

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

**Ministère de l'Enseignement Supérieur et de la Recherche
Scientifique**



**UNIVERSITE DE M'SILA
FACULTE DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE**

**MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME
D'INGENIEUR D'ETAT EN GENIE ELECTROTECHNIQUE**

OPTION : COMMANDE ELECTRIQUE

THEME

**IMPLEMENTATION D'UN CONTROLEUR
PID POUR LA COMMANDE DES
MACHINES DC SUR UN dsPIC
APPLICATION A LA MACHINE CNC**

Proposé et Dirigé par:

M. MAROUANE BENARROUDJ

Présenté par :

IDHIR NEZZARI

RIAD CHEMALI

FAYCAL LAAMECHE

Année universitaire : 2010 / 2011

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche
Scientifique



UNIVERSITE DE M'SILA
FACULTE DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME
D'INGENIEUR D'ETAT EN GENIE ELECTROTECHNIQUE

OPTION : COMMANDE ELECTRIQUE

THEME

**IMPLEMENTATION D'UN CONTROLEUR
PID POUR LA COMMANDE DES
MACHINES DC SUR UN dsPIC
APPLICATION A LA MACHINE CNC**

Proposé et Dirigé par:

M. MAROUANE BENARROUDJ

Présenté par :

IDHIR NEZZARI

RIAD CHEMALI

FAYCAL LAAMECHE

Année universitaire : 2010 / 2011

Remerciement :

Nous tenons tout d'abord à remercier énormément Monsieur **BENARROUDJ Marouane** Maître Assistant au Département d'électronique Faculté de Technologie Université de M'sila, pour son encadrement et ses précieux conseils grâce à qui ce travail a vu le jour.

Nous remercions également les membres du jury qui ont bien voulu examiner notre travail.

Merci à tous les enseignants qui ont participé à notre formation, sans oublier tous nos amis et camarades.

Dédicace

Je dédie ce travail à mes parents qui se sont sacrifiés tout au long de mes études.

A mes grandes mères.

A mes frères Abdelhak, Zakari et Yasser.

A mes sœurs Saida et Chafia ainsi qu'à leurs maris.

A mes neveux Fouzi, Jakob, les deux Sami, Adem et Yoursen.

A tous mes oncles, tantes, cousins et cousines. Et surtout Ferhat.

A tous mes amis Kamel, Abdessalam, Nista, Fateh, Walid et Reda.

A ceux que j'aime et qui m'aiment.

Idir

Dédicace

Je dédie ce travail à mes parents qui se sont sacrifiés tout au long de mes études.

A mes frères Yacin, Zinelabidine, Fateh, Walid et Aymane.

A tous mes oncles, tantes, cousins et cousines. Et surtout Bilal.

A tous mes amis ; les deux Fouzi et Issam.

Fayçal

Dédicace

A la mémoire de mon père, ainsi à ma mère qui a sacrifié le long de mes études.

A mes frères Abdelhamid, Salim, Fouaz, Marouane et Abdeldjalil.

A mes sœurs Thorya et Nacira.

A tous mes oncles, tantes, cousins et cousines. Et surtout oncle Abdelhamid.

A tous mes amis.

Riad

SOMMAIRE

Introduction générale	01
 CHAPITRE I : Introduction à la CNC	
I.1.Définition.....	03
I.2. Principe de fonctionnement.....	04
I.3. Les composants de la CNC	04
I.3.1. Un contrôleur de machine (Logiciel).....	04
I.3.2. Les pilotes (cartes de commande).....	05
I.3.3. La machine-outil	05
I.3.4. Les codeurs de position	05
I.4. Types des machines CNC	06
I.5. Utilisation	06
I.5.1. Utilisations industrielles	06
I.5.2. Utilisation personnel	06
I.6. Structure de la machine du projet.....	06
I.6.1. Caractéristiques des axes	07
_ L'axe X	07
_ L'axe Y	08
_ L'axe Z.....	08
I.6.2. La broche	08
I.6.3. Les encodeurs	09
I.6.4. L'alimentation	09
I.6.5. Les fins de course	10
I.7. Conclusion	10
 CHAPITRE II : Etude comparative entre Mach3 et EMC2	
II. Logiciels de contrôle	11
II.1. Le logiciel de contrôle Mach3.....	11
II.1.1. Description.....	11

II.1.2. Que peut faire Mach3	12
II.1.3 Modes opératoires de Mach3	12
a) Mode manuel (Jogging).....	12
b) Mode MDI	13
c) Mode automatique.....	14
II.1.4. Le port parallèle du PC.....	14
II.1.5. Configuration de Mach3	14
II.1.5.1. Configuration initiale	14
a) Définition des adresses du Port (s).....	14
b) Signaux de sorties des axes.....	15
c) Signaux d'entrées.....	16
f) Motor Tuning and Setup.....	17
h) Configuration de la broche	17
II.2. Logiciel de commande EMC2	18
II.2.1. Introduction.....	18
II.2.2. Comment marche EMC2	19
II.2.3. Interface de l'utilisateur	20
II.2.4. Configuration d'EMC2.....	21
a) Test de latence	21
b) Adresse des ports	21
c) Réglage du port parallèle.....	22
d) Configuration des axes.....	23
e) Définir la vitesse et l'accélération d'avance.....	23
II.3. PID intégré sur EMC2.....	24
II.4. La différence entre Mach3 et EMC2	24
II.5. Niveau des signaux logiques.....	25
II.5.1. Comment travail les signaux de pas	25
II.6. Conclusion	25

