peut utiliser des transistors bipolaires. Sachant que les transistors doivent être commandés en courant ce qui est difficile pour les microcontrôleurs.

La solution est donc d'utiliser des transistors de type MOSFET, qui sont commandés en tension mais lors de la conduction ils ont des pertes en puissance égale à  $[ R_{ds(on)} \cdot I_d^2 ]$ .

Donc pour notre hacheur nous avons utilisé 4 transistors MOSFET canal N associés avec des dissipateurs de chaleur.

Le hacheur est en pont de quatre MOSFET de type IRFP 450 capable de supporter un courant jusqu'à 14A et une tension de 500v et  $R_{ds}(ON) = 0.400\Omega$ .

Pour la commande des transistors du coté bas il n'y a pas de problème car la tension de grille est supérieure à celle de source. Pour ceux du haut, il faut mettre un artifice afin d'avoir une tension de commande de grille au dessus de la tension de source de transistor. En effet le potentiel de la source du transistor du haut se trouve quasiment au potentiel  $V_{cc}$ .

On met en place deux drivers de MOSFET de type IR2110 pour élever les tensions des grilles qui doivent être supérieures à  $V_{cc}$ .

L'IR2110 est un driver de MOS, son rôle ici est de commander correctement un demi-pont de MOS, donc il nous faut deux drivers pour commander un pont complet. Ce circuit possède des pompes de charges intégrées qui permettent de générer la tension de grille du MOS-N "high side" (en haut).

La grille du Mos-n supérieur est flottante vis à vis de la charge.

## II.2- L'isolation optoélectronique :

Notre hacheur peut fonctionner avec des gammes de tensions et des courants élevés, donc il est nécessaire d'avoir une isolation optoélectronique pour protéger le microcontrôleur et l'ordinateur afin d'éviter les risques des surtensions.

L'isolation par des optocoupleurs (6N137) est une bonne solution car elle offre une isolation en tension jusqu'à 2500v, en plus ; elle a une vitesse de 10Mbits/s.

La MLI utilisée a une fréquence de 11 KHz avec une résolution de 1/1023, le 6N137 est donc bien approprié.

Il existe un petit problème, le 6N137 fonctionne avec une logique négative, la solution est donc de mettre un inverseur TTL 7404 entre les optocoupleurs et les drivers des MOSFET.

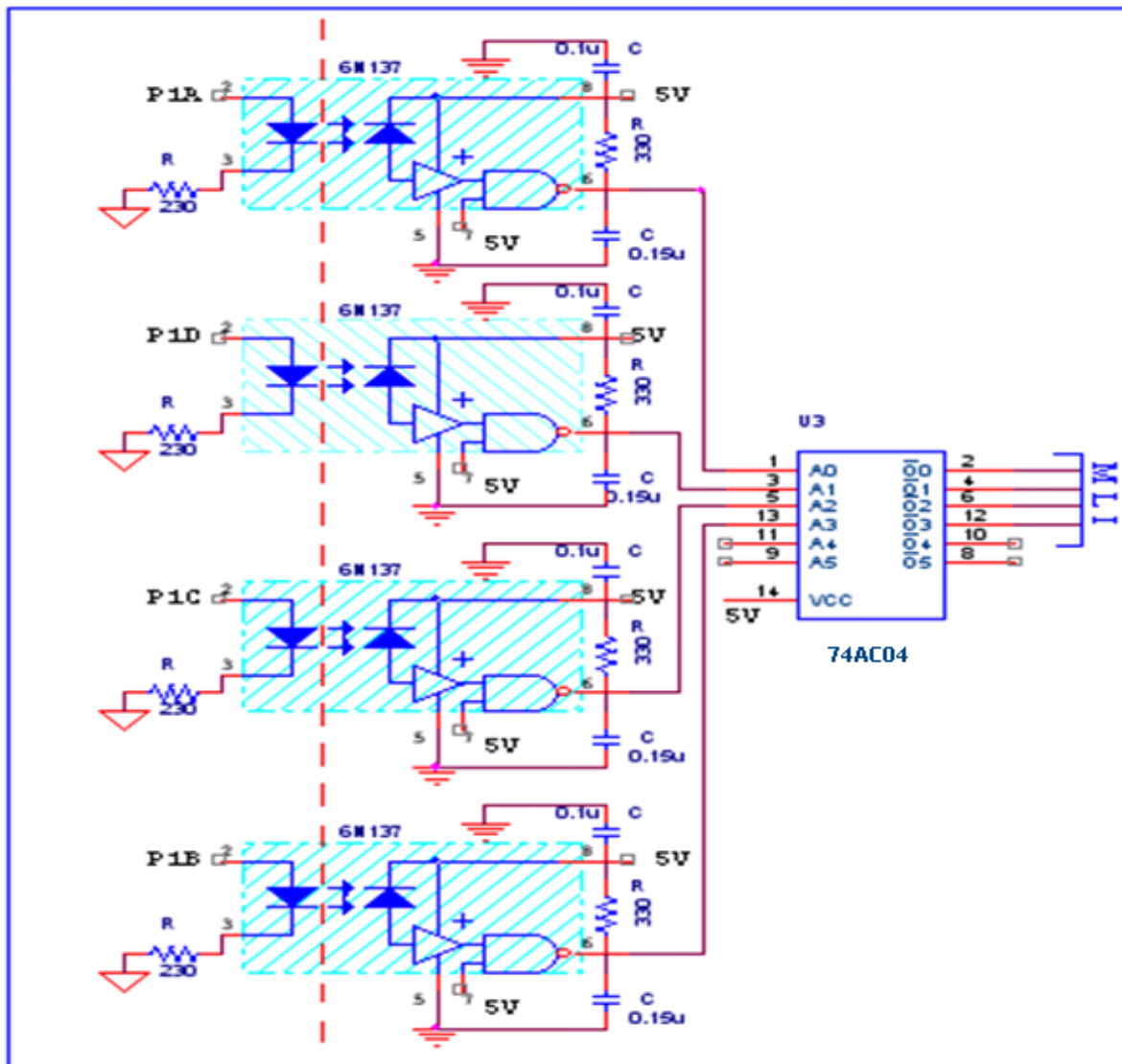


Figure III.4 : L'étage de protection

L'alimentation du microcontrôleur, le compteur quadrature, l'étage d'entrée des optocoupleurs, et l'encodeur optique vient du câble USB. Les autres composants sont alimentés par l'alimentation de la carte.

### II.3- La chaîne d'acquisition de vitesse du moteur :

Pour calculer la vitesse instantanée du moteur on a besoin d'un encodeur optique monté sur l'axe du moteur, d'un compteur quadrature et le microcontrôleur.

L'encodeur optique est fixé sur l'axe du moteur comme indiqué sur la figure III.5.

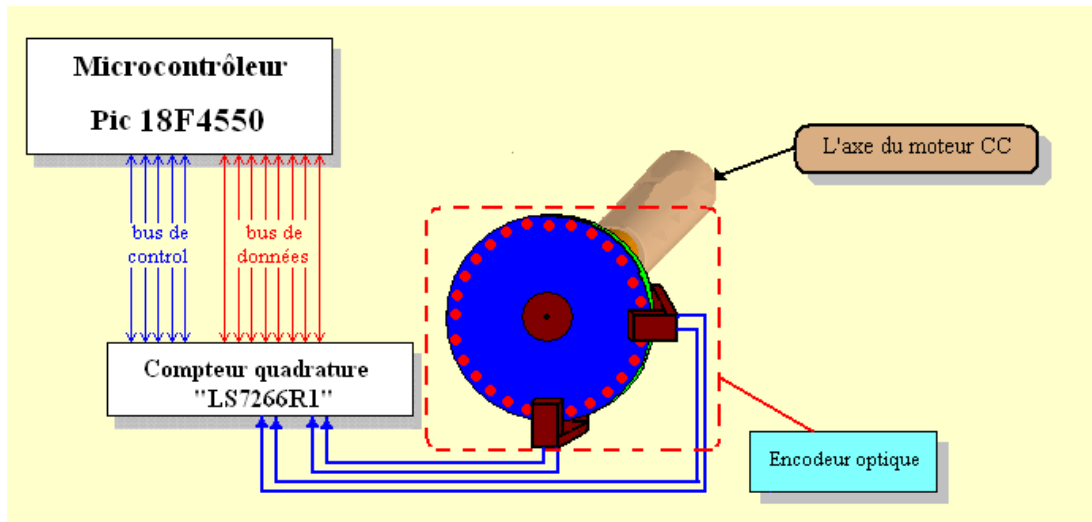


Figure III.5 : L'acquisition de la vitesse

### II.3.1- L'encodeur optique :

Un encodeur optique est un appareil qui permet de convertir un mouvement de rotation en signal électrique. Il comporte un disque (généralement en verre) solidaire d'un axe. Lorsque l'axe tourne, le disque tourne avec lui. Sur ce disque sont tracés deux séries de petits traits, décalées l'une par rapport à l'autre. Une source de lumière se trouve devant chaque série de traits et de l'autre côté du disque (en face de chaque source lumineuse) se trouve un récepteur de lumière. Ce récepteur convertit le signal lumineux reçu en signal électrique. Lorsqu'un trait passe entre la source et le récepteur, ce dernier ne reçoit plus de lumière, il génère un signal à l'état bas (0V). Si le disque tourne, le trait est remplacé, devant la source lumineuse, par une zone transparente. Le récepteur reçoit alors de la lumière et il génère un signal à l'état haut (+5V). Ceci est vrai pour les deux séries de traits. Lorsque l'axe tourne, le codeur optique génère donc deux séries d'impulsions (à 0V et à 5V) phase A et B décalées dans le temps l'une par rapport à l'autre, Figure III-7 [S11].

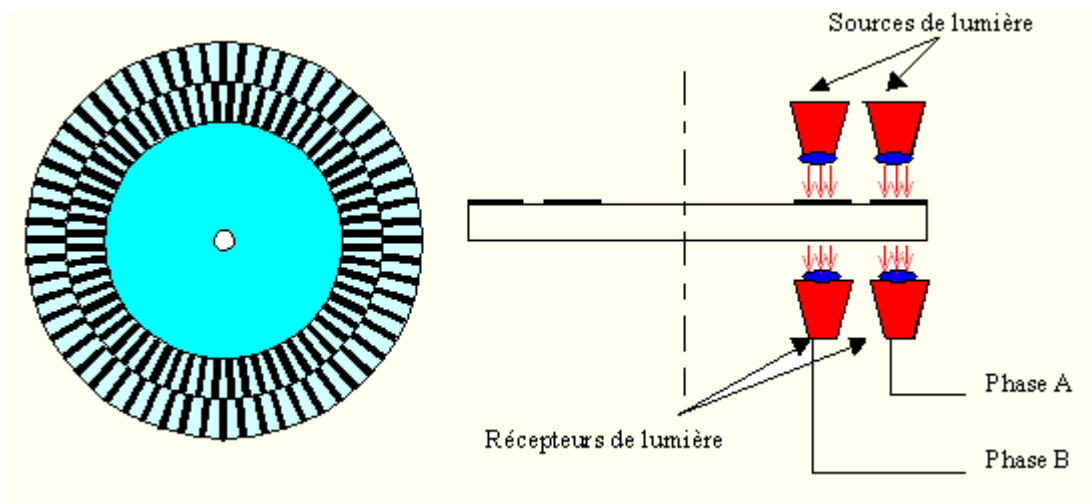
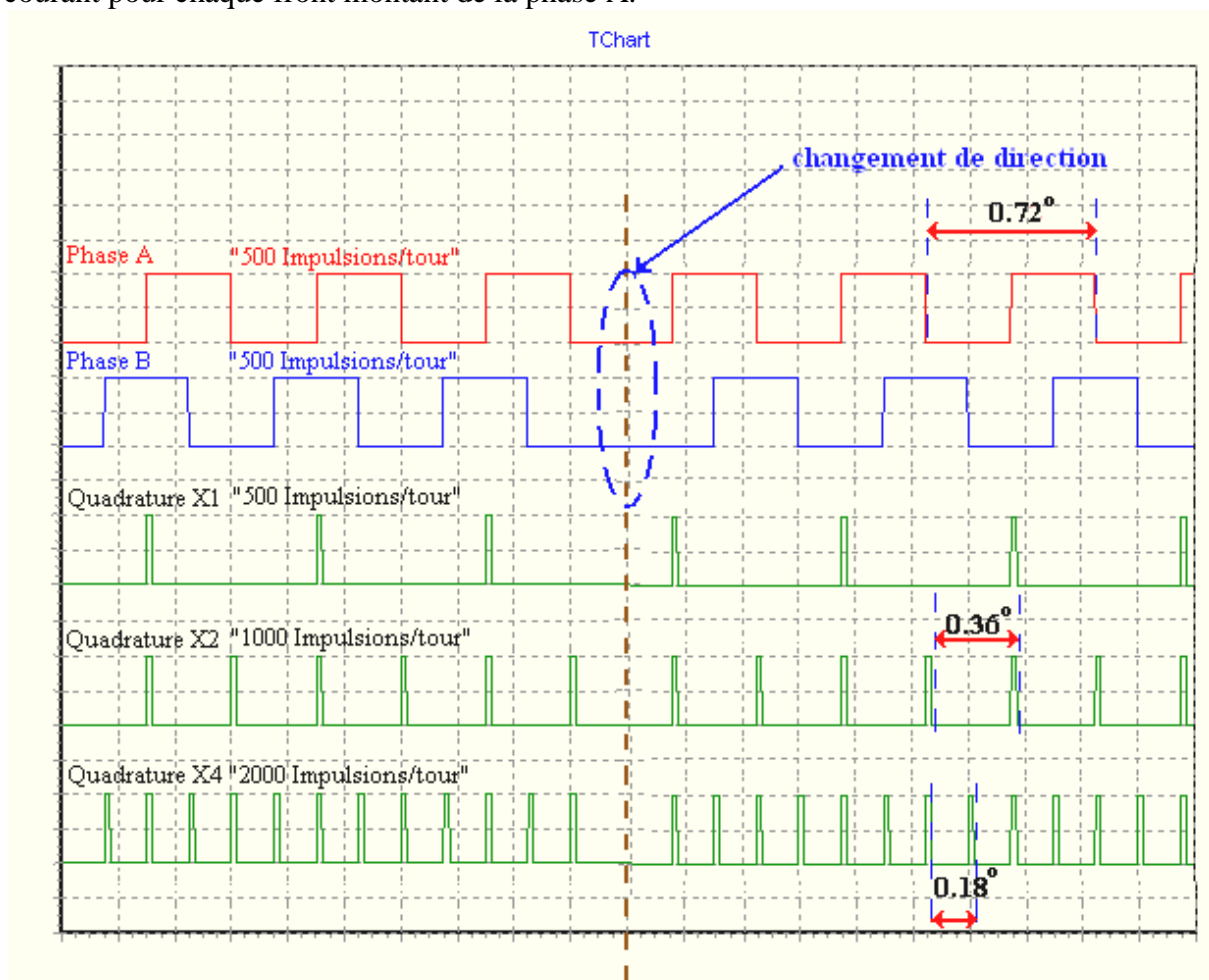


Figure III.6 : principe de fonctionnement d'un encodeur optique

Un encodeur optique a une caractéristique principale : c'est le nombre d'impulsions par tour. Ça représente le nombre de traits contenus dans chaque série. Plus ce nombre d'impulsions est élevé, plus le codeur optique est précis et cher. Ce nombre d'impulsions permet donc de savoir de quelle valeur d'angle tourne l'axe. En effet, pour 500 impulsions, chaque alternance "noir transparent" représente  $360^\circ / 500 = 0,72^\circ$ . Selon le sens de rotation on voit d'abord la phase A puis la phase B ou l'inverse. Ceci nous permet de dire dans quel sens tourne l'axe. Le chronogramme suivant montre comment on peut utiliser ces données.

Pour déterminer le sens de rotation il suffit de regarder, sur le front montant de la phase B, dans quel état se trouve le signal phase A, à 0V c'est un sens de rotation et à 5V c'est l'autre sens. Il faut ensuite ajouter ou soustraire, en fonction du sens de rotation,  $0,72^\circ$  à l'angle courant pour chaque front montant de la phase A.



**Figure III.7** : Les signaux visualisés par l'interface Delphi générés à partir d'un encodeur 500 impulsions / tour.

On peut augmenter la précision du codeur optique en utilisant les deux fronts (montant et descendant) des phases A et B. On a donc maintenant 2000 impulsions / tour et on multiplie par 4 la précision du codeur. Le chronogramme de la Figure III.7 indique comment faire.

On compte cette fois-ci le nombre d'impulsions régénérées et chaque impulsion indique une évolution de  $0,18^\circ$  de l'angle. La détection du sens de rotation reste la même qu'auparavant. C'est cette gestion du codeur qui est utilisée dans cette interface.

### II.3.2- le compteur quadrature

Pour calculer les impulsions générées par l'encodeur optique on utilise le compteur quadrature LS7266R1 qui a les caractéristiques suivantes :

- Deux compteurs 24-bit (comptage /décomptage).
- Multiplicateur de la résolution X1, X2 ou X4.
- 2 axes comparateurs de 24-Bit.
- Le choix du mode est indépendant pour chaque axe.
- 17 mégahertz en mode quadrature.
- 4 registres de control.
- Registre de statut lisible.
- Filtrage numérique des entrées en quadrature.
- Indicateurs d'erreur pour le bruit excessif.
- Tension de fonctionnement 5Volts.

Le LS7266R1 est un module utile dans les applications de contrôle de mouvement. Les deux compteurs 24-bit, les registres de control, et les différents modes de fonctionnement ont permis au microcontrôleur de connaître la direction, la position, et l'index de l'un ou des deux encodeurs incrémentaux optiques [23] .

#### Les registres de LS7266R1 :

Le LS7266 permet de :

- Récupérer les données du codeur incrémental.
- Transformer ces données en une donnée de position réelle (là où on est)(POSITION FEEDBACK PROCESSOR)

LS7266R1 a un ensemble de registres associé à chaque axe X et Y. Tous les registres de l'axe X ont le préfixe de nom X, de même ; tous les registres de l'axe Y ont le préfixe Y.

La sélection d'un registre spécifique pour la Lecture/Ecriture est faite par le décodage des trois bits les plus significatifs (D7-D5) du bus de données. L'entrée CS est l'entrée de validation du LS7266 pour la Lecture ou l'Ecriture.

XPR et YPR	: preset registers.
XCNTR et YCNTR	: les registres des compteurs.
XOL et YOL	: output latches registers.
XBP et YBP	: pointeurs des octets.
XFLAG et YFLAG	: registres de flag.
XPSC et YPSC	: filter clock prescalers.
XRLD et YRLD	: reset and load signal decoders.
XCMR et YCMR	: registres de mode de comptage.



XIOR et YIOR : registres de control des entrées /sorties.  
 XIDR et YIDR : registres de control des index.

Chacun de ces CNTRs est de 24 bits. Chaque CNTR peut être chargé par le contenu de son PR associé.

En effet, les OLs sont les sorties des CNTRs. Chaque OL est de 24 bits. Le contenu de chaque CNTR peut être chargé dans son OL, ensuite ce contenu transmis vers le microcontrôleur via le bus de données, octet par octet, dans une séquence de trois octets.

Les opérations d'écriture et de lecture sur le registre OL ou le registre PR se font toujours octet par octet. L'octet accédé est l'octet adressé par le pointeur des octets BPs.

À la fin de chaque cycle d'opération d'écriture ou de lecture sur les registres OL ou PR, le pointeur des octets (BP) associé est incrémenté automatiquement pour pointer à l'octet suivant [23].

D7	D6	D5	$\overline{C/D}$	$\overline{RD}$	$\overline{WR}$	$\overline{X/Y}$	$\overline{CS}$	FONCTION
X	X	X	X	X	X	X	1	Désactivez les deux axes pour la lecture/écriture.
X	X	X	0	1		0	0	Écrivez au segment d'octet de XPR adressé par XBP
X	X	X	0	1		1	0	Écrivez au segment d'octet de YPR adressé par XBP
0	0	0	1	1		0	0	Écrivez à XRLD
0	0	0	1	1		1	0	Écrivez à YRLD
1	0	0	1	1		X	0	Écrivez à XRLD et YRLD
0	0	1	1	1		0	0	Écrivez à XCMR
0	0	1	1	1		1	0	Écrivez à YCMR
1	0	1	1	1		X	0	Écrivez à XCMR et YCMR
0	1	0	1	1		0	0	Écrivez à XIOR
0	1	0	1	1		1	0	Écrivez à YIOR
1	1	0	1	1		X	0	Écrivez à XIOR et YIOR
0	1	1	1	1		0	0	Écrivez à XIDR
0	1	1	1	1		1	0	Écrivez à YIDR
1	1	1	1	1		X	0	Écrivez à XIDR et YIDR
X	X	X	0	0	1	0	0	Lisez le segment d'octet de XOL adressé par XBP
X	X	X	0	0	1	1	0	Lisez le segment d'octet de YOL adressé par YBP
X	X	X	1	0	1	0	0	Lisez XFLAG
X	X	X	1	0	1	1	0	Lisez YFLAG

Figure III.8 : les modes d'adressage des registres du LS7266

### III- L'interface utilisateur DELPHI :

Sur l'ordinateur, on peut utiliser une interface similaire aux interfaces utilisées pour les communication RS-232, parce que la CDC est une classe standard dans l'USB, et Microsoft a mis en application un driver qui supporte l'émulation de l'interface RS-232.

Ce driver CDC fournit la couche qui fait la liaison entre le matériel USB et le driver d'UART.

**Remarque :** Cette solution est impossible avec un dispositif USB low-speed parce qu'il n'a pas un endpoint de type bulk.

La vitesse de transfert des données avec cette méthode peut aller jusqu'à 640 Kbits/s = 80 Kbytes/s. Donc, c'est beaucoup plus rapide que le RS-232 où la vitesse maximale est de 20 Kbits/s.

Ceci permet à l'interface sur l'ordinateur de rester sans changement parce que de sa perspective, il voit toujours l'interface RS-232.

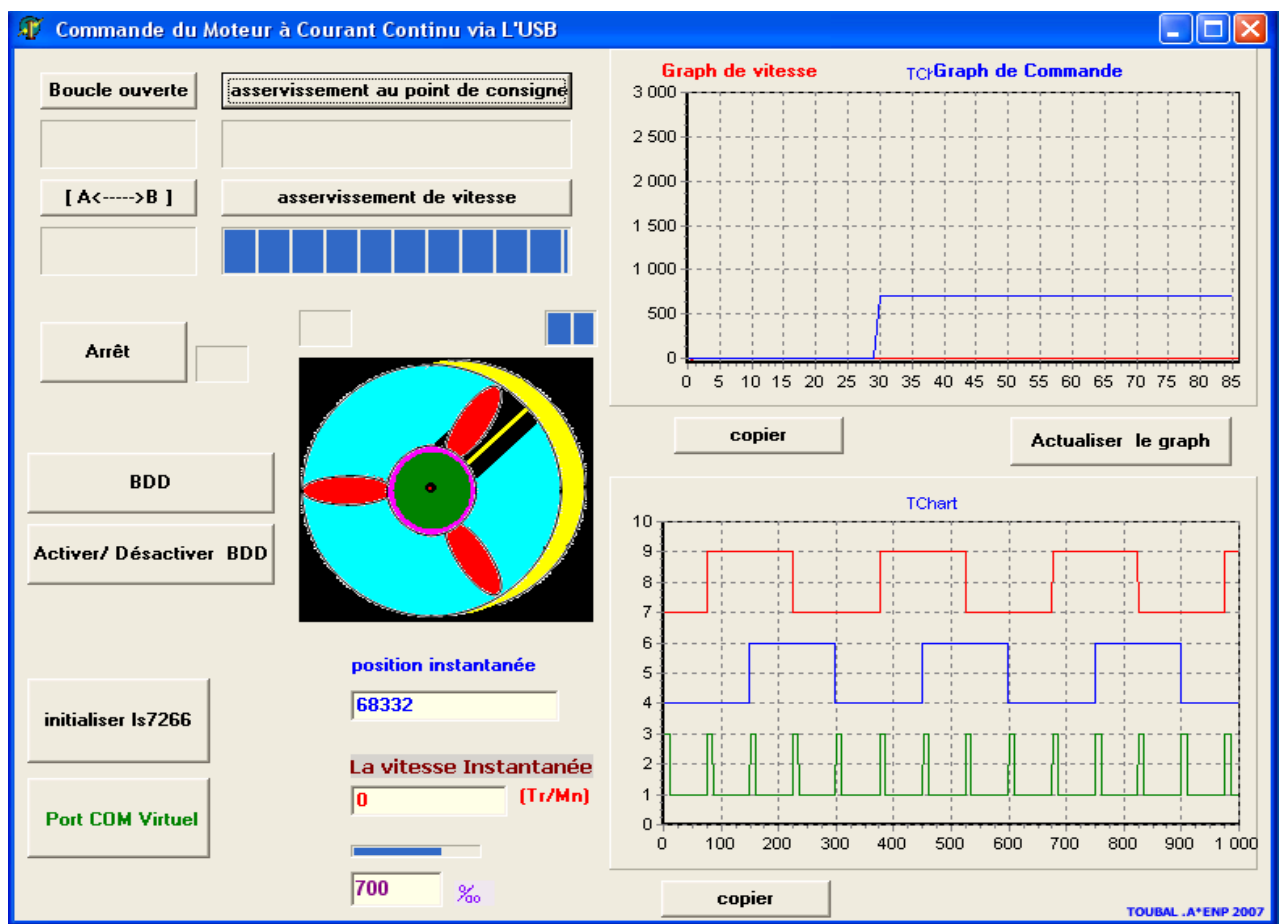


Figure III.9 : la fenêtre principale de l'interface utilisateur

Notre carte de commande à base de microcontrôleur 18F4550 a un périphérique USB full-speed. Nous avons utilisé le langage Delphi pour programmer l'interface utilisateur.

Pour accéder au port COM (virtuel), nous avons utilisé le composant Delphi TComPort. Ce composant peut également être utilisé avec toutes les versions de Delphi et de C++ Builder. C'est un composant très utile pour les programmeurs de Delphi et C. Nous utilisons ce composant pour envoyer et recevoir des vecteurs d'octets. La taille de chaque vecteur peut aller jusqu'à 64 octets.

☞ Les données envoyées vers la carte microcontrôleur sont :

- ✍ Les paramètres d'initialisation de LS7266 (mode de comptage, remise à zéro des compteurs).
- ✍ Marche/Arrêt du moteur.
- ✍ Le mode de fonctionnement sélectionné (Direct en boucle ouverte, asservissement de position, asservissement de vitesse, asservissement de déplacement entre deux angles...).
- ✍ Les valeurs de  $K_p$ ,  $K_i$ ,  $K_d$  (Les paramètres de régulateur PID).
- ✍ La vitesse consigne du moteur ou la vitesse max et le niveau d'accélération pour le profil de vitesse.
- ✍ Le sens de rotation du moteur pour le mode direct et le mode vitesse.
- ✍ le rapport cyclique qui doit être imposé par l'utilisateur en cas d'utilisation de mode Direct.

☞ Les données reçues à partir de la carte microcontrôleur sont :

- ✍ L'état du moteur (Marche /Arrêt)
- ✍ La vitesse instantanée du moteur.
- ✍ Le rapport cyclique appliqué par le microcontrôleur sur le hacheur.
- ✍ Le sens réel de rotation du moteur.
- ✍ Le temps exact entre deux échantillons (vitesse et position) envoyés.

Les données envoyées ou reçues sont encapsulées dans des vecteurs d'octets.

À partir de l'interface DELPHI l'utilisateur initialise les paramètres de comptage de compteur LS7266. Il peut remettre à zéro les compteurs de position de chaque canal et choisir le canal utilisé. En plus il peut sélectionner le mode de comptage désiré.

Il y a trois modes de comptage :

- 1- Quadrature X1 : le compteur s'incrémente par 1 pour chaque front montant de la phase A de l'encodeur.
- 2- Quadrature X2 : le compteur s'incrémente par 1 pour chaque front montant de la phase A et la phase B de l'encodeur.
- 3- Quadrature X4 : le compteur s'incrémente par 1 pour chaque front montant et chaque front descendant de la phase A et la phase B de l'encodeur.

Comme la carte peut fonctionner selon 4 modes ; l'interface utilisateur doit répondre à cette dernière, pour cela des fenêtres vont être ouvertes pour chaque mode quand on clic sur le bouton correspondant.

## III- 2.Les modes utilisés :

### III- 2.1.Le mode Direct (en boucle ouverte) :

Dans ce mode, l'utilisateur est chargé d'imposer la valeur de MLI qui est entre 0 et 1000 %, et le sens désiré de rotation du moteur.

L'interface va afficher la vitesse et la position instantanées.



Figure III.10 : Le mode Direct

### III- 2.2.Le mode d'asservissement de vitesse :

Les données à entrer, dans ce mode, par l'utilisateur sont : la vitesse consigne, les paramètres de correcteur PID ( $K_p$ ,  $K_i$ ,  $K_d$ ) et le sens de rotation.

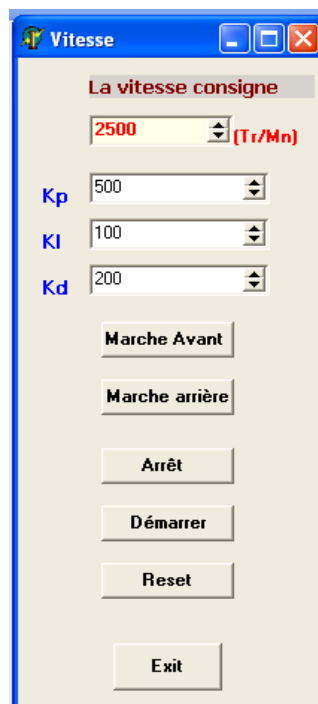


Figure III.11 : Le mode Vitesse

### III- 2.3.Le mode asservissement de position :

Dans ce mode l'utilisateur est chargé d'entrer la position consigne et les paramètres de correcteur PID ( Kp , Ki , Kd) .

Si l'utilisateur choisit d'activer le profil de vitesse, il est nécessaire d'entrer la valeur de vitesse max et le niveau d'accélération pour que le graphe de vitesse du moteur prend la forme d'un trapèze ou d'un triangle.

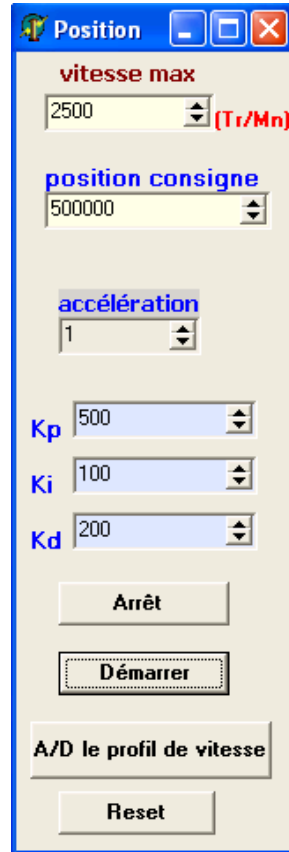


Figure III.12 : Le mode Position

Si le mode de comptage utilisé est Quadrature X4 et l'encodeur optique est de 500 impulsions par tour, donc chaque 2000 impulsions représente un tour.

La précision dans ce cas est égale à  $360^\circ/2000 = 0,18^\circ$ . Donc chaque impulsion représente  $0,18^\circ$  de rotation de l'axe moteur.

Pour faire tourner le moteur de N tours en avant, il suffit d'ajouter ( $N*2000$ ) à La position actuelle. Et pour faire tourner le moteur de N tours en arrière, il suffit de soustraire ( $N*2000$ ) de la position actuelle.

#### Exemple :

Position actuelle 1200000.

Pour 2250.78 tours en avant, la position consigne =  $1200000 + (2250.78 * 2000) = 5701560$ .

Pour 2250.78 tour en arrière, la position consigne =  $1200000 - (2250.78 * 2000) = -3301560$ .

### III- 2.4. Le mode asservissement point à point :

Ce mode nous a permet de commander le déplacement entre deux position (deux angles) avec ou sans profil de vitesse.

L'utilisateur est chargé d'entrer les deux angles des positions consignes, les paramètres du correcteur PID (  $K_p$  ,  $K_i$  ,  $K_d$  ) , la valeur de vitesse max et le niveau d'accélération en cas d'utilisation de profil de vitesse .

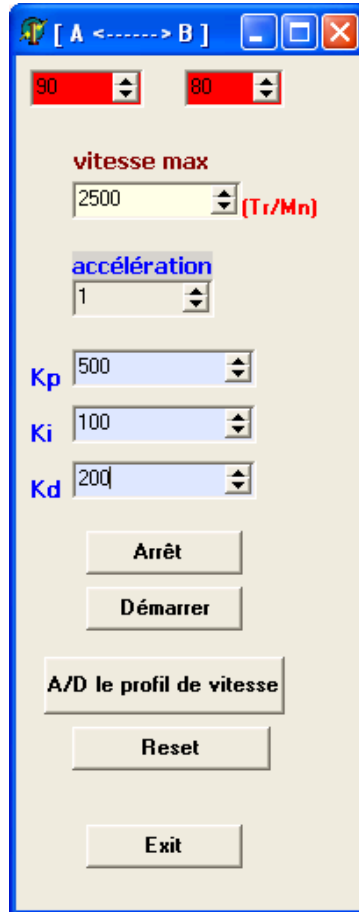


Figure III.13 : Le mode asservissement point à point

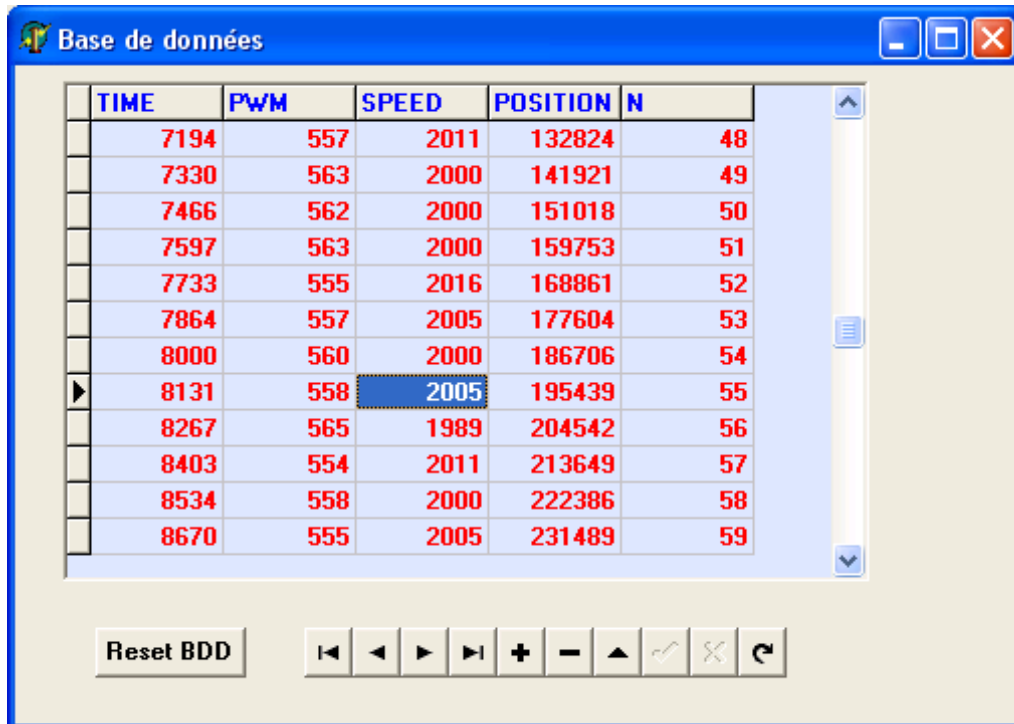
### III- 3. la base de données :

Une petite base de données a été implémentée afin d'enregistrer toutes les données reçus pendant le mouvement ; Un fichier d'extension (.DBF) a été créé.