CHAPITRE III : Identification et régulation du système

III.1 Identification	26
III.1.1 Introduction	26
III.1.2 Modélisation et choix du moteur à courant continu :.....	26
III.1.2.1. Introduction :.....	26

III.1.2.2. Description d'un moteur à courant continu	26
III.1.2.3. Modélisation mathématique	27
III.1.2.4. Fonction de transfert du moteur	28
III.1.3. Identification du modèle du moteur à cc	30
III.1.3.1. Détermination expérimentale des paramètres du modèle	30
A. Détermination de la résistance d'induit R et De la constante K	30
A.1 Détermination de la résistance d'induit R seule	30
A.2 Détermination de la constante K seule	31
A.3 Détermination conjointe de R et K	31
B. Détermination de l'inductance d'induit L	31
C. Détermination du coefficient de frottement F	32
D. Détermination du moment d inertie J	32
III.1.3.2. Identification de type « Boite noire » du moteur à cc	32
III.1.3.2.1. système du premier ordre Boucle ouverte	33
III.1.3.2.2. les méthodes de détermination τ et K	34
III.1.4. Conclusion	36
III.2 Régulation :	36
III.2.1 introduction	36
III.2.2. L'objectif de la commande PID	37
III.2.3. Asservissements et rôle des coefficients :	37
III.2.3.1. Asservissement P :	37
III.2.3.2. Asservissement PI	38
III.2.3.3. Asservissement PID continu	38
III.2.3.4. Résumé sur l'action des paramètres (coefficients)	39
III.2.4. Réglage des coefficients	39
III.2.4.1. Méthode de Ziegler-Nichols	40
III.2.5. Discrétisation de la commande PID analogique	41
III.2.5.1 Asservissement en position avec un régulateur PID discret	41
III.2.5.2 Synthèse du régulateur PID discret	43
III.3. Conclusion	43

CHAPITRE IV : ETUDE DU **dspic30F2010**

IV.1

Introduction	44
--------------------	----

IV.2 contrôleur de signaux numériques dsPIC ® DSC	44
IV.3 L'architecture de traitement des données de dsPIC ® DSC.....	45
IV.3.1 mémoire de DSC dsPIC	46
IV.3.1.1 registre de fonctionnement spécialisé «Special Function Registers» (SFR)..	47
IV.3.1.2 Statique RAM (SRAM).....	47
IV.3.1.3 Mémoire de programme	48
IV.3.2 Modèle du programmeur.....	49
IV.3.3 Représentation des données numériques.....	50
IV.3.4 L'architecture de CPU.....	52
IV.3.5 Le coprocesseur DSP	53
IV.4. Les périphériques de dsPIC30F2010	54
IV.4.1 Contrôleur de moteur pwm.....	54
IV.4.1.1 Introduction.....	54
IV.4.1.2 Registres de contrôle (SFR) du module PWM.....	55
IV.4.2 Interface de l'Encodeur en quadrature (QEI).....	58
IV.4.2.1 Introduction.....	58
IV.4.2.2 Registres de contrôle du module QEI	59
IV.4.2.3 Modes de fonctionnement de l'encodeur en Quadrature.....	60
IV.5 Conclusion.....	60

CHAPITRE V : Réalisation de la carte de commande

V.1.	
Introduction	61
V.2. Conception et réalisation de la carte de commande	61
V.2 .1. Schéma fonctionnel	61
V.2.2.Fabrication du circuit	62
V.2.3. Notre carte de commande	62
V.2.3.1. Alimentation	62
V.2.3.2. Circuit de l'horloge externe	62
V.2.3.3. Les entrées.....	63
- Pas et direction (STEP/DIR).....	63
- QEA et QEB	63

- Port RJ11	63
- Port série	63
V.2.3.4. les sorties	63
- Les signaux Haut et Bas	63
V.2.4. Schéma PCB sous Altium	65
V.3. L'étage de puissance (le Hacheur)	67
- L'isolation.....	69
V.4.1. Extraction de la réponse indicielle de l'axe X	70
V.4.1.1. Récupération des données de l'encodeur	70
V.4.1.2. Conversion des donnés	70
V.4.2. Extraction de la réponse en position	71
V.4.3. Calcule de τ_m et K_m et la fonction de transfert	72
V.4.3.1. Fonction de transfert en vitesse	73
V.4.3.2. Fonction de transfert en position	73
V.5. Configuration et Programmation du dsPIC.....	74
V.5.1. Outils de développement	74
V.5. 2. Configuration du dsPIC30F2010	75
a) Configuration d'Interface (QEI).....	75
b) Configuration du module PWM	75
c) Configuration du module UART	76
d) Configuration de l'Oscillateur	76
V.6. Organigrammes du programme	77
V.7. Implémentation du contrôleur PID	78
V.7.1. Fonctions de la bibliothèque PID et structure de données	78
V.8.1. signaux de commande	80
V.8.2. partie puissance	80
V.9. Choix des coefficients du régulateur PID	81
V.10. Test avec le logiciel de commande Mach3	82
- Programme test.....	82
V.11. Conclusion	82
Conclusion générale	83