Les données structurées dans la base de données sont :

- Le temps(en ms) quand l'échantillon a été envoyé.
- La vitesse instantanée du moteur (Tr/mn).
- La commande instantanée (la valeur de MLI %).
- La position instantanée.
- Le numéro d'échantillon.

On peut visualiser les données enregistrées par l'interface Delphi, Figure III-14, ou bien par l'utilisation de Microsoft Excel, ce dernier nous permet de tracer le graphe de chaque type de données (graphe de vitesse, de position ou de commande) (Figure III-15).



TIME	PWM	SPEED	POSITION	N
7194	557	2011	132824	48
7330	563	2000	141921	49
7466	562	2000	151018	50
7597	563	2000	159753	51
7733	555	2016	168861	52
7864	557	2005	177604	53
8000	560	2000	186706	54
8131	558	2005	195439	55
8267	565	1989	204542	56
8403	554	2011	213649	57
8534	558	2000	222386	58
8670	555	2005	231489	59

Figure III.14 : La base de données visualisée par Delphi

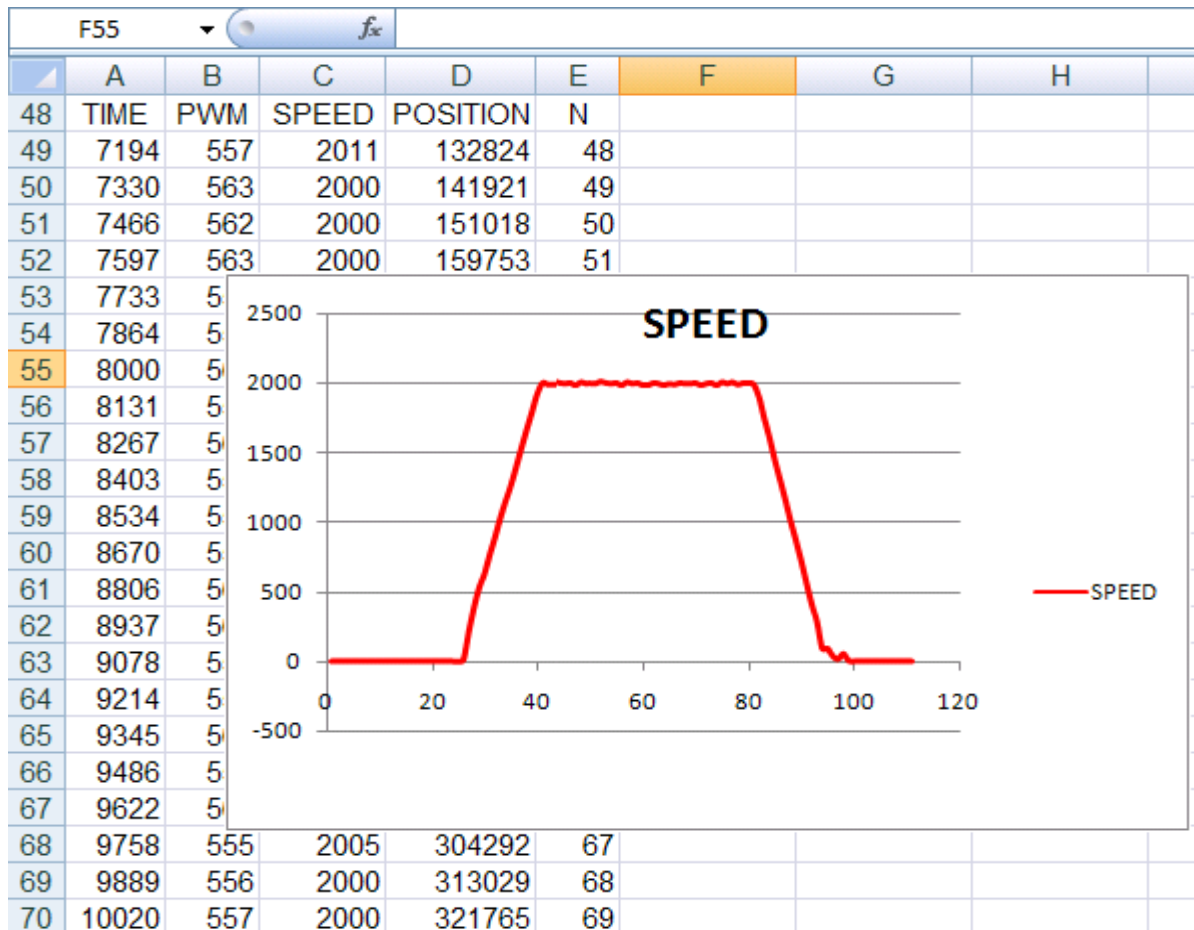


Figure III.15 : La base de données visualisée par Microsoft Excel

## IV . Implémentation de la commande :

### IV.1- La commande du hacheur :

#### IV.1.1 La modulation en largeur d'impulsion (MLI) :

La MLI ou PWM (Pulse Wide Modulation) est utilisée pour varier le rapport cyclique. En somme, il s'agit d'un signal binaire (un signal qui peut prendre 2 états) de fréquence fixe dont le rapport cyclique peut être modulé par logiciel.

Le rapport cyclique d'un signal binaire à fréquence fixe peut être défini comme étant le rapport entre le temps où il se trouve à l'état « 1 » par rapport au temps total d'un cycle.

Un cycle n'étant constitué, par définition, que d'un état « 1 » suivi d'un état « 0 », la somme des temps des 2 états étant constante.

Notons donc qu'il y a 2 paramètres qui définissent un signal « MLI » :

- \* La durée d'un cycle complet ou, par déduction, sa fréquence de répétition ou sa période.
- \* Le rapport cyclique

Donc, si on pose :

- $T_{on}$  = durée de l'état haut



- $T_{off}$  = durée de l'état bas
- $T_c$  = Durée d'un cycle =  $T_{on} + T_{off}$ .
- $\tau$  = le rapport cyclique =  $T_{on} / T_c$  [12].

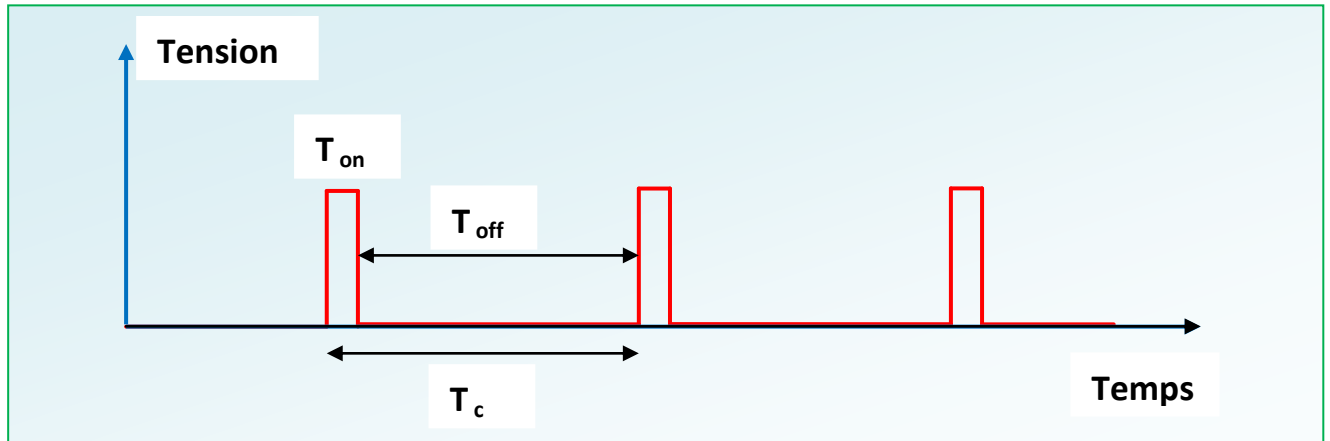


Figure III.16 : Exemple d'un signal PWM avec un rapport cyclique de 10%

#### IV.1.2 - Génération des signaux MLI avec le PIC18F4550 :

Le PIC 18F4550 a un mode MLI amélioré (Enhanced PWM Mode). Le mode fournit des options supplémentaires de sortie MLI pour un plus large intervalle des applications de commande. Le module est une version compatible du module standard de CCP et offre jusqu'à quatre sorties, P1A, P1B, P1C, P1D. L'utilisateur peut également choisir la polarité du signal (actif-haut ou actif-bas) et le mode de MLI. On peut configurer le mode et la polarité de sortie du module en plaçant le P1M1 :P1M0 et les bits CCP1M3 :CCP1M0 du registre CCP1CON.

##### Période de la MLI :

La période de la MLI est fixée par écrit au registre PR2. La période de la MLI peut être calculée en utilisant l'équation suivante :

$$\text{Période de MLI} = (PR2 + 1) * 4 * T_{osc} * TMR2 \text{ prédiviseur} \quad [11].$$

##### $T_{on}$ de la MLI :

Le  $T_{on}$  de MLI est fixé par écrit au registre CCPR1L et aux bits <5:4 > de CCP1CON. Une résolution jusqu'à 10-bit est disponible.

Le CCPR1L contient les huit MSBs et les bits <5:4 > de CCP1CON sont les deux LSbs. Cette valeur 10-bit est représentée par CCPR1L : CCP1CON <5:4 >.

Le  $T_{on}$  de la MLI est calculé par l'équation suivante.

$$T_{on} = CCPR1L:CCP1CON<5:4> \cdot TOSC \cdot (TMR2 \text{ prédiviseur}).$$

La procédure exacte est donc selon la figure III.17 la suivante :

- Le Timer 2 compte : on imagine que le signal CCP vaut actuellement « 0 »
- TMR2 arrive à la valeur de PR2
- Au cycle suivant, TMR2 repasse à « 0 », CCP1 passe à « 1 »
- En même temps, la valeur programmée comme consigne (CCPR1L :CCP1CON<5:4>) est copiée dans le registre final de comparaison (CCPR1H).
- TMR2 arrive à la valeur de la copie de la seconde valeur de consigne (CCPR1L :CCP1CON<5:4>), CCP1 passe à « 0 »
- TMR2 arrive à la valeur de PR2
- Au cycle suivant, TMR2 = 0, CCP1 vaut « 1 », et ainsi de suite [11].

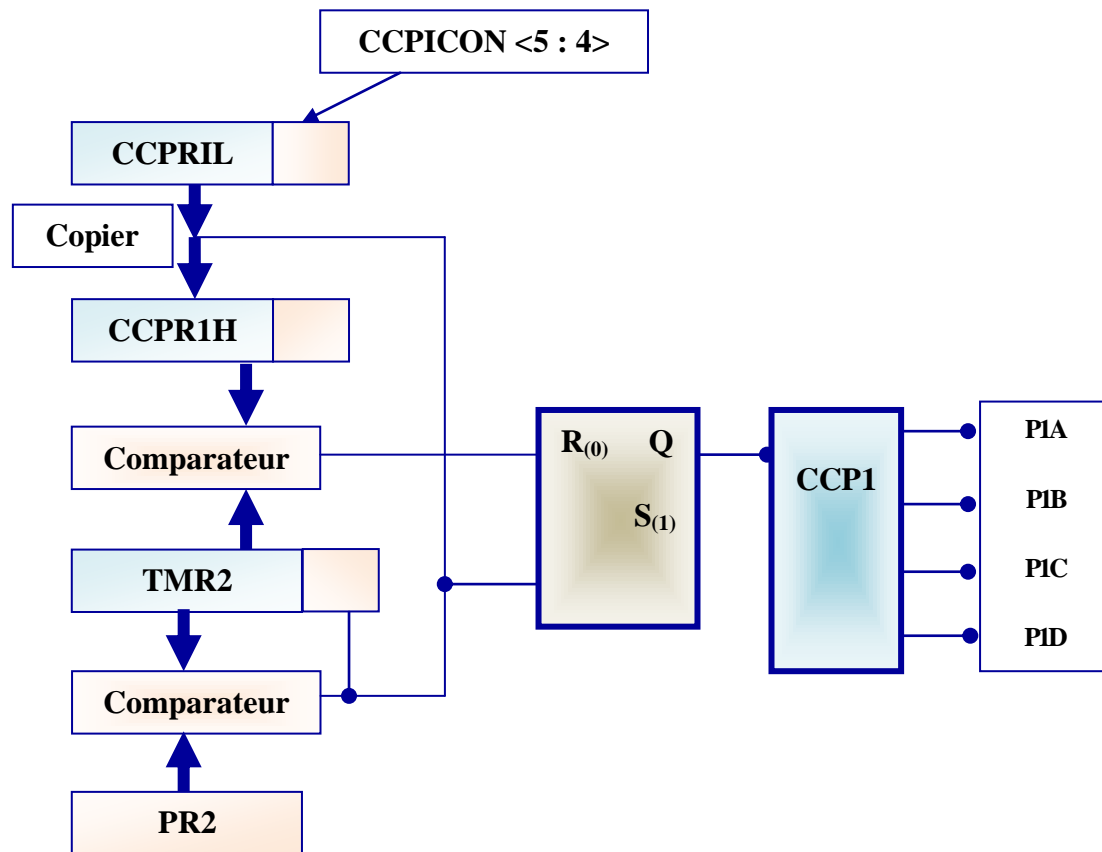


Figure III.17 : Schéma bloc simplifié du module «MLI »

#### IV.1.3- Configurations des sorties MLI du module CCP :

Les bits <7:6>du registre CCP1CON permettent l'une des quatre configurations possibles :

- Sortie Simple

- Sortie Half-Bridge
- Sortie Full-Bridge, mode Forward.
- Sortie Full-Bridge, mode Reverse.

Dans notre cas on utilise les deux derniers (Full-Bridge mode Forward et Full-Bridge mode Reverse) [5].

### Le mode Full-Bridge :

Dans le mode de Full-Bridge, quatre pins sont utilisés comme sorties ; cependant, seulement deux sorties sont activées à la fois.

- En mode Forward, le pin P1A reste en activité et le pin P1D est modulé.
- En mode Reverse, le pin P1C est reste en activité et le pin P1B est modulé. Ceux-ci sont illustrés sur le schéma de la figure III.18 a et b.

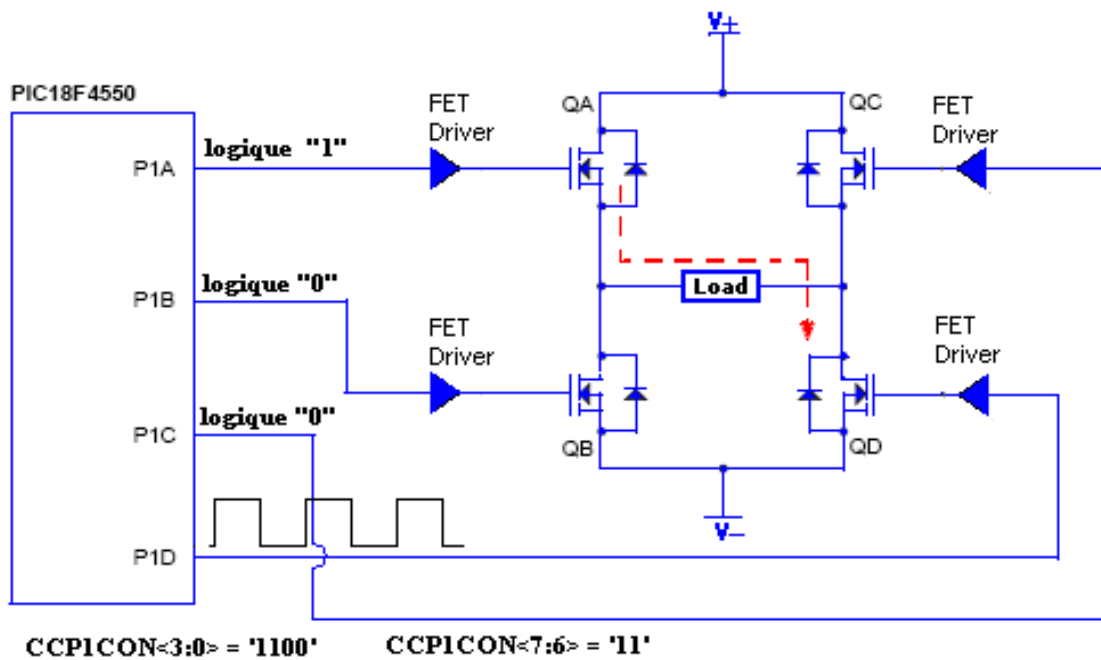


Figure III.18.a : Le Full-bridge FORWARD

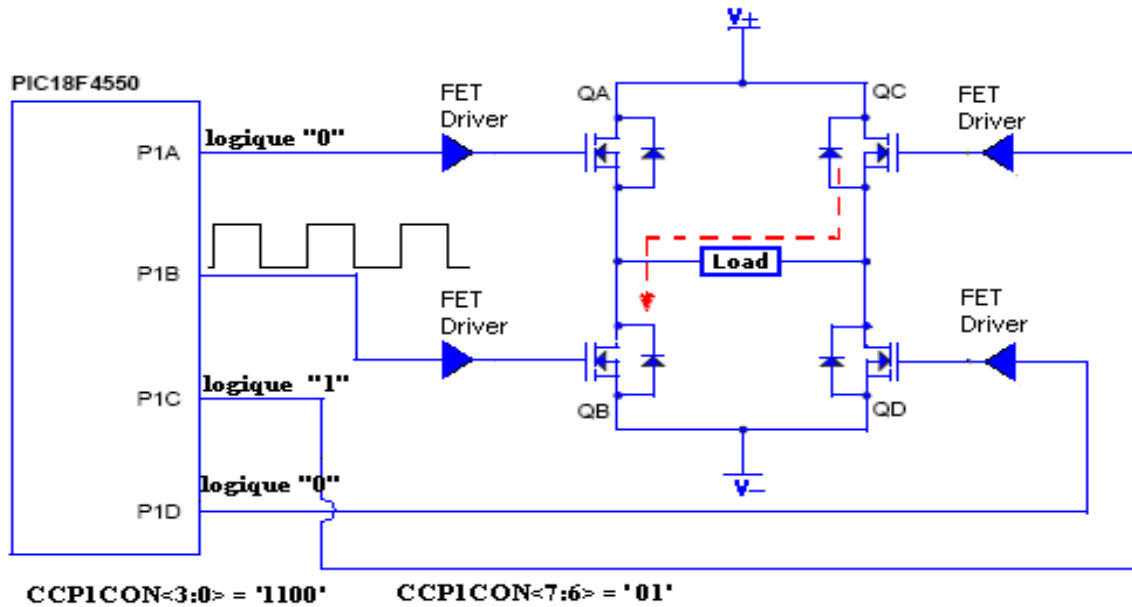


Figure III.18.b : Le Full-bridge REVERSE

### Le changement de direction de mode Full-Bridge

En mode de sortie Full-Bridge, le bit P1M1 dans le registre de CCP1CON permet à l'utilisateur de contrôler la direction de forward/reverse. Quand le code d'application change ce bit de contrôle de direction, le module assumera la nouvelle direction sur le prochain cycle de la MLI.

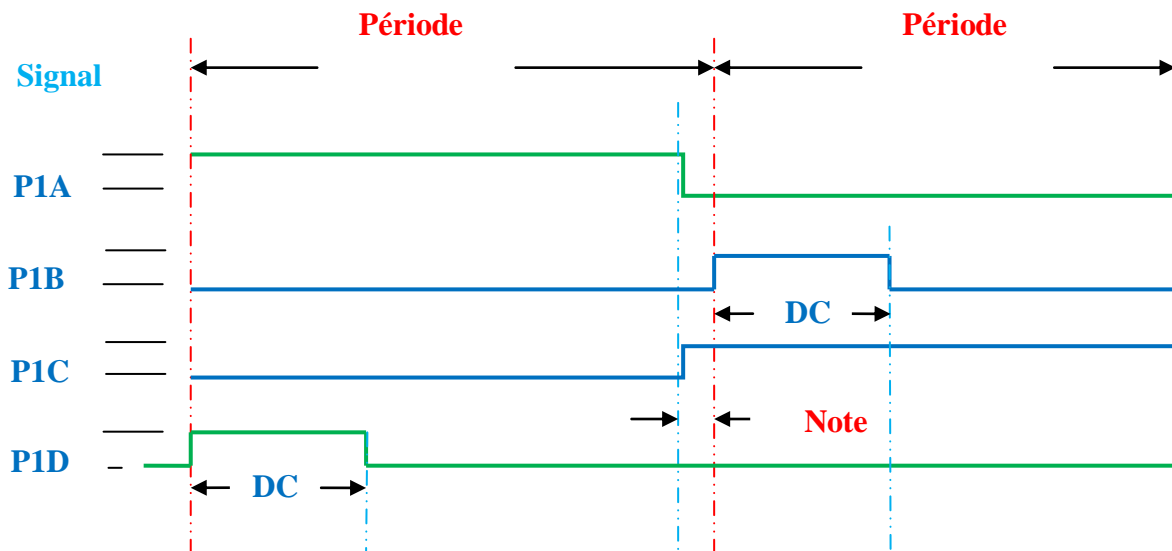


Figure III.19 : Le changement de direction de mode Full-Bridge [5].

**Note :** Juste avant la fin de la période actuelle de la MLI, les sorties modulées (P1B et P1D) sont mises dans leur état inactif, alors que les sorties non modulées (P1A et P1C) sont

activés pour piloter dans le sens opposé. Ceci se produit dans un intervalle de temps de (4 TOSC \* (Timer2 Prescale Value)) avant le début de la période suivante de la MLI.

## IV.2. Les routines d'acquisition de la position et la vitesse du moteur :

L'encodeur optique envoie ses impulsions vers le compteur quadrature (LS7266) qui est piloté par le microcontrôleur 18F4550.

Une routine d'interruption (basée sur le TMR0) s'exécute toutes les 5.46 ms, elle génère des signaux de commande vers le LS7266 pour charger le contenu des deux compteurs XCNTR et YCNTR dans les registres XOL et YOL respectivement puis met à 1 la variable « update » de type char pour indiquer que le chargement de CNTR dans OL est fait.

Cette solution a pour but d'éviter d'interrompre la communication USB qui ne supporte que de brèves routines d'interruption.

Quand « update » est mis à 1, une petite routine sert à remettre « update » à 0 et terminer le processus d'acquisition de position par la lecture de la valeur de compteur de l'axe (l'axe X ou l'axe Y) choisie par l'utilisateur via l'interface Delphi.

La lecture des 24 bits de registre OL de compteur accède toujours à un octet à la fois, c'est l'octet adressé par le pointeur BP. À la fin de chaque cycle de lecture des données d'OL, le BP associé est automatiquement incrémenté pour adresser le prochain octet.

La soustraction des deux valeurs successives de position nous donne la valeur de la vitesse du moteur (après certains calculs selon le nombre d'impulsions par tour de l'encodeur qui est reçu via l'USB à partir de l'interface utilisateur), on stock cette valeur dans la variable speed qui est de 32bits. .

La taille de chaque registre (XCNTR ou YCNTR) des compteurs est de 24bits (16777215) qui nous permet d'aller jusqu'à 8388 tours (pour 2000 impulsions/tour) avant que le compteur déborde.

On a choisi le point 8400000 comme le zéro de notre calcul de position, donc quand l'utilisateur remet les compteurs à zéro, en effet ; on charge la valeur 8 400000 dans les compteurs, mais pour l'utilisateur il voit toujours la valeur zéro.

Cette solution nous permet d'imposer des valeurs de positions négatives ou positives. Pour le pilotage de LS7266, on a utilisé le pilote LS7266.h qui est réalisé pour les microcontrôleurs Atmel , mais avec quelques changements nécessaires pour adapter leurs fonctions avec le PIC18f4550 et le hard de notre application.

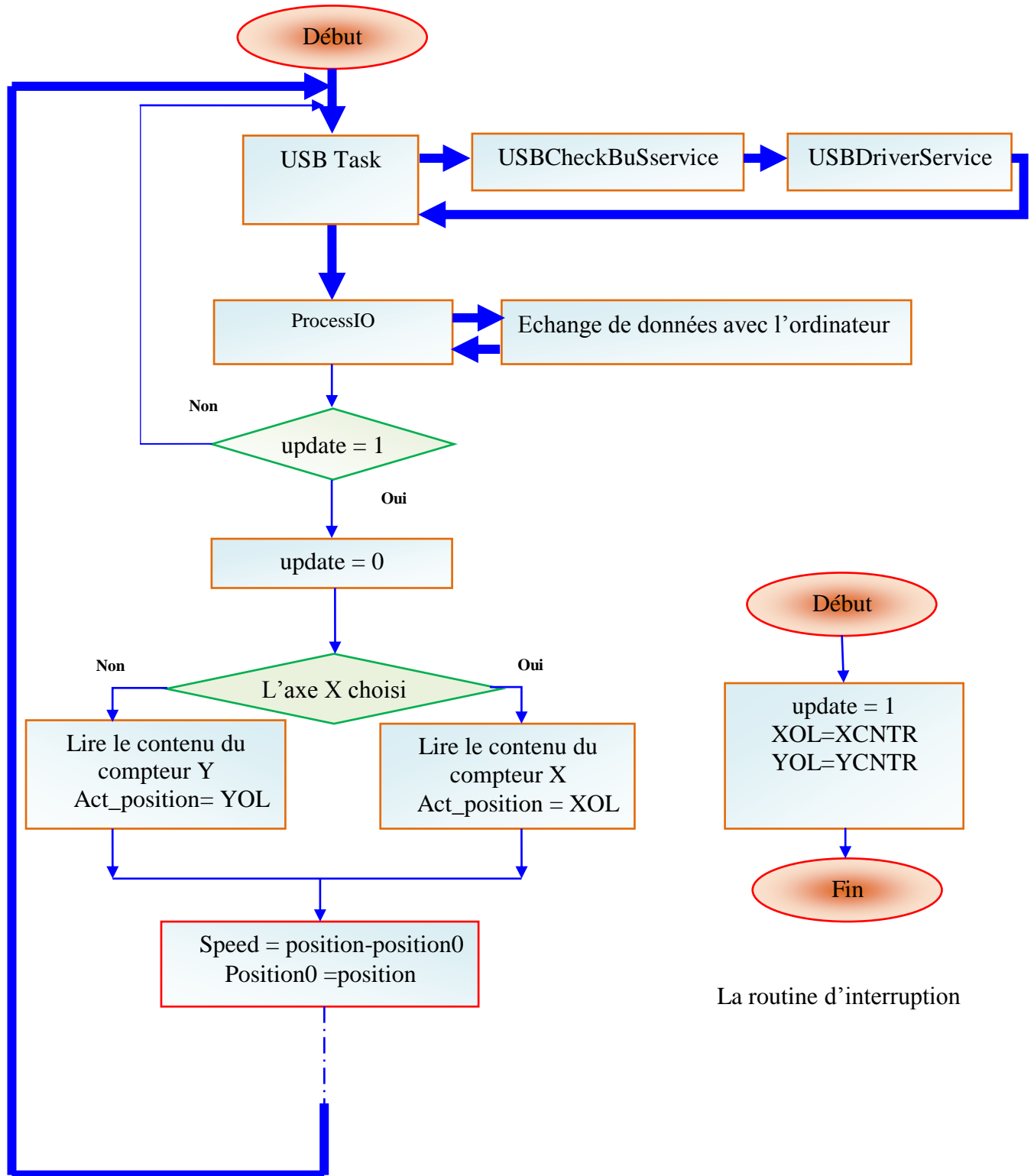


Figure III.20 : Organigramme de l'acquisition de vitesse et la tâche USB

### IV.3. La boucle de régulateur PID

Le but de l'asservissement est d'être capable de contrôler avec précision un moteur ; or le moteur à courant continu a une vitesse qui varie facilement, il faut d'une part être capable de la mesurer et d'autre part, pouvoir faire varier la tension du moteur pour qu'il tourne correctement. Pour cela il existe plusieurs algorithmes. Le plus répandu est la boucle de régulation PID [19]. C'est celui que nous allons l'implémenter ici.

Les trois composantes du PID sont :

☞ **'P'** proportionnelle, consiste à dire que plus l'erreur est grande et positive et plus il faut augmenter la tension aux bornes du moteur (pour rattraper cette erreur). La tension aux bornes du moteur est alors directement proportionnelle à l'erreur :

$$\text{Sortie}_{(P)} = kP * (\text{Erreur}).$$

$kP$  étant le coefficient de proportionnalité de la composante proportionnelle.

☞ **'I'** intégrale, vient s'ajouter à la première composante en évitant cette fois l'accumulation de l'erreur au cours du temps :

$$\text{Sortie}_{(I)} = kI * (\text{la Somme Des Erreurs}).$$

$kI$  étant le coefficient de proportionnalité de la composante intégrale.

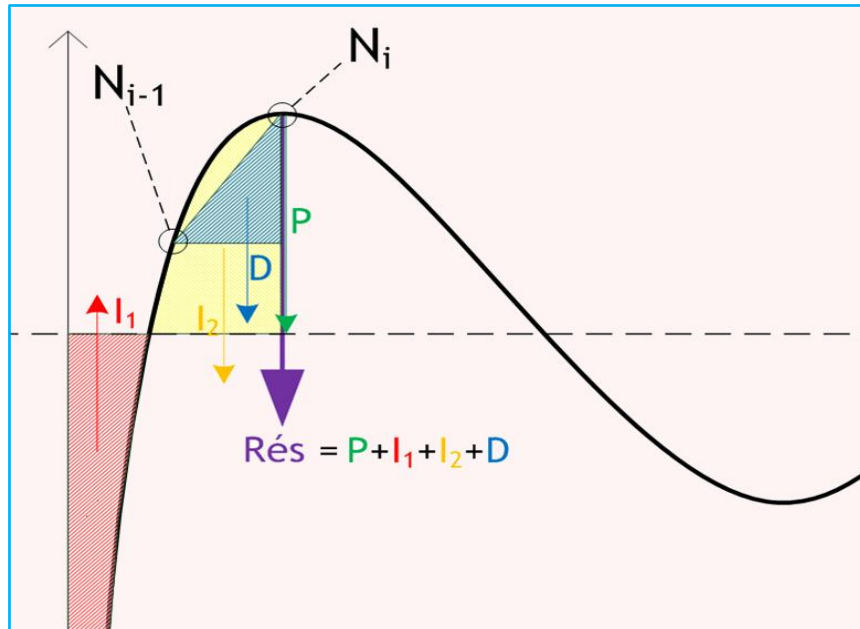
☞ **'D'** dérivée, permet d'anticiper et d'amortir le mouvement, en prenant en compte la variation de l'erreur au cours du temps. Lorsqu'on met le moteur sous tension, il va mettre un certain temps pour réagir (constante de temps mécanique). Pendant ce temps, l'erreur va continuer à augmenter ce qui implique une augmentation excessive de la tension. Pour diminuer ce phénomène "d'emballement", on va introduire un troisième élément qui permettra d'en réduire les effets. On se base sur la différence (la variation) de l'erreur entre deux mesures. Il s'agit de la dérivée de l'erreur.

$$\text{Sortie}_{(D)} = kD * (\text{Erreur} - \text{Erreur}_0).$$

$kD$  étant le coefficient de proportionnalité de la composante dérivée.

Enfin; On obtient :

$$\text{Sortie}_{(PID)} = kP * \text{Erreur} + kI * \text{SommeDesErreurs} + kD * (\text{Erreur} - \text{Erreur}_0).$$



**Figure III.21** : les composants d'un PID

L'entrée de l'algorithme est l'erreur calculée ( $u_0$ ), la sortie de l'algorithme est la valeur de MLI qui doit être appliquée à l'hacheur.

Cette routine de PID est utilisée dans le mode d'asservissement de vitesse et le mode d'asservissement de position.

La routine de calcul du PID peut être utilisée pour l'asservissement de vitesse ou de position, selon le mode de fonctionnement sélectionné.

Le terme proportionnel de l'algorithme PID fournit un résultat qui est en fonction de l'erreur immédiate  $u_0$  (erreur de position ou de vitesse),

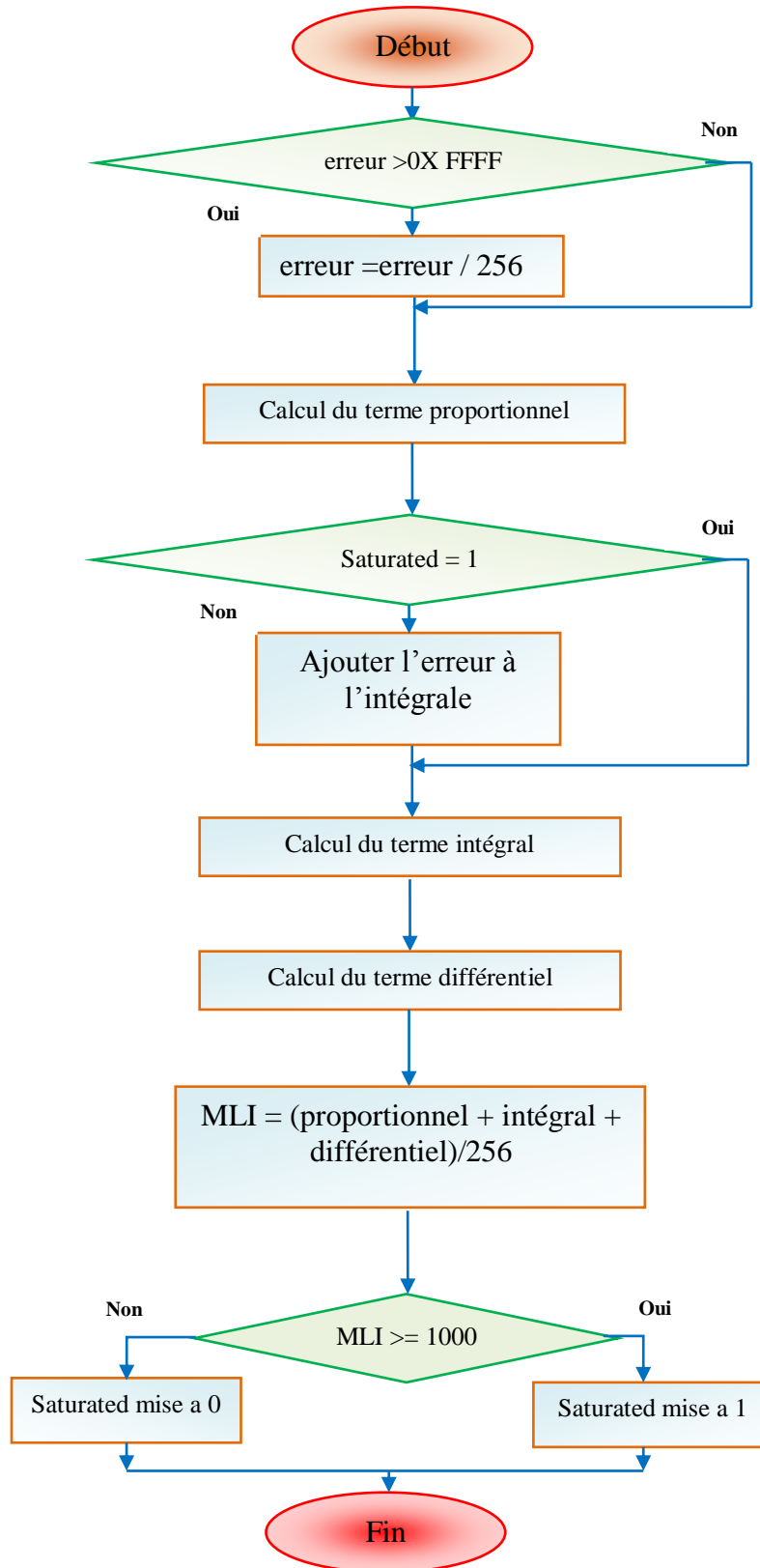
Le terme intégral de l'algorithme accumule les erreurs successives calculées pendant chaque itération de la boucle d'acquisition et améliore le gain en boucle ouverte de basse fréquence du système.