La liste des Figure

Figure I.1 Corps d'une machine CNC.....	04
Figure I.2 encodeur optique incrémentale.....	05
Figure I.3 partie mécanique de notre machine.....	07
Figure I.5 les encodeurs.....	08
Figure I.6 l'alimentation.....	09
Figure I.7 les fins de course.....	10
Figure II.1.1 image de l'écran principal de Mach3.....	11
Figure II.2 capture de <i>Jogging</i>	13
Figure II.3 capture de MDI.....	13
Figure II.5 port parallèle.....	14
Figure II.6 menu <i>Config</i>	14
Figure II.7 sélection des ports	15
Figure II.8 menu de sorties au moteurs	15
Figure II.9 signaux d'entrées	16
Figure II.10 onglet de signaux de sorties	16
Figure II.11 capture de réglage du moteur	17
Figure II.12 configuration de la broche	17
Figure II.13 EMC2	18
Figure II.14 interface axis	20
Figure II.15 exemples configuré	21
Figure II.16 réglage du port parallèle	22
Figure II.17 page de configuration de l'axe X.....	23
Figure II.18 définir la vitesse et l'accélération max	23
Figure II.19 schéma fonctionnel d'une boucle PID	24
Figure II.20 forme du signal STEP	25
Figure III.1 : Schéma d'un moteur à courant continu	27
Figure III.2 : Schéma équivalent d'un moteur à courant continu	27
Figure III.3 : Schéma bloc du modèle de moteur électrique en vitesse	29
Figure III.4 : Schéma du moteur	33
Figure III.5 : Courbe de la sortie $U_o(t)$	34
Figure III.6 : Calcul de τ	34
Figure III.7: (a) la réponse indicielle de moteur. (b) estimation de τ_m	35

Figure. III.8 Schéma bloc d'un système avec correcteur PID (Boucle fermée)	37
Figure .III.9 Régulateur PID continu	39
Figure. III.10 estimation de K_u et T_u	40
Figure. III.11 Schéma bloc d'un correcteur PID numérique	41
Figure .III.12 Régulateur PID discret	42
Figure IV.1 : dsPIC30F famille de la Conversion de la puissance et du contrôle de moteur.....	44
Figure IV.2 : dspic302010.....	45
Figure IV.3 : architecture pure de Harvard	46
Figure IV.4 : architecture modifiée de Harvard	46
Figure IV.5 : mémoire de programme de dspic30F2010.....	48
Figure IV.4 : architecture modifiée de Harvard	49
Figure IV.7 : représentation d'un nombre entier Signé en complément à deux	50
Figure IV.8 : représentation d'un nombre fractionnel Signé en Q(15)	51
Figure IV.9 : schéma bloc du module PWM	55
Figure IV.10 : actualisation de la période de PWM en mode de fonctionnement libre.....	56
Figure IV.11 : calcule de la période de PWM.....	57
Figure IV.12 : signaux PWM du bord alignés	57
Figure IV.13 : interface de signaux de l'encodeur en quadrature.....	57
Figure IV.14 : schéma bloc de l'interface QEI	59
Figure IV.15 : signal de l'encodeur en mode X4	60
Figure IV.16 : signal de l'encodeur en mode X2	60
Figure V.1 : Schéma fonctionnel.....	61
Figure V.2 : schéma électrique complet.....	64
Figure V.3 : Implémentation des composants de la carte de commande.....	65
Figure V.4 : Schéma du côté cuivre.....	66
Figure V.5 : figure de la carte.....	67
Figure V.6 : Hacheur de la machine CNC	68
Figure V.7 : Typon de Hacheur.....	69
Figure V.8 : La réponse indicielle en position (step).....	71
Figure V.9 : La dérivée de la réponse du système à un échelon (step/sec).....	72

Figure V.10 : Package du programmeur ICD2.....	74
Figure V.11 : les pins du module QEI.....	75
Figure V.12 : les sorties des signaux PWM.....	75
Figure V.13 : les pins du module UART.....	76
Figure V.14 : vers l'oscillateur extérieur	76
Figure V.15 : organigramme du programme implanté dans le dspic30F2010	77
Figure V.16 : régulateur PID.....	78
Figure V.17 : les deux signaux PWM complémentaires de commande.....	80

Bibliographie

[I.1] : <http://reprap.org/wiki/Gcodes>

[I.2] : H. buyse <<capteur de position>>. Université catholique de Louvain

[I.3] : P.Hodod-Daniel JF-kelly <<Build your own CNC Machine>>

[II.1] : << Mach3 CNC Controller Software Installation and Configuration >> Version3

[II.2] : << Using Mach3 Mill >>A user's guide to installation, configuration and operation

[II.3] : << Manual integrator de EMC2>>

[II.4] : << EMC2 pour débutant>>

[III.1] :P. Barret «Machines Electriques», Ellipses, Paris, 2002

[III.2] : « Conversion électromécanique d'énergie La Machine à Courant Continu »

Lycée Richelieu TSI 1 Sébastien GERGADIER 2007

[III.3] :thèse de doctorat (moncef Gossa) l'institut national des sciences appliqués de Toulouse « modélisation dans l'espace d'état et commande numérique d'un moteur à courant continu»

[III.4] : Chi-tsong chen << Analog and Digital Control System Disign >> Stat
University of new york at stony Brook

[IV.1] : <<dsPIC30F Family Reference Manual >> © 2003 Microchip Technology
Inc.