Si la sortie MLI est saturée une variable appelée « saturated » est mise à 1 sinon cette variable est mise à zéro. Si « saturated » est égale à 1 parce que la sortie de MLI pendant la période précédente est saturée l'erreur actuelle de position ne doit pas être ajoutée à la valeur intégrale. Ceci évite une condition connue sous le nom de 'integrator-windup' qui se produit quand le terme intégrale continue à accumuler l'erreur quand la sortie MLI est saturée [22].

Le terme différentiel de l'algorithme PID est une fonction de la différence entre l'erreur actuelle et l'erreur précédente.

Après que les trois termes de l'algorithme PID soient additionnés, le résultat obtenu de 32 bits sera décalé de 8 bits à droite, puis on enregistre les 16 bits de poids faible dans la variable YPID et on teste si la valeur de YPID est supérieure à 1000 donc la variable « saturated » est mise à 1.



**Figure III.22 :** l'organigramme de La boucle de PID

- $k_P$  et  $k_I$  améliorent le temps de réponse mais rendent le système moins stable.
- $k_I$  permet en plus d'éliminer l'erreur statique, mais il crée beaucoup d'oscillations.
- $k_D$  ralentit la réponse, mais permet d'atténuer les oscillations et donc rend le système plus stable.

Pour avoir un bon système, il faut donc trouver un compromis entre la rapidité, la stabilité et l'erreur statique. Si on veut gagner du temps, il vaut mieux être capable de pouvoir régler les trois coefficients de l'asservissement à la volée, sans avoir à recompiler le programme ou reprogrammer le  $\mu C$ , pour cela on envoie les trois coefficients  $K_p$ ,  $K_I$  et  $K_D$  via l'USB à partir de l'interface utilisateur [20].

#### IV.4. Le profil de vitesse :

Pour le contrôle optimum du mouvement, on doit appliquer une méthode qui permet de contrôler l'accélération et la décélération du moteur.

Le mouvement sans profil de vitesse sera brusque, causant l'usure excessive des composants mécaniques et dégradant la performance de l'algorithme de compensation.

Pour cette application, un profil simple de vitesse qui génère la trajectoire de vitesse trapézoïdale ou triangulaire a été mis en application.

Cette trajectoire est définie par trois phases successives :

- **accélération constante** ( $a = A$ ), la vitesse augmente linéairement
- **accélération nulle** ( $a = 0$ ), la vitesse est constante et maximale
- **décélération constante** ( $a = -A$ ), la vitesse diminue linéairement

Ce profil peut être utilisé dans le mode vitesse ou position. Le mode vitesse change très peu de choses, c'est à dire que dans le cas du mode vitesse, la phase de décélération n'existe pas. C'est là la seule différence entre le mode position et le mode vitesse.

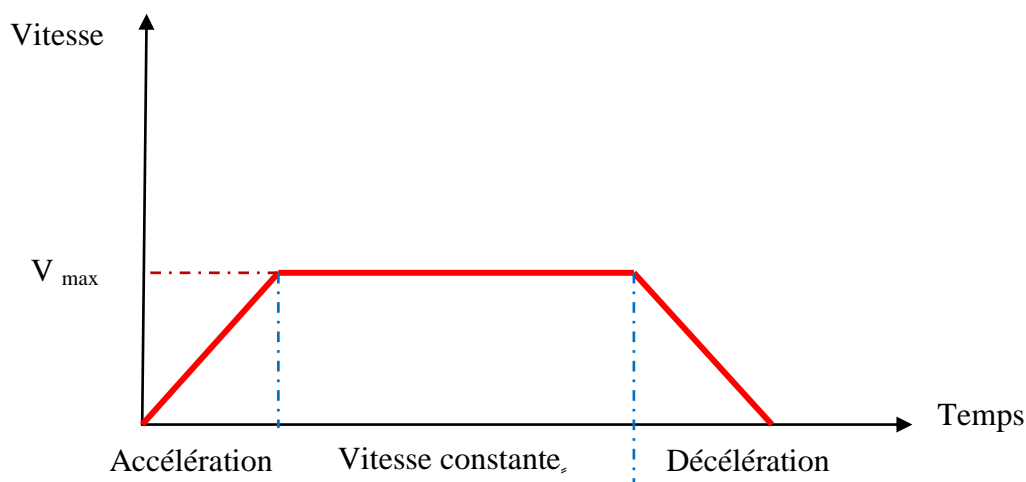


Figure III.23 : graphe de variation de la vitesse

La consigne de vitesse de moteur (vitesse de référence) est variée de 0 à  $V_{\max}$  en ajoutant une valeur d'offset à la vitesse commandée dans chaque période d'actualisation de PID jusqu'à ce que la vitesse commandée est égale à  $V_{\max}$ .

L'utilisation du profil de vitesse avec le mode position se fait comme suit :

- on calcule la distance totale à parcourir selon la position actuelle et la position consigne.
- on divise cette distance sur deux et on la stocke dans la variable "mid\_distance" qui est de 32bit.
- on met la vitesse commandée (com\_speed) à zéro.
- on commence la phase d'accélération incrémentant la valeur de "com\_speed" avec l'appel de boucle de PID de vitesse chaque période jusque à "com\_speed= speed\_max" .
- Quand "com\_speed= speed\_max" on calcule la distance parcourue et on le stock dans la variable distance qui est de 32 bit. Dans ce point la, on arrête l'incréméntation de "com\_speed".
- la variable "com\_speed" reste constante jusqu'à ce que l'erreur de position est inférieure à "distance" ou la position instantanée supérieure à "mid\_distance" qui veut dire que le mouvement est dans la deuxième moitié de déplacement totale, dans ce cas la, on entame la phase de décélération.
- on décréménte la valeur de "com\_speed" jusqu'à ce que la pente de l'accélération est négative et "com\_speed < 60Tr/s", En suite, on exécute la boucle de PID deux fois par période, une fois pour l'asservissement de vitesse et l'autre pour l'asservissement de position , la valeur de MLI prend la plus petite valeur entre les deux (MLI\_S ou MLI\_P).

**Remarque :** on arrête la décélération quand "com\_speed" est inférieure à 60Tr/s pour éviter l'arrêt du moteur avant que la position consigne est atteinte.

Donc, on fait l'asservissement de vitesse tant que la pente d'accélération est positive ou la valeur de "com\_speed" est supérieure à 60 tr/mn. Cette solution nous permet d'avoir une bonne précision concernant l'asservissement de position.

#### IV.5. Protection du matérielle :

Pour éviter les situations indésirables, une stratégie de protection à été implémentée afin d'arrêter le moteur dans le cas où la communication USB est perturbée pour telle ou telle raison.

La stratégie consiste à activer le watchdog du microcontrôleur après le début d'échange des données entre l'interface utilisateur et la carte et efface le timer de watchdog dans chaque routine d'émission ou de réception des données au niveau de microcontrôleur.

Donc, s'il y a un problème de communication USB le watchdog va exécuter une reset qui arrête le moteur automatiquement après deux seconds.

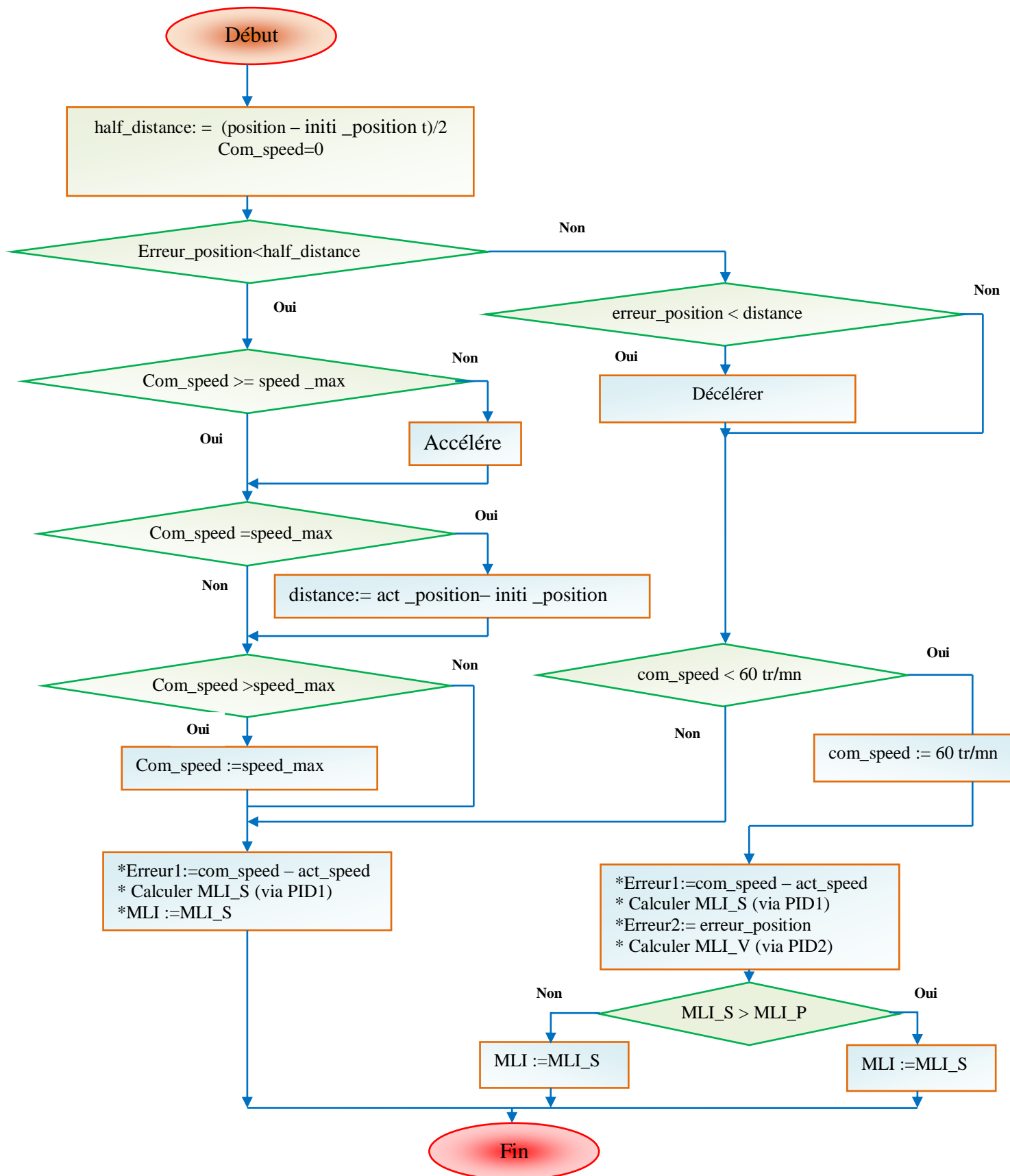


Figure III.24 : L'organigramme de profil de vitesse.

# CHAPITRE IV

## ESSAIS ET RÉSULTATS

## I-Introduction :

Pour tester l'efficacité et les performances de la carte on a effectué une série d'essais.

Les tests sont effectués sur le mode vitesse, le mode position avec ou sans profil de vitesse. Pour le mode vitesse on a changé les paramètres du PID afin de voir l'influence de chaque composant (proportionnel, intégral, dérivée)

Dans ces tests on a utilisé un moteur DC de « globe motors » 24V, solidaire à un encodeur optique « HEDS-5505-A04 » à 500 impulsions par tour.

L'utilisation du mode de comptage quadrature- X4 du circuit LS7266 nous a permis d'augmenter la résolution du comptage jusqu'à 2000 impulsions par tour, ce qui améliore les performances de la carte.

L'utilisation de Microsoft Excel et la base de données générée par l'application nous facilite le traçage et l'exploitation des résultats de chaque test.

### I.1-Le mode vitesse :

#### Test 1 : régulateur proportionnel.

Via l'interface utilisateur on a introduit les paramètres suivants :

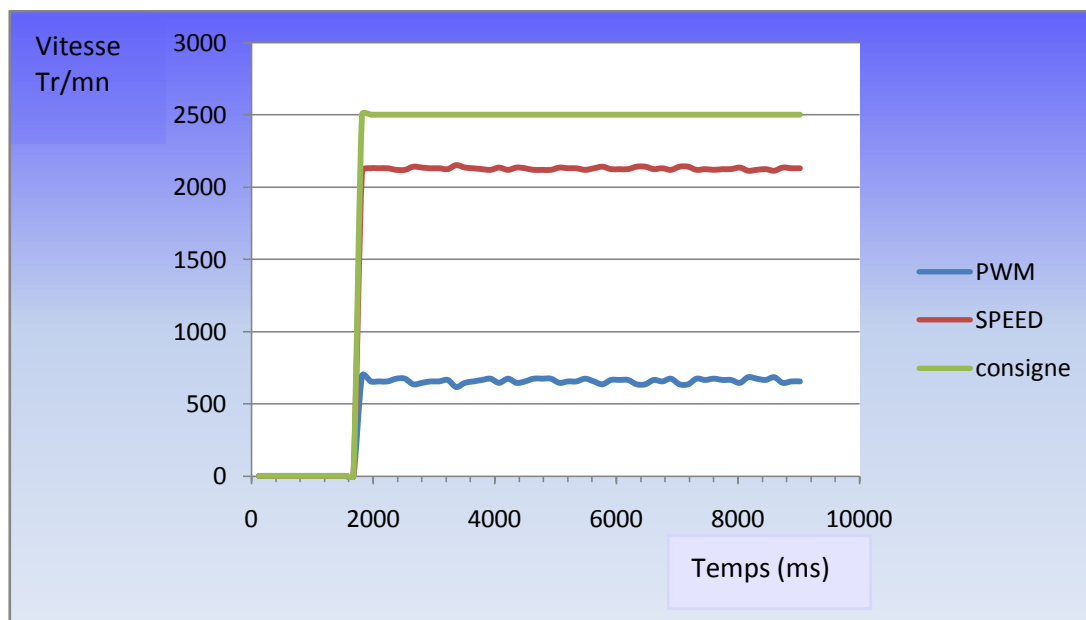
Vitesse consigne : 2500 Tr / mn.

$K_p = 5000$ .

$K_i = 0$ .

$K_d = 0$ .

On a eu le graphe de vitesse suivant :



**Figure IV.1 :** asservissement de vitesse avec régulateur proportionnel

On remarque que la vitesse instantanée se stabilise autour la valeur 2200 Tr/mn, ce qui nous donne une erreur statique de 300 Tr/mn.

On doit utiliser donc un terme intégrateur pour corriger cette erreur statique.

**Test 2 : régulateur proportionnel et intégrateur.**

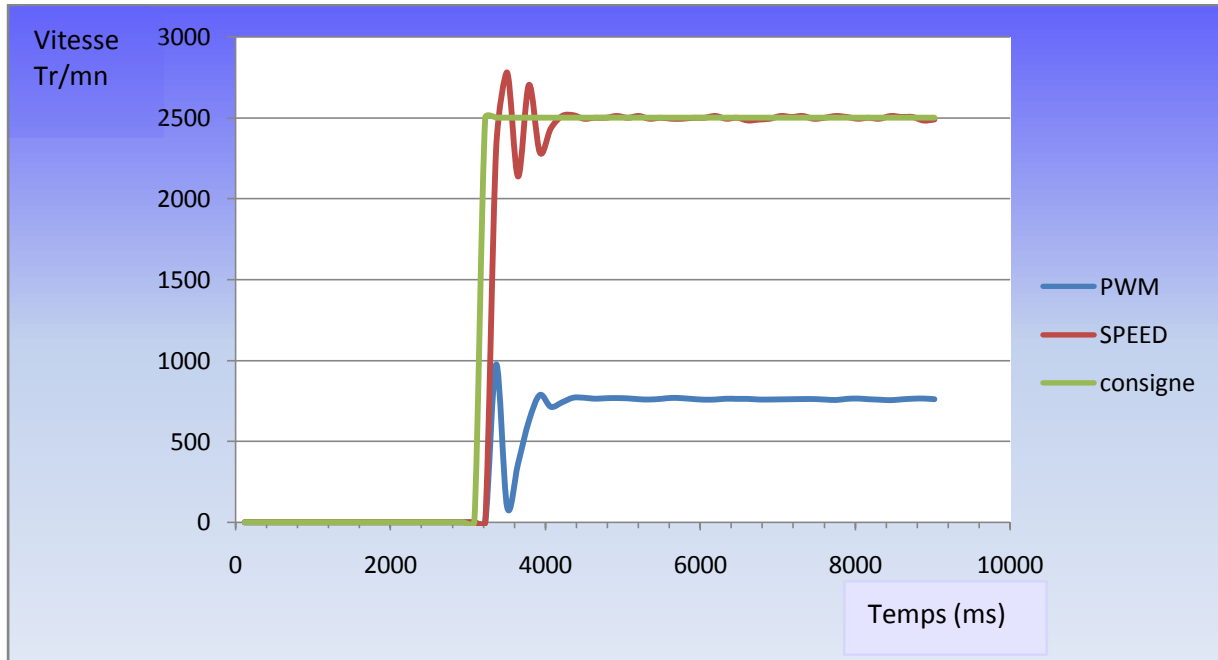
Dans le deuxième test on garde la même vitesse consigne et on change les paramètres du PID comme suit :

$K_p=50$ .

$K_i=500$ .

$K_d=0$ .

On a eu le graphe de vitesse suivant :



**Figure IV.2 :** asservissement de vitesse avec régulateur proportionnel et intégrateur

On remarque que la vitesse instantanée dépasse la vitesse consigne avant de se stabiliser proche de la vitesse consigne.

Plus le coefficient  $K_i$  est grand, plus le système oscille et plus les dépassements sont importants. Le terme dérivé permet alors de limiter les dépassements et les oscillations.

### Test 3 : régulateur proportionnel, intégrateur et dérivateur(PID).

Le terme dérivateur permet de « lisser » la réponse des moteurs. Pour que cet asservissement soit efficace, il nous faut régler au plus fins les trois coefficients  $K_p$ ,  $K_i$ ,  $K_d$ .

Dans le troisième test on garde la même vitesse consigne et on change les paramètres de PID comme suit :

$K_p= 500$ .

$K_i=100$ .

$K_d=250$ .

On a eu le graphe de vitesse suivant :

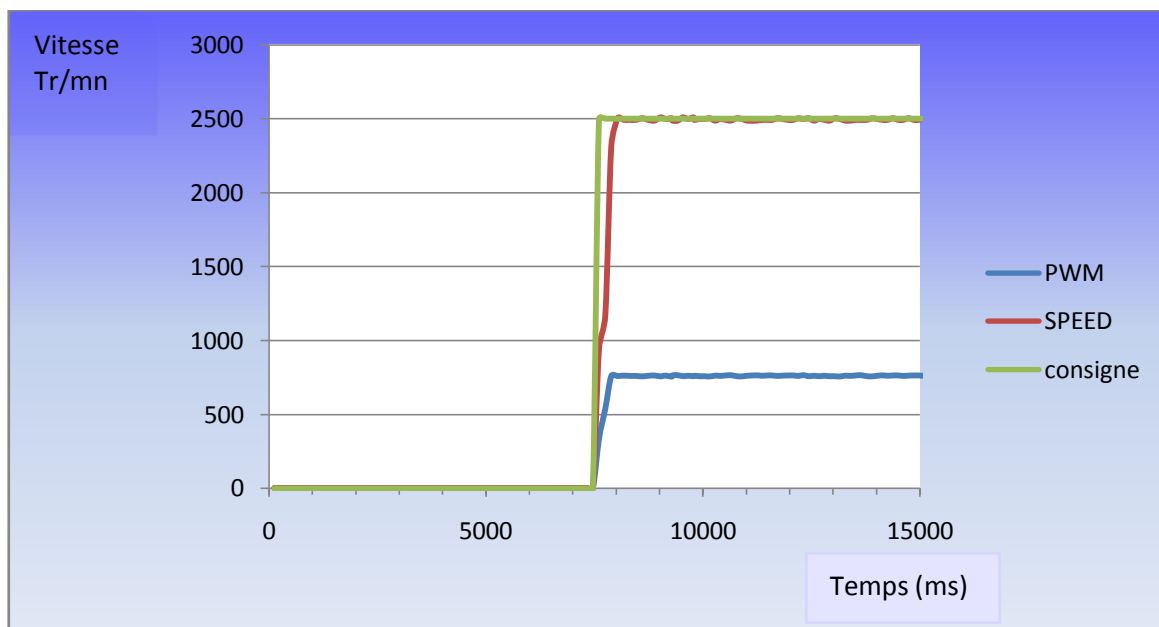


Figure IV.3 : asservissement de vitesse avec régulateur PID.

On remarque l'absence de dépassement de la consigne, un temps de réponse acceptable et une stabilité de la vitesse instantanée.

#### I.2-Le mode Position :

Avec le mode position on peut activer ou désactiver le profil de vitesse, pour cela on a effectué les tests suivants.

#### Teste 4 : sans profil de vitesse.

Via l'interface utilisateur on a réinitialisé le LS7266, ce qui met la position actuelle comme une position de référence « position 0 », et on a introduit les paramètres suivants :

Profil de vitesse : **désactivé**.

Position consigne : **3000000**. ( $3000000 / 2000 = 1500$  Tr)

Vitesse max : **inutilisée**.

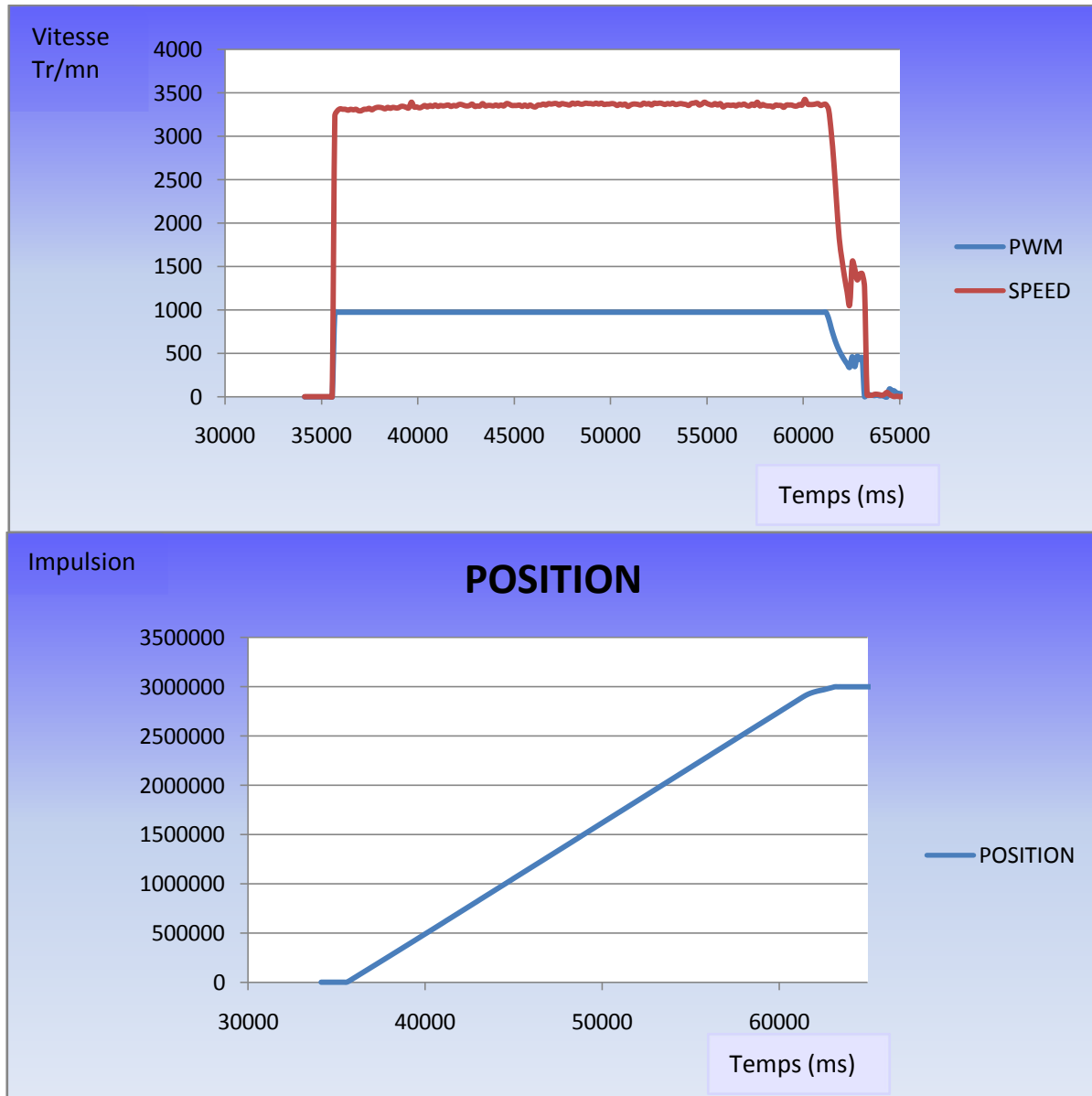


$K_p=500$ .

$K_i=100$ .

$K_d=200$ .

On obtient le graphe de la variation de la vitesse et de la commande suivant :



**Figure IV.4 :** asservissement de position sans profil de vitesse.

On remarque à partir des graphes que la variation de la vitesse a une forme rectangulaire, ce qui implique une variation brusque de la vitesse du moteur. Cette variation cause l'usure excessive des composants mécaniques et dégrade la performance de l'algorithme de compensation.

Pour cette raison, il est préférable d'utiliser un profil de vitesse qui génère une trajectoire de vitesse trapézoïdale ou triangulaire.

**Teste 5:** avec profil de vitesse.

Via l'interface utilisateur on réinitialise le LS7266, ce qui met la position actuelle comme une position de référence « position 0 », et on introduit les paramètres suivants :

Profil de vitesse : **activé**.

Vitesse max : **2500 Tr / mn**.

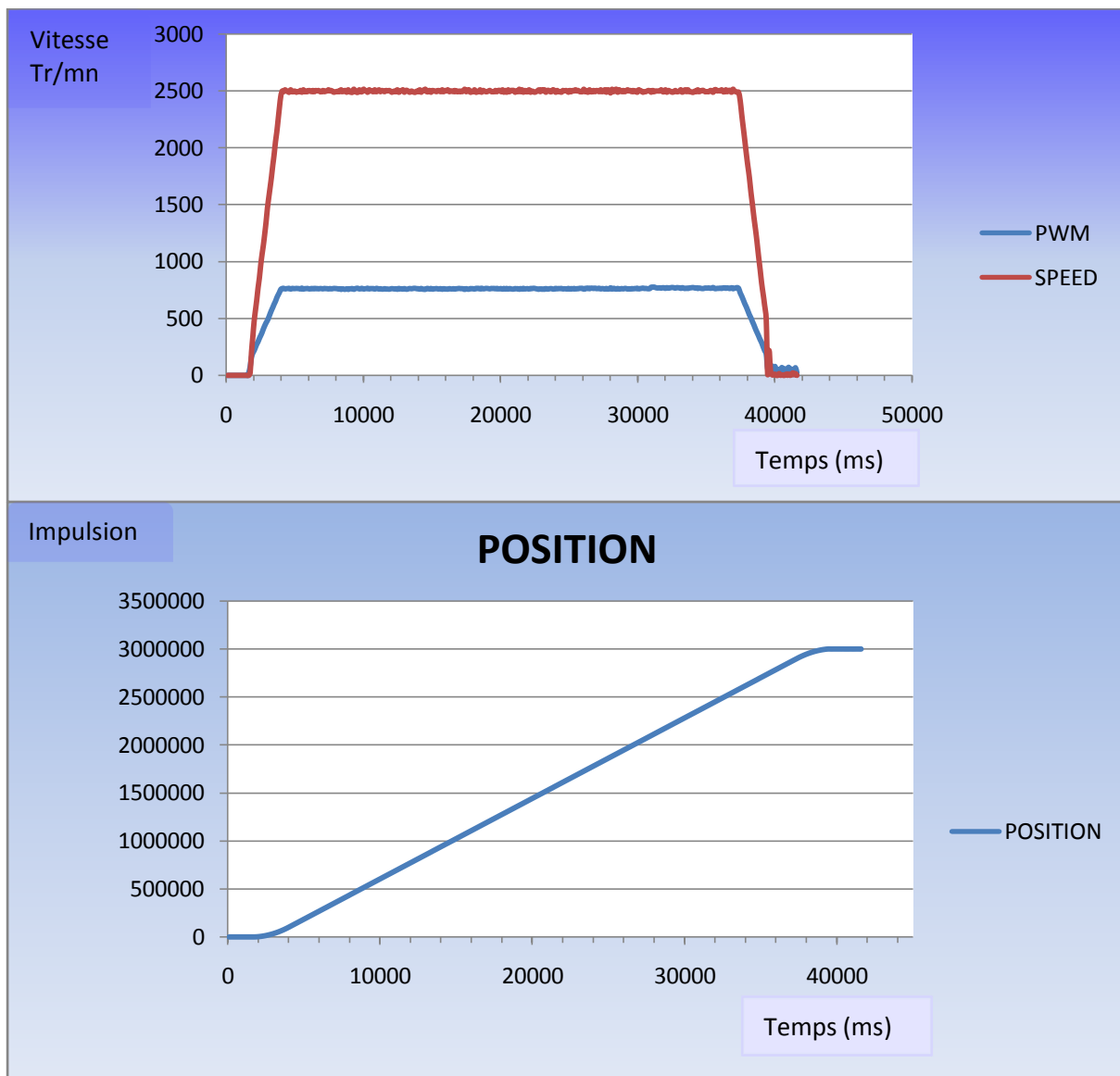
Position consigne : **3000000** .(3000000 /2000 = 1500 Tr )

Kp = **500**.

Ki = **100**.

Kd = **200**.

On aura le graphe de vitesse suivant :



**Figure IV.5 :** asservissement de position avec profil de vitesse

Dans ce test, on a un déplacement de position « 0 » à 3000500 (1500 Tr et 45°), cette distance est très suffisante pour que la vitesse instantanée atteigne la vitesse max, ce qui nous donne une trajectoire de vitesse sous forme trapézoïdale.

On remarque aussi que le graphe de variation de position est plus lisse par rapport au test précédent.

**Test 6 :** avec profil de vitesse et déplacement court.

Via l'interface utilisateur on réinitialise le LS7266, ce qui met la position actuelle comme une position de référence « position 0 » et on introduit les paramètres suivants :

Profil de vitesse : **activé**.

Vitesse max : **3000 Tr/mn**.

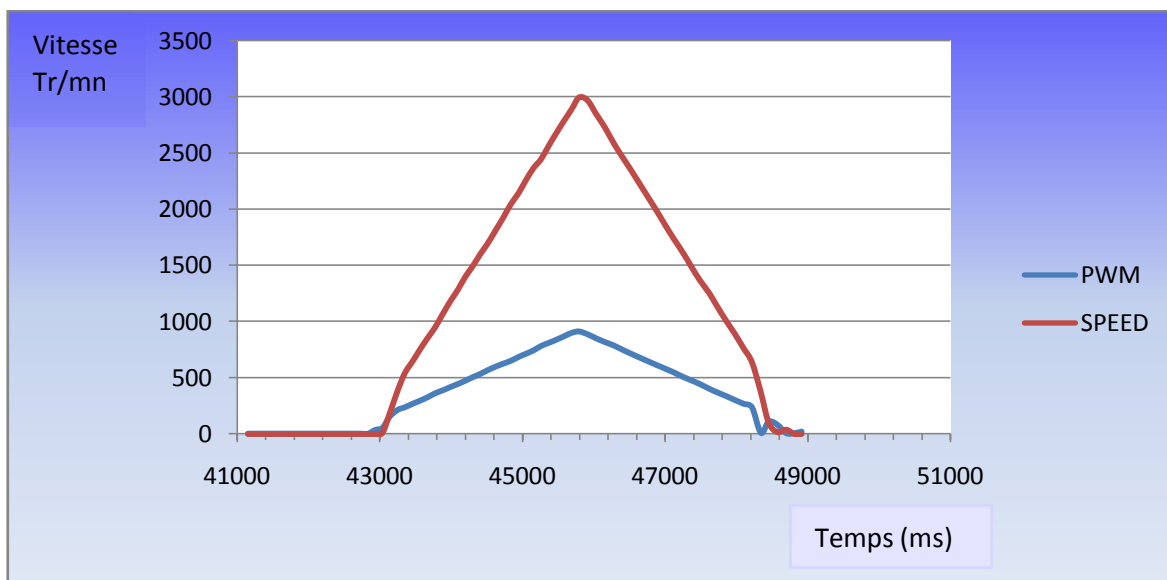
Position consigne : **350000**. ( $350000 / 2000 = 175$  Tr )

Kp = **500**.

Ki = **100**.

Kd = **200**.

On aura le graphe de vitesse suivant :



**Figure IV.6 :** asservissement de position sans profil de vitesse et déplacement court.

Dans ce test, on a un déplacement de position de « 0 » à 350000 (175 Tr), cette distance est insuffisante pour que la vitesse instantanée atteigne la vitesse max, ce qui nous donne une trajectoire de vitesse sous forme triangulaire.

## Conclusion générale

La liaison USB est une nouvelle génération de la liaison série classique, elle offre des solutions acceptables pour les différentes applications rencontrées actuellement vu les avantages suivants :

- ☞ Interface pratique et disponible (jusqu'à 127 périphériques).
- ☞ Interface universelle qui peut supporter tous les périphériques.
- ☞ Le faible coût, la rapidité, la facilité de connexion et déconnexion, le branchement et le débranchement à chaud, la souplesse de leur câble et toutes les autres performances encouragent l'utilisation du port USB.

L'objectif de ce mémoire est :

- de nous familiariser avec USB
- Réaliser les fonctions logiques suivantes :
  - Interface USB
  - Gestion du protocole de communication.
- Réaliser une carte prototype, pour la commande d'un moteur à courant continu, ayant les fonctions de base suivantes :
  - Recevoir les informations de consigne de vitesse, le sens de rotation, le mode choisi....
  - Acquisition de la vitesse et la position du moteur.
  - Pilotage des moteurs à courant continu selon différents modes.

Nous pouvons conclure que ces objectifs ont été atteints, car une carte de commande de moteurs à courant continu via un port USB a été réalisée et testée avec succès.

Nous avons réalisé la commande MLI, avec la possibilité de changer ses paramètres à partir de l'interface utilisateur sur l'ordinateur.

Pour la communication USB nous avons utilisé une méthode développée par Microchip en 2004 (**RS-232 Emulation over USB**). Cette méthode fournit les avantages suivants :

- Elle a peu ou pas d'impact sur l'application logiciel de l'ordinateur.
- Elle exige des changements minimaux aux codes d'applications (RS-232) existants.
- Elle minimise la durée du cycle de développement.
- Elle élimine la nécessité de supporter et mettre à jour un driver de Windows qui est une tâche trop exigeante.

Nous avons testé notre carte avec différents moteurs à courant continu et sur différents ordinateurs, sous Windows millenium et XP, de bonnes performances ont été obtenues.

Le contrôle du moteur est très précis au plus d'un 1/5 de degré (360°/2000). Puisque l'encodeur est relié directement à l'axe de moteur, on le garantit de détecter chaque petite rotation.