[IV.2] : <<dsPIC30f data sheet>> © 2003 Microchip Technology

Introduction générale

Durant ces dernières années, la commande des machines électriques a subi des progrès significatifs. Ces progrès sont essentiellement dus à la révolution technologique en électronique numérique, ce qui a permis le développement de solutions numériques efficaces avec une possibilité d'implanter des algorithmes plus complexes. Les premières implantations d'algorithmes de commande de machines électriques furent réalisées avec des solutions analogiques. Ces solutions assuraient la réalisation de contrôles ayant une large bande passante et une haute résolution. Cependant, elles manquaient de fiabilité, vu leur sensibilité aux perturbations et aux variations de paramètres de contrôle liées aux contraintes thermiques des circuits analogiques de contrôle. Par la suite, les solutions numériques se sont naturellement imposées afin de remédier à ces inconvénients.

De nos jours, les implantations numériques sont en majorité basées sur les microprocesseurs et les DSP (*Digital Signal Processor*). Ces solutions numériques sont équipées d'unités arithmétiques et logiques ALU (*Arithmetic Logic Unit*) dédiées à la réalisation des calculs arithmétiques et logiques des algorithmes de commandes. Elles intègrent également des périphériques tels que les convertisseurs analogiques/numériques et les « timers » bien adaptés aux besoins de commande de machines électriques. Leur utilisation a permis de résoudre les problèmes liés à l'utilisation des commandes analogiques. Par ailleurs, elles présentaient un grand intérêt économique et une meilleure flexibilité de conception.

Les machines CNC (computer numerical control) ont fait preuve dans ce domaine. Leur haute précision introduit le besoin intensif d'une meilleure régulation, ce qui répondue avec la régulation PID.

Notre projet a pour but d'implémenter un PID numérique sur dsPIC de Microchip pour la régulation en position des machines CNC tout en passant par la réalisation de la commande.

Ce mémoire contient Cinq chapitres organisés de la façon suivante :

Nous présentons dans le premier chapitre les machines CNC en général, décrivant leurs constituants et leur principe de fonctionnement, nous donnons également une présentation de la partie mécanique de notre machine.

Nous abordons dans le deuxième chapitre les logiciels de commande de la machine CNC, nous parlerons plus précisément sur le Mach3, ses fonctionnements, ses modes opératoires et sa configuration. Et Nous terminons par donner une description des signaux de sortie du logiciel.

Le troisième chapitre est consacré à l'identification des moteurs à courant continu et la régulation PID. Commenant d'abords par un petit rappel sur les moteurs à courant continu, puis nous allons expliquer quelques méthodes d'identification, après nous parlerons sur PID, son objectif et son réglage (méthode de Ziegler Nichols), nous finissons par présenter le PID discret.

Nous décrivons dans le quatrième chapitre le cerveau de toutes opérations de notre machine; le **dsPIC30F2010** et en particulier, son architecture et quelques de ses modules utilisés dans le projet.

Dans le dernier chapitre nous allons réaliser la carte de commande et l'étage de puissance en donnant leurs schémas. Ensuite nous faisons une identification de l'axe X ; extraction de sa réponse et le calcul de la fonction de transfert. Enfin, nous configurons le dsPIC, en donnant un organigramme général du programme et l'implémentation du PID.

Ce travail est ensuite clôturé par une conclusion et quelques prescriptifs pour des travaux futurs.

CHAPITRE I

Introduction a la CNC

I.1.Définition :

CNC ou la Commande Numérique par Ordinateur (*Computer Numerical Control*) est un terme très large qui englobe une variété de types de machines, le tout avec différentes tailles, formes et fonctions. Mais la façon la plus simple de penser à propos de CNC est de simplement comprendre qu'il s'agit de savoir utiliser un ordinateur comme un moyen de contrôler une machine qui sculpte des objets utiles à partir de blocs solides de matériaux.

La conception et la construction de la commande numérique par ordinateur(CNC) diffère grandement de celui des machines-outils conventionnelles. Cette différence s'explique par les exigences de niveaux de performance plus élevé. Les machines CNC emploient souvent les différents éléments mécatronique qui ont été développées au fil des années. Cependant, la qualité et la fiabilité de ces machines dépend des plusieurs éléments: La structure et la matière de construction par exemple les rails et vis de parcourt des axes, les paramètres des moteurs, la commande, le logiciel et l'interface opérateur.

Le contrôle d'une machine outil à l'aide d'informations stockées sur l'ordinateur est appelé la Commande Numérique par Ordinateur. Les informations stockées dans l'ordinateur peuvent être lues par des moyens automatiques et convertis en signaux électriques, nous n'avons qu'à construire des objets sur le logiciel CAD (conception assistée par ordinateur) et de les transformer en G-code [I.1] (script spécifique aux machines CNC) à l'aide du logiciel CAM (fabrication assistée par ordinateur). Un troisième logiciel (qui peut être : Mach3, EMC2, etc.) va interpréter ces codes et va générer des signaux électriques qui vont commander les axes d'avance de la machine CNC.

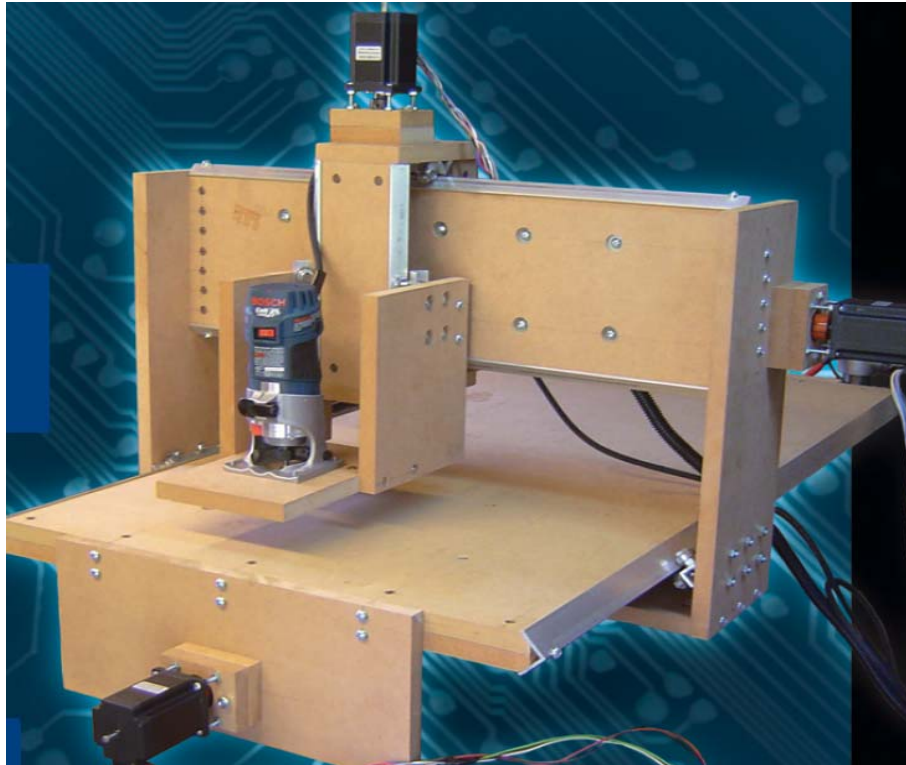


Figure I.1 Corps d'une machine CNC à 3 axes.

I.2. Principe de fonctionnement :

La machine CNC travaille avec des systèmes de contrôle en boucle fermée.

Des ordres vont être générés vers la carte de commande par le biais d'un logiciel spécifique installé sur l'ordinateur (voire chapitre II).

La carte de commande va traiter ces informations et générer des signaux MLI afin d'obtenir les déplacements voulus par le biais des moteurs d'axes. Des contrôles de vitesse et de position seront alors effectués de manière continue par la machine.

I.3. Les composants de la CNC :

I.3.1. Un contrôleur de machine (Logiciel):

Le logiciel lit et interprète le programme pour envoyer les consignes vers les cartes de commande de chaque axe.

I.3.2. Les pilotes (cartes de commande) :

Les signaux de commande de la machine sont adaptés par les pilotes afin qu'ils soient assez puissants et convenablement pour les moteurs d'entraînement des axes de la machine-outil. (Voire chapitre IV)

I.3.3. La machine-outil :

Les axes de la machine déplacent par des vis, des étagères ou des courroies qui sont alimentés par des servomoteurs ou des moteurs pas à pas.

I.3.4. Les codeurs de position :

Ils ont pour rôle de renvoyer la position actuelle d'un axe. Cette information permettra à la CNC de corriger sa consigne jusqu'à la position désirée. [I.2]

Un encodeur optique incrémental à quadrature de phase (figure I.2) est relié au rotor du moteur, ce qui nous permet de récupérer une information sur l'incrémentangulaire du rotor, à partir de laquelle on peut calculer la position par rapport à sa position initiale. Il s'agit d'un disque en rotation doté de deux rangées de fentes décalées. Un système optique détecte ces fentes et génère des impulsions par tour de rotor, sur deux signaux en quadrature de phase.

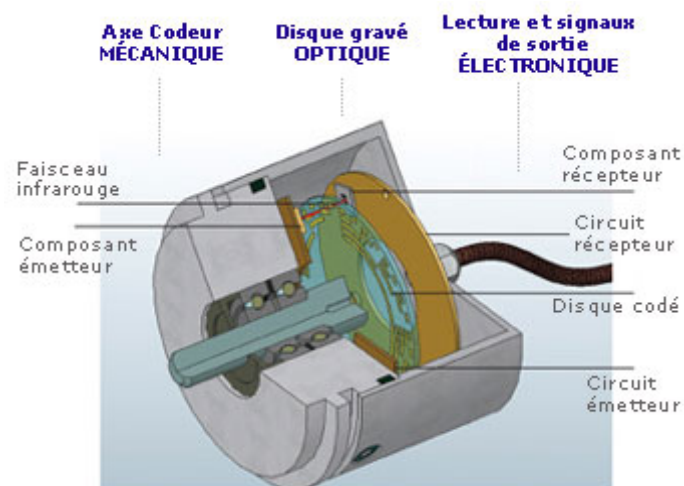


Figure I.2 Encodeur optique incrémentale.