Le compteur quadrature utilisé LS\_7266 a des registres de comptage de 24 bits, ce qui nous permet d'aller jusqu'à 8388 Tr (pour 2000 impulsion / tour) sans déborder le compteur.

L'implémentation du profil de vitesse améliore la performance de l'asservissement de position et assure la longue vie du moteur et des mécanismes associés.

La protection du matériel est faite par l'utilisation de timer de watchdog qui arrête le moteur automatiquement lorsque il ya un problème de communication USB.

La protection de l'ordinateur et de la carte de commande est faite par l'utilisation des optocoupleurs afin d'éviter les surtensions probables de l'étage de puissance.

L'environnement MPLAB IDE, MPLAB C18 et le KIT MPLAB ICD2 ont été utilisés pour l'implémentation et le débogage sur le PIC18F4550. Le langage DELPHI a été utilisé pour réaliser l'interface utilisateur.

L'implémentation de la communication USB et de la commande du hacheur par le microcontrôleur PIC18F4550 ont nécessité la maîtrise de l'architecture de ce dernier et une familiarisation avec son environnement de développement.

Le circuit PIC18F4550 est très intéressant par son rapport qualité/prix, car il a prouvé qu'il est le plus performant dans sa série, avec son module USB qui supporte le Full speed, son module CCP amélioré (4 sorties MLI) qui facilite la commande des systèmes électronique, et sa rapidité qui peut aller jusqu'à 12MIPS.

Nous avons cette carte de commande via l'USB pour un usage dans le laboratoire de robotique à l'Ecole Nationale Polytechnique d'ALGER. Le projet final pourrait être utilisé dans le laboratoire.

L'application est munie d'une base de données simple a exploitée, très utile pour analyser les résultats des expériences et elle peut être utilisée avec Microsoft Excel, ce qui nous permet de visualiser les données acquises aisément sous forme de courbes.

Notons que ce travail n'est qu'un début pour introduire les notions de base à la compréhension du protocole USB et à l'utilisation de la liaison USB qui peuvent servir dans pas mal de domaines entre autre la commande des systèmes en ce qui nous concerne.

Nous espérons qu'il y'aura d'autres travaux autour de l'USB afin de bien maîtriser cette technologie qui est relativement nouvelle et exigeante car elle nécessite des connaissances en informatique et en électronique.

# Bibliographies

- [1] JAN AXELSON, **USB Complete Everything You Need to Develop Custom USB Peripherals**, 2nd Edition, Lakeview Research, 2001.
- [2] [Compaq, Intel, Microsoft, NEC] **Universal Serial Bus Specification** Revision 1.1  
September 1998.
- [3] Xavier Fenard, **Le bus USB : Guide du concepteur**, Dunaud 2004.
- [4] Don Anderson ET Dave Dzartko, **Universal Serial Bus System Architecture**, mindshare. 2<sup>nd</sup> Edition 2001.
- [5] Microchip Technology Inc, **PIC18F2455/2550/4455/4550 Data Sheet**,  
www.microchip.com ,2004.
- [6] Matthieu KUHN, **L'USB ET SA NORME**, u.s.b.free.fr,2002.
- [7] USB Implementers Forum Inc, **USB Specification Revision 2.0**,  
www.usb.org/developers/usb20 , 2000.
- [8] USB Implementers Forum Inc, **USB Class Definitions for Communication Devices** Version 1.1, www.usb.org, 1999.
- [9] Microchip Technology Inc **,Emulating RS-232 over USB with PIC18F4550**,  
www.microchip.com ,December 2004.
- [10] Microchip Technology Inc, **PICmicro. 18C MCU Family Reference Manual**,  
www.microchip.com ,2000.
- [11] BIGONOFF, **La programmation des pics la gamme mid-range par l'étude des 16F87X**, <http://www.abcelectronique.com/bigonoff/>,révision 1.1.
- [12] BIGONOFF, **La programmation des pics Migration vers les 18FXX8**, Révision  
alpha 2, <http://www.abcelectronique.com/bigonoff/>.
- [13] Microchip Technologie Inc, **MPLAB C18 C COMPILER USER'S GUIDE**,  
www.microchip.com.
- [14] USB Implementers Forum, **Universal Serial Bus Communications Class Subclass Specification for Ethernet Emulation Model Devices**, Revision 1.0, www.usb.org,  
February 2005.
- [15] USB Implementers Forum, **USB Monitor Control Class Specification**, Revision 1.0,  
www.usb.org, January 1998.
- [16] USB Implementers Forum **Device Class Definition for Human Interface Devices (HID) Firmware Specification** Version 1.1, www.usb.org , June 2001.
- [17] Microchip Technologie Inc **Power Management for PIC18 USB Microcontrollers with NanoWatt Technology**, www.microchip.com.,2004.
- [18] D'AZZO, J. J., HOUPIS, C. H. **Linear Control Systems : Analysis and Design**, 4<sup>th</sup>  
edition, McGraw-Hill,1995.
- [19] KUO, B. C. **Automatic Control Systems**, Prentice-Hall, 8<sup>ème</sup> edition, 2003.
- [20] PHILLIPS, C., HARBOR, R. **Feedback Control Systems**, 2nd edition, Prentice-Hall,  
1991.

- [21] Microchip Technologie Inc, **Low \_cost Bidirectional Brushed Dc motor control**, [www.microchip.com](http://www.microchip.com).
- [22] Microchip Technologie Inc. **Brush-DC Servomotor Implementation using PIC17C756A**. [www.microchip.com](http://www.microchip.com).
- [23] LSI Computer, systems, **24\_BIT DUAL\_AXIS QUADRATURE COUNTER**, [www.alldatasheet.com](http://www.alldatasheet.com),2004.
- [24] CHOWDHURY S. P. ; BASU S. K. ; MONDAL R, **A laboratory model of microcomputer based speed control of a d.c. motor**, IEEE/PES International Power Meeting-India, New Delhi , INDE (28/10/1990) 1992, vol. 7, no1, pp. 403-409 (7 ref.)
- [25] Tang K.S., Kim Fung Man, Guanrong Chen, Kwong S, **An optimal fuzzy PID controller**, Industrial Electronics,IEEE Transactions on , Volume: 48 , Issue: 4 Aug. 2001 Pages:757 – 765
- [26] LIN Chun-Liang (1) ; JAN Horn-Yong (1) ; SHIEH Niahn-Chung, **GA-based multiobjective PID control for a linear brushless DC motor**, Institute of Electrical and Electronics Engineers, New York, NY, ETATS-UNIS (1996).

**Sites Internet :**

- [ S1 ] <http://www.usb.org/>
- [ S2 ] [http://developer.apple.com/DOCUMENTATION/DeviceDrivers/Conceptual/USBBook/USOverview/chapter\\_2\\_section\\_3.html](http://developer.apple.com/DOCUMENTATION/DeviceDrivers/Conceptual/USBBook/USOverview/chapter_2_section_3.html).
- [ S3 ] <https://www.jungo.com/support/documentation>
- [ S4 ] <http://xavier.fenard.free.fr>
- [ S5 ] <http://www.beyondlogic.org/usbnutshell/>
- [ S6 ] <http://www.poirrier.be/~jean-etienne/notes/structordi/ports/usb.php>
- [ S7 ] <http://www.abcelectronique.com/acquier/usb.html>
- [ S8 ] [http://g.fondeville.free.fr/usb\\_cours.html](http://g.fondeville.free.fr/usb_cours.html)
- [ S9 ] <http://www.usb-by-example.com>
- [S10] <http://www.usbman.com>
- [S11] [http://ltc.cit.cornell.edu/courses/ee476/FinalProjects/s2005/dbb28\\_aml54/](http://ltc.cit.cornell.edu/courses/ee476/FinalProjects/s2005/dbb28_aml54/)
- [S12] <http://membres.lycos.fr/michelhubin/CONSEILS/>
- [S13] <http://www.pulsewan.com/data101/>

---

---

# ANNEXES

---

---



---

## Installation et utilisation de la carte :

- 1- copiez le fichier mchpcdc.inf dans le dossier X:\WINDOWS\inf de votre ordinateur.  
Généralement le dossier inf est un dossier caché donc vous devez cocher l'option d'affichage des dossiers cachés (figure A.1).

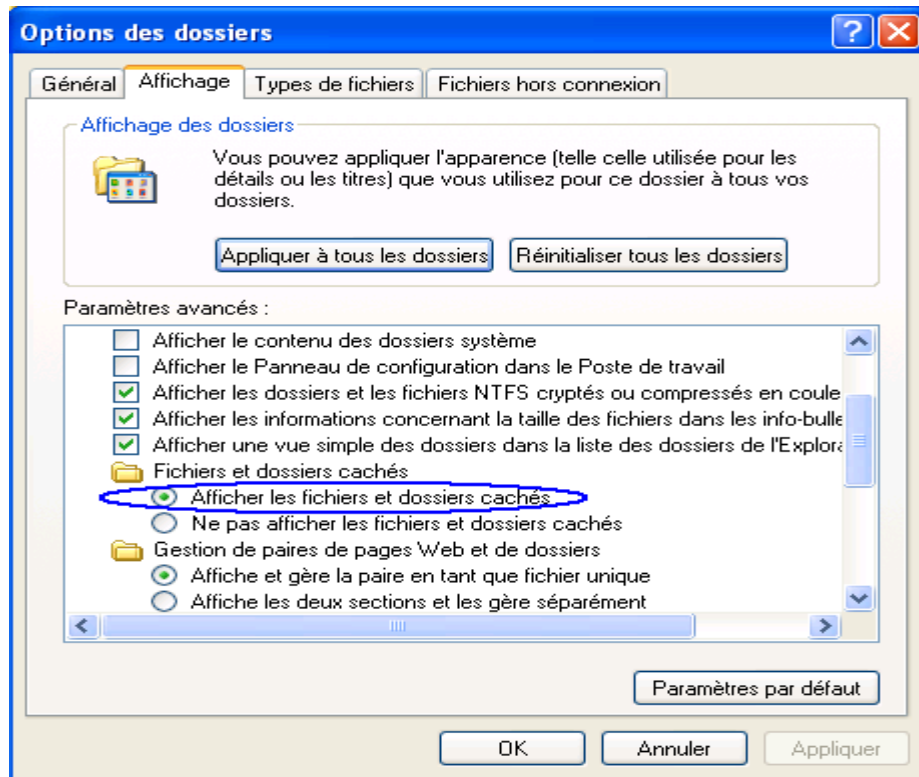
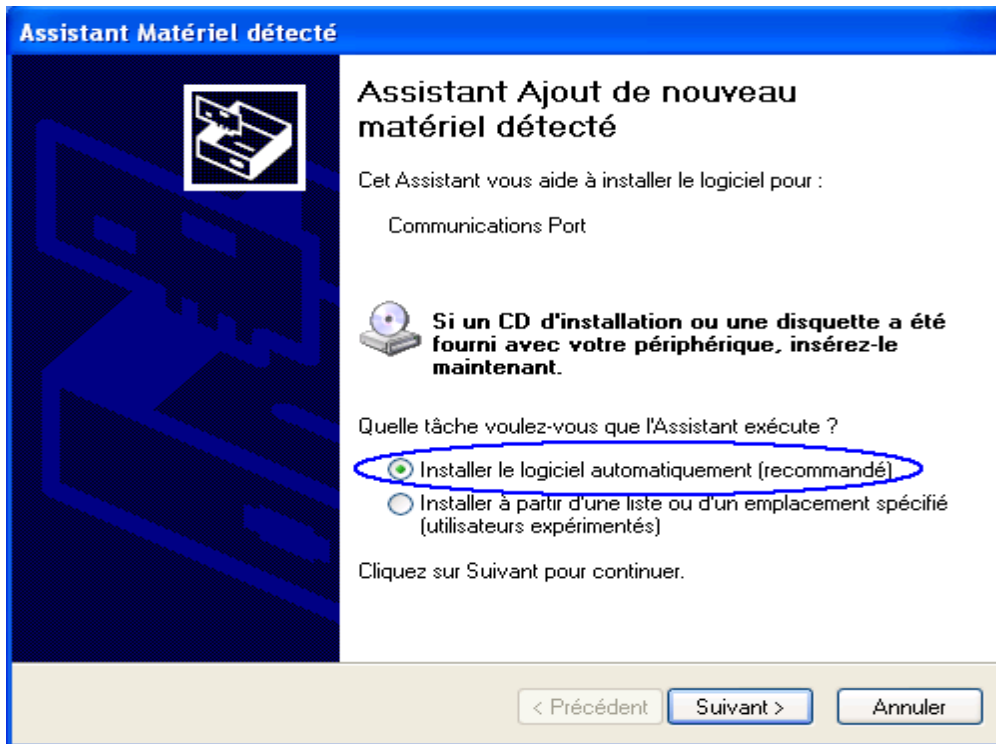


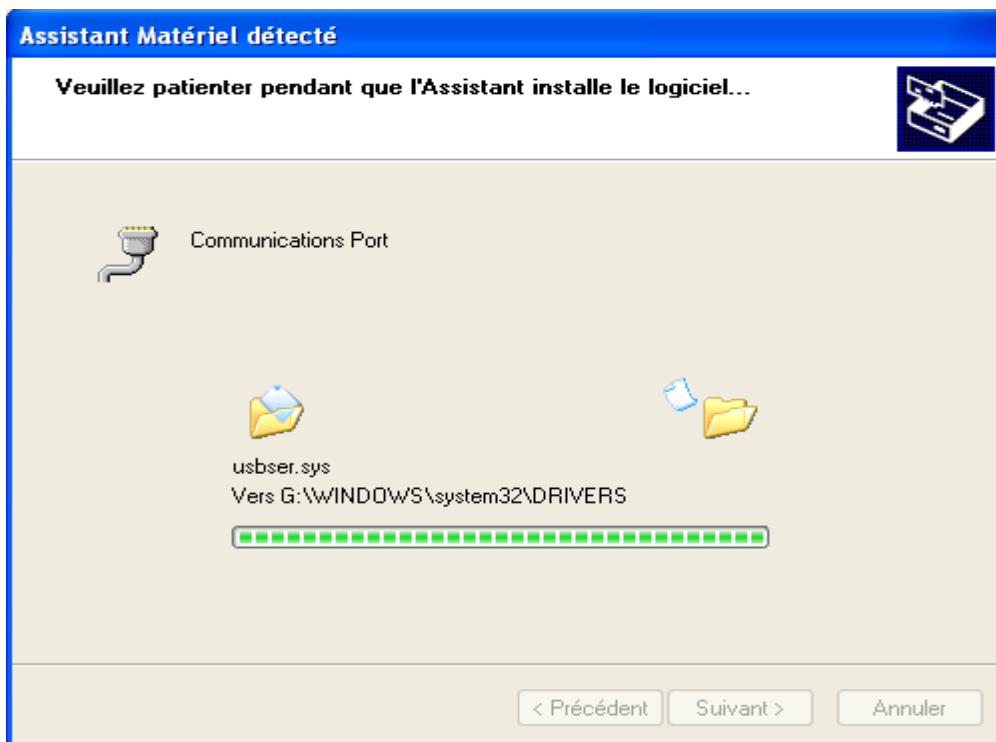
Figure A.1

- 2- copiez le fichier USB.exe sur votre ordinateur.
- 3- branchez le câble USB de la carte à l'ordinateur, la fenêtre suivante va apparaître.



**Figure A. 2**

**4-** sélectionnez (installer le logiciel automatiquement) et vous allez voir les fenêtres suivantes successivement.

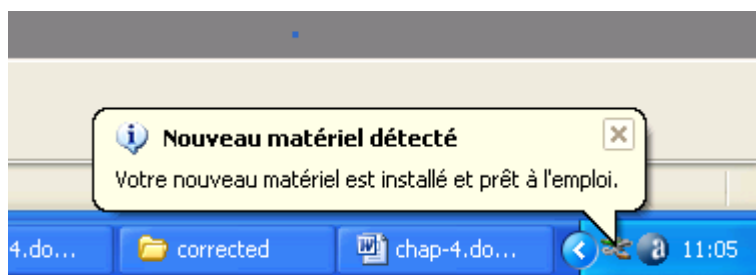


**Figure A. 3**



**Figure A. 4**

**5-** cliquez sur terminer et attendez quelques secondes jusqu'à l'apparition de la fenêtre suivante.



**Figure A. 5**

**6-** débranchez le câble USB et puis double clic sur le fichier USB.exe.  
La carte est bien installée.

## Remarque :

Toutes les étapes précédentes se font une seule fois pendant le premier branchement de la carte seulement. Au prochain branchement la carte sera reconnue automatiquement.