I.4. Types des machines CNC :

La CNC peut être appliquée à fraiseuses, machines de tournage, machines à rectifier, Aléseuses, coupe flamme, machines de forage, etc.

Ces machines varient à leurs fonctionnements et leurs domaines d'utilisation :

- Dans l'industrie d'élimination des métaux.
- Dans l'industrie de transformation des métaux.
- Dans l'industrie de l'usinage par décharge électrique.
- Dans l'industrie du bois.

I.5. Utilisation :

On peut utiliser les machines CNC dans le domaine industriel, comme on l'utilise dans le domaine personnel. [I.3]

I.5.1. Utilisations industrielles :

Les applications industrielles pour les machines CNC sont basées principalement sur la sculpture du métal pour créer une forme désirée. Le métal est largement utilisé pour la production de presque tout ce que nous voyons autour de nous, alors on trouve les machines CNC presque dans toutes les industries.

I.5.2. Utilisation personnel :

Il y a une grande communauté d'amateurs et de bricoleurs dans le monde entier qui est intéressée par le concept de machines CNC. Roboticiens, artisans, bricoleurs, les machinistes à domicile, propriétaires de petites entreprises, passionnés de technologie, et des artistes utilisent les machines CNC dans leurs travaux.

I.6. Structure de la machine du projet :

L'image suivante présente la structure de la machine CNC que nous comptons d'utiliser pour valider notre carte de commande à réaliser. Cette machine, ainsi que notre travail font partie d'un projet PNR proposé par M.BENARROUDJ et accepté par le ministère cette année.

Cette machine est construite en métal et en aluminium, ce qu'il a rendu robuste et précise. Elle est constituée de 3 axes, une broche, 3 encodeurs et 6 fins de course comme il est montré sur la figure (Fig I.3).

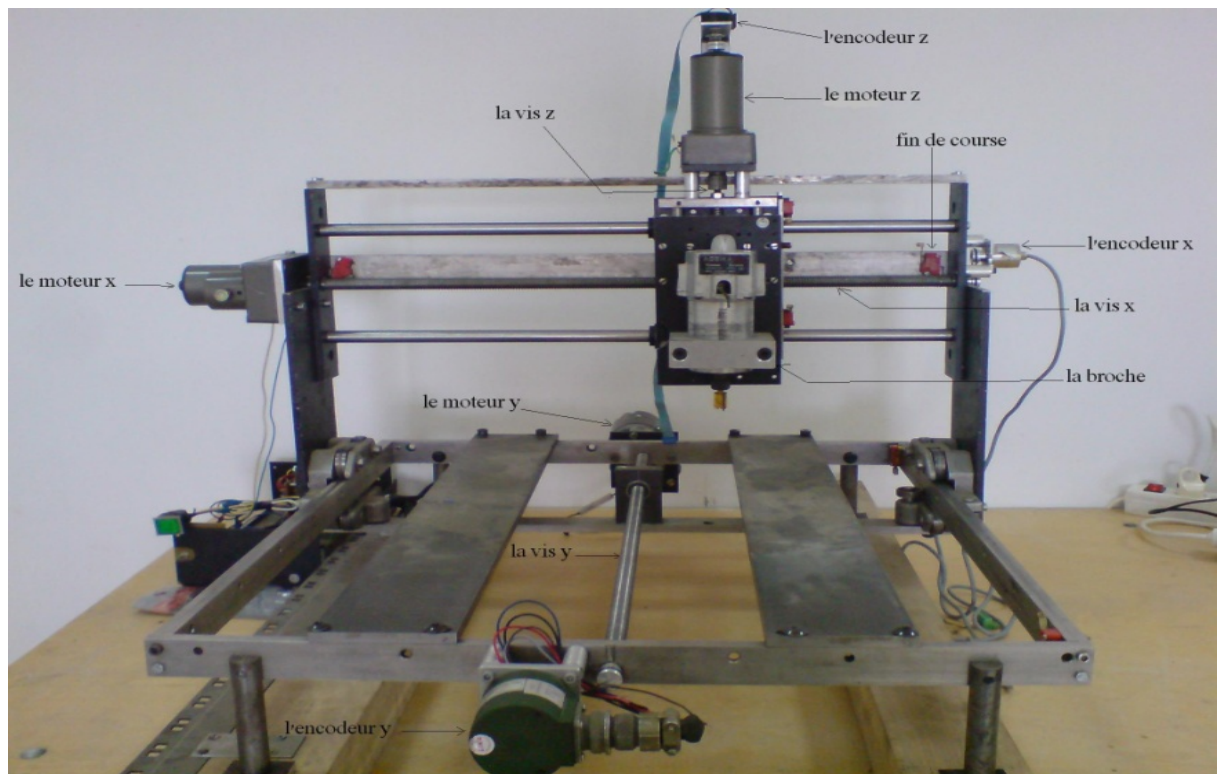


Figure I.3 partie mécanique de notre machine.

I.6.1. Caractéristiques des axes : Nous avons 3 axes X, Y et Z chaque axe possède un moteur DC et un encodeur incrémental spécifique.

L'axe X :

- Moteur : récupéré du vérin de la parabole.
- Alimentation : 35 v (DC).
- Parcours : 665 mm.
- Pas de la vis : 3.175 mm/tour.

- Vitesse max : 238.125 mm/sec.
- Courant max : 3 A.

L'axe Y :

- Alimentation : 35 v (DC).
- Parcours : 665 mm.
- Pas de la vis : 2.83 mm/tour.
- Vitesse max : 238.125 mm/sec.
- Courant max : 3 A.

L'axe Z :

- Moteur : récupéré du vérin de la parabole.
- Alimentation : 35 v (DC).
- Parcours : 115 mm.
- Pas de la vis : 6 mm/tour.
- Vitesse max : 238.125 mm/sec.
- Courant max : 3 A.

I.6.2. La broche :

- La marque : AOSIKA.
- Alimentation : 220-230/110 v (AC).
- Vitesse max : 30000 tour/min.
- Puissance : 440 W.

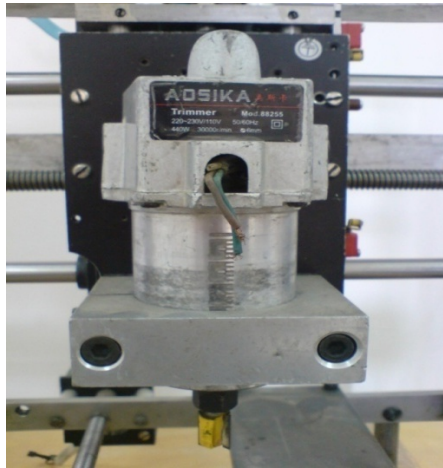


Figure I.4 La broche

I.6.3. Les encodeurs :

Les encodeurs utilisés dans la machine sont récupérés de la casse, il a fallut donc les identifier avant de les utiliser. Nous avons trouvé que l'encodeur correspondant à l'axe X génère 40 impulsions par révolution.

La figure I.5 montre les trois encodeurs utilisés dans le projet.

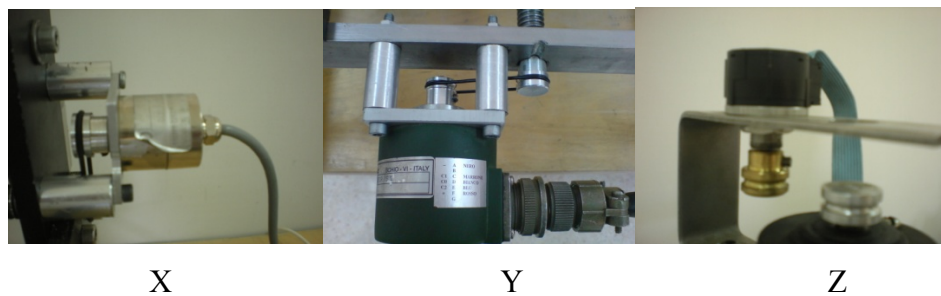


Figure I.5 les encodeurs

I.6.4. L'alimentation :

Notre alimentation comprend transformateur 220V / 35V, un redresseur de tension AC/DC et des dispositifs de protection.



Figure I.6 l'alimentation

I.6.5. Les fins de course :

Chaque axe a besoin de 2 commutateurs placés aux extrémités, pour éviter les dépassements imprévus de l'outil en dehors de son espace de travail.

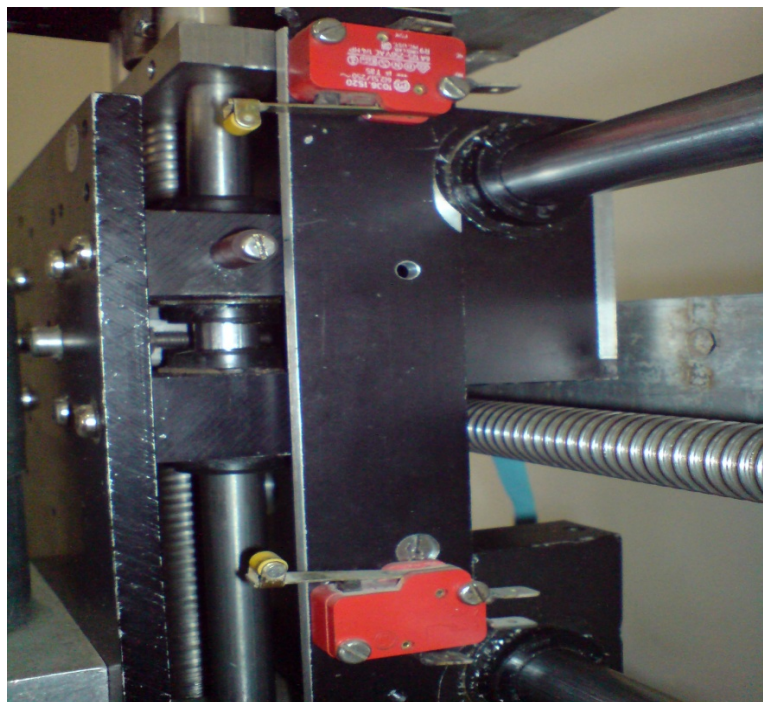


Figure I.7 les fins de course.