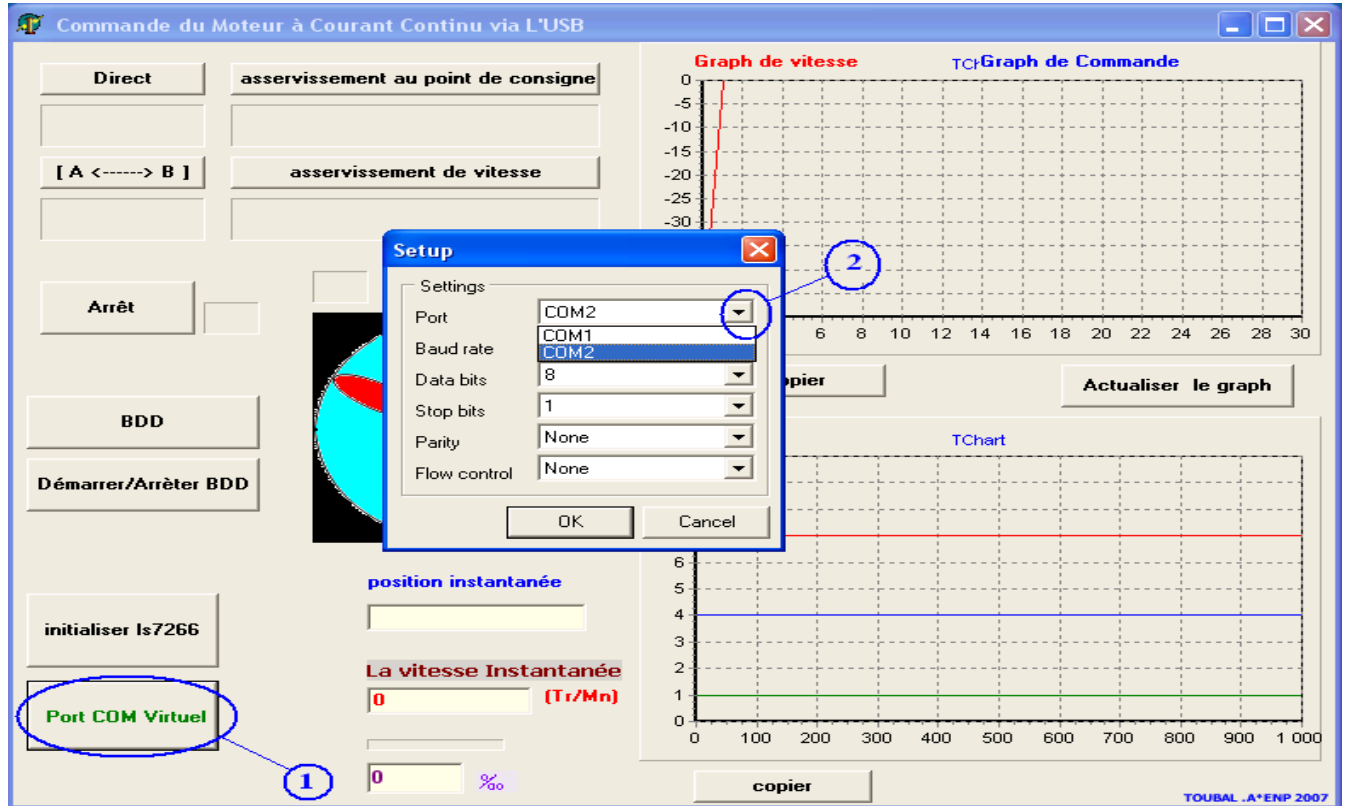


Figure A. 6

- 7- Un simple clic sur le bouton (Port COM Virtuel) puis un simple clic sur l'éditeur Port. Fermez le fichier USB.exe puis rebranchez le câble USB de la carte.
- 8- alimentez la carte et connectez-la au moteur et à la source de tension continue.
- 9- double clic sur le fichier USB.exe.

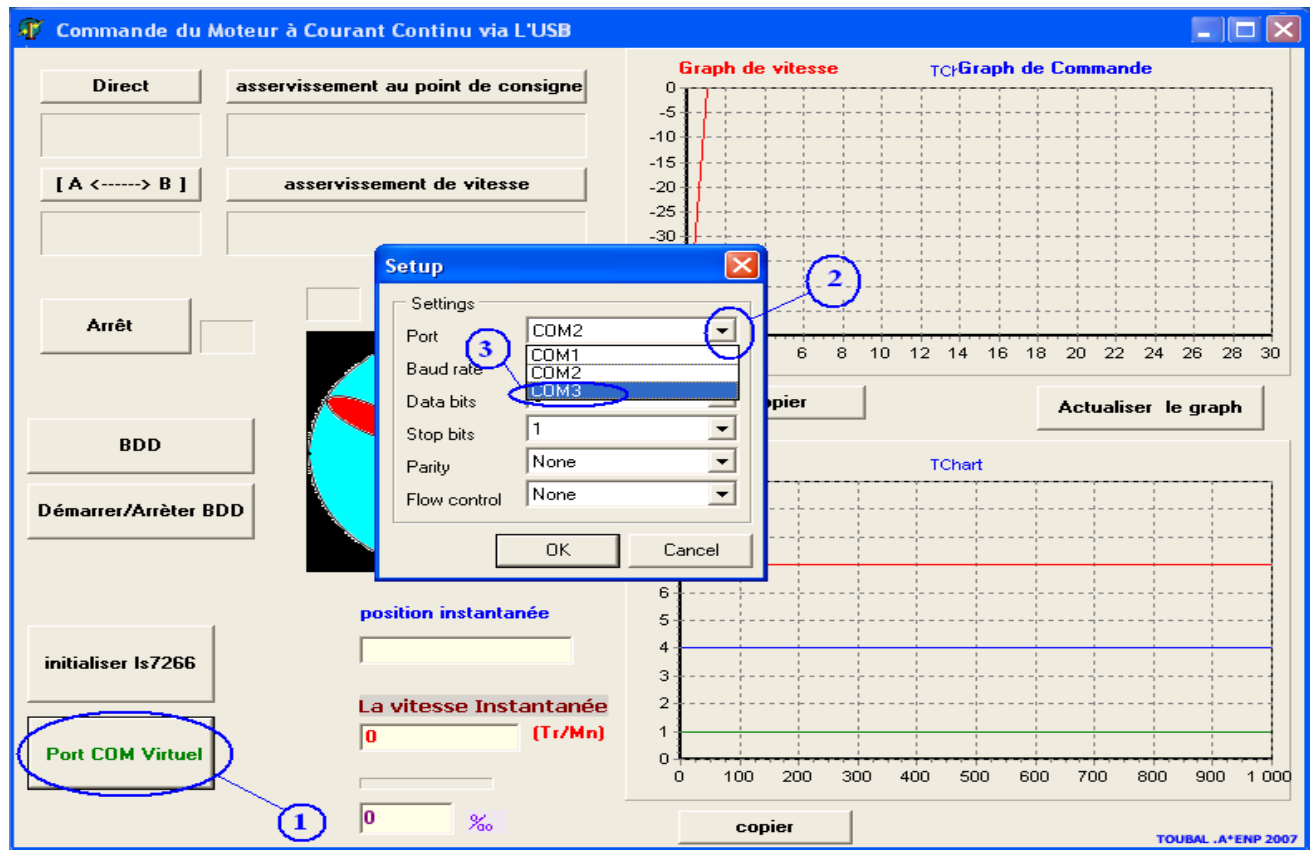
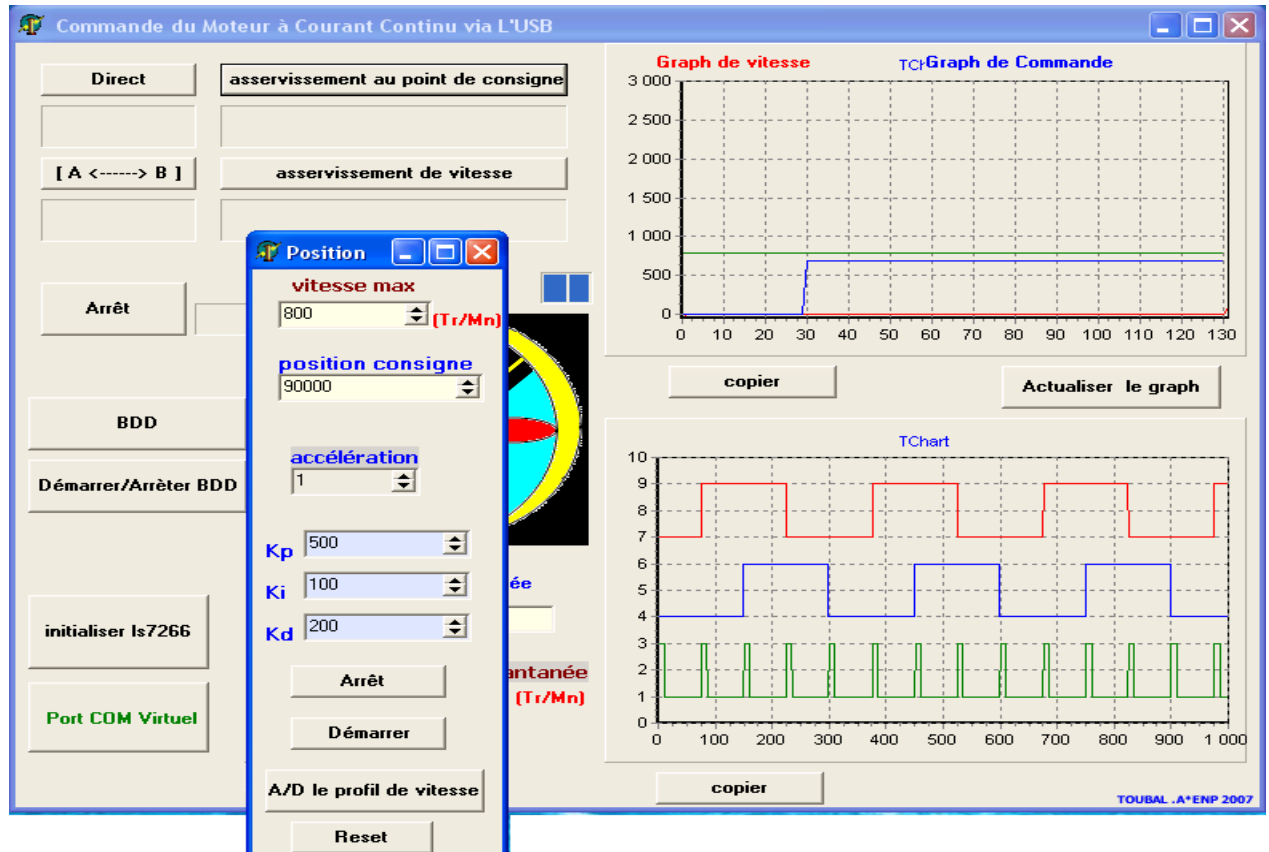


Figure A. 7

- Un simple clic sur le bouton (Port COM Virtuel) puis un simple clic sur l'éditeur Port.
- Vous allez voir un autre COM (virtuel) ajouté sur votre ordinateur, ce n'est pas forcément COM3 il peut être COM4..5..6 , ça dépend de l'ordinateur.
- sélectionnez le COM qui est ajouté, vous allez voir le petit carré au dessous du bouton MLI commence a clignote, ça nous indique que l'ordinateur est en communication avec la carte.

Si non : fermez le fichier (USB.exe), reset la carte (poussez le bouton poussoir sur la carte) et ouvrez le fichier (USB.exe) puis sélectionnez le COM ...comme la première fois. Si l'ordinateur communique avec la carte, entrez le nombre des trous du disque du capteur de vitesse pour que le programme puisse calculer la vitesse instantanée du moteur.



**Figure A. 8**

Pour commander le moteur sans asservissement de la vitesse, sélectionnez le bouton MLI, et entrez la valeur du rapport cyclique (MLI) désirée, la valeur de ce rapport cyclique (MLI) est entre 1 et 1000 %.

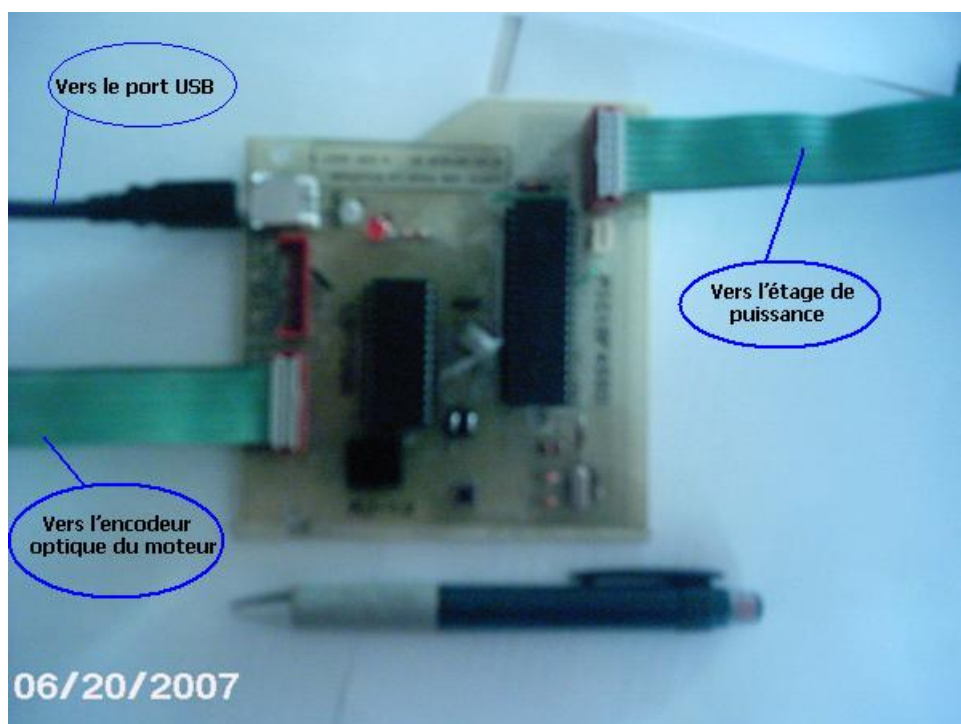
Pour commander votre moteur avec asservissement de la vitesse, sélectionnez le bouton (Asservissement de la vitesse), et entrez les valeurs de Kp,Kd,Ki et la vitesse consigne, à condition que le moteur peut atteindre une vitesse supérieure ou égale à cette vitesse avec la tension max de la source d'alimentation.

Pour commander votre moteur avec asservissement de la position, sélectionnez le bouton (Asservissement au point de consigne), et entrez les valeurs de Kp, Kd,Ki,la vitesse max, le niveau d'accélération et la position consigne, à condition que la valeur de position consigne et appartient à l'intervalle [ -8400000 , 8400000].

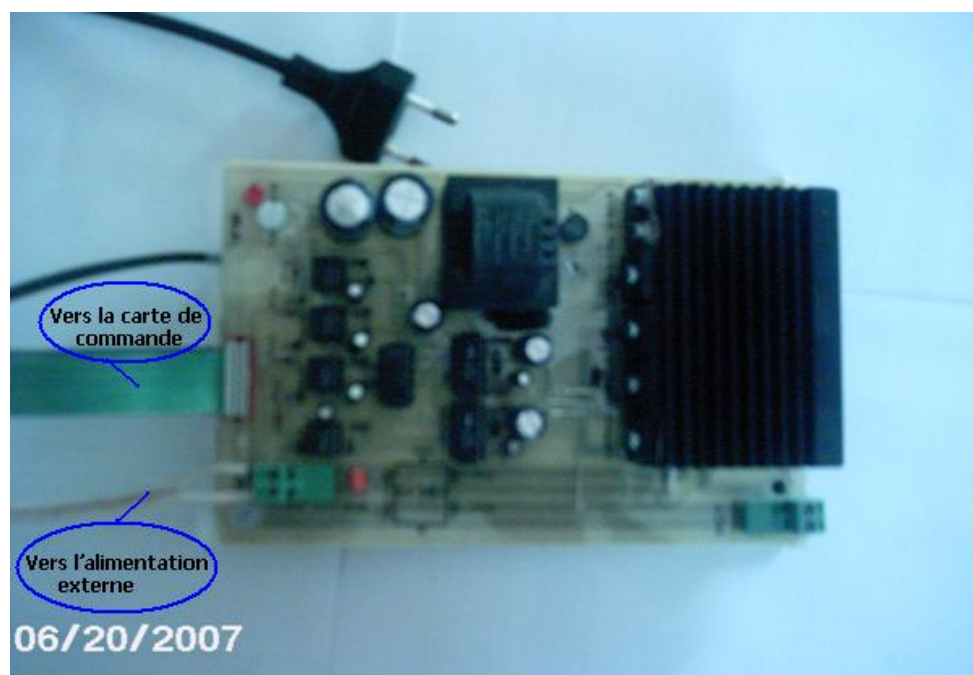
La fenêtre principale affiche :

- La valeur du rapport cyclique (MLI) (%) appliquée au moteur.
- La vitesse instantanée.
- La position instantanée.
- Le graphe de la variation de vitesse et de position.
- Les signaux de l'encodeur selon le mode de comptage choisi via la fenêtre « initialiser LS7266 ».

L'image du moteur sur l'interface tourne quand le moteur est en rotation.



**Figure A. 9 : La carte de commande USB**

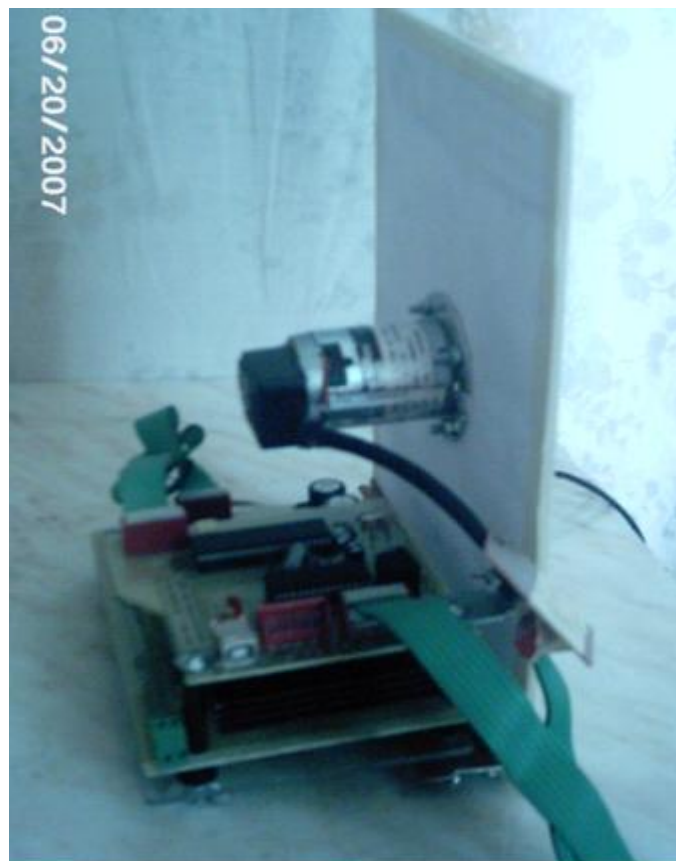


**Figure A. 10 : L'étage de puissance**





**Figure A. 11 : La carte de commande et l'étage de puissance**



**Figure A. 12 : La carte de commande, l'étage de puissance et le moteur DC**