I.7 conclusion :

Après avoir étudié la machine CNC nous avons remarqué la nécessité d'un asservissement précis des moteurs pour le bon fonctionnement de la machine. Dans le chapitre III nous allons développer quelques notions sur l'identification et la modélisation des moteurs ainsi que leur asservissement. Ces notions seront plus tard utilisées lors de la réalisation de la carte de commande.

CHAPITRE II

Etude comparative entre

Mach3 et EMC2

II. Logiciels de contrôle

Il y a trois types de logiciels qu'on doit utiliser pour arriver faire fonctionner la machine CNC. Le premier est un logiciel spécial qui permet de concevoir des objets en deux et en trois dimensions, qu'on appelle CAD (*computer aided design*). Le second est le CAM (*computer-aided manufacturing*). Ce Logiciel prend la conception créée par le premier logiciel et la convertit en un langage de G-Code . Ce G-Code est ensuite utilisé par un troisième logiciel, Ce Logiciel génère des commandes (signaux appropriés électriques) via les ports de communications comme le port parallèle ou le port série du PC qui vont servir de consignes aux cartes de commandes associées aux axes d'avances de la machine CNC. Le présent chapitre, nous faisons une étude comparative entre de deux logiciels du troisième type les plus utilisés dans ce domaine. Ce sont les logiciels «Mach3 » et « EMC2 ».

II.1. Le logiciel de contrôle Mach3 [II.1], [II.2]

II.1.1. Description

Mach 3 d'ArtSoft USA est disponible dans une version gratuite et une version commerciale. Les deux versions sont identiques, mais la version gratuite limite le

script qu'elle exécute à 500 lignes contrairement à la version commerciale qui laisse le programme dépassant les 10.000.000 lignes.

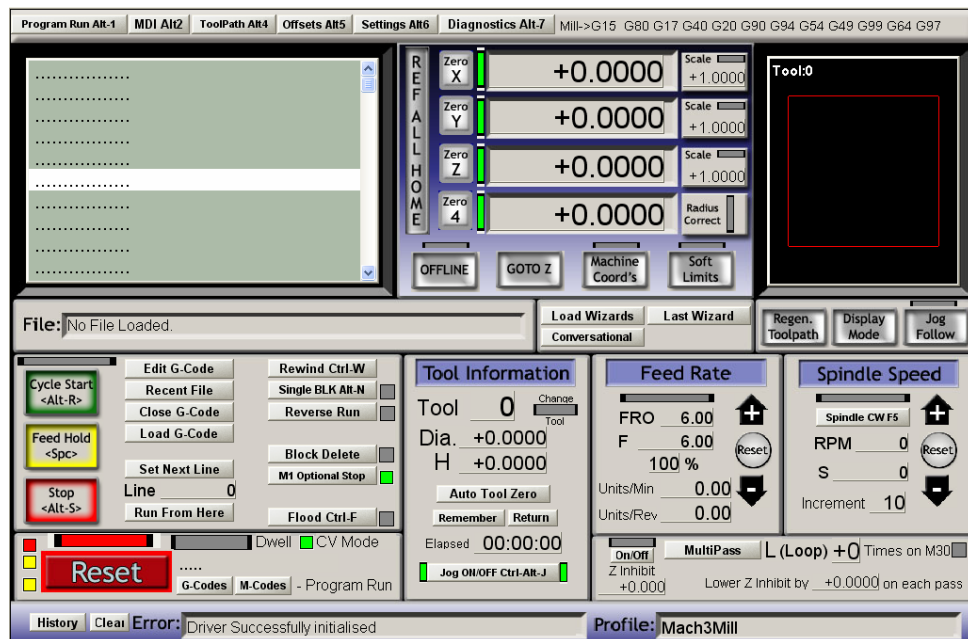


Figure II.1 image de l'écran principal de Mach3.

Au lancement du logiciel Mach3, on remarque qu'il y a beaucoup de boutons, afficheurs, et d'autres éléments qui encombre l'écran. Mais en réalité il y a peu de choses à configurer dans le logiciel pour faire fonctionner la machine.

II.1.2. Que peut faire Mach3

Mach3 est un programme très flexible et paramétrable à n'importe quelle machine CNC. Il demande au préalable qu'on lui définit toutes les caractéristiques de la machine afin qu'il puisse l'actionner. Voici quelques paramètres à définir sur Mach3 :

- Un arrêt d'urgence (Estop), un bouton qui doit être prévu sur chaque machine.
- Type de la consigne à générer, STEP/DIR ou bien Quad A/B.
- Les limites de l'espace de travail.
- Brochage des fins de course avec le PC.
- Paramètres du contrôleur PID de l'outil".
- La vitesse Max pour chaque axe.

II.1.3 Modes opératoires de Mach3

a) Mode manuel (Jogging)

Une fois le Mach3 lancé, Ce mode s'affiche en appuyant sur la touche TAB du clavier. Dans ce mode Nous pouvons commander manuellement la position de l'outil. La figure (II.2) montre l'interface de commande manuelle de Mach3.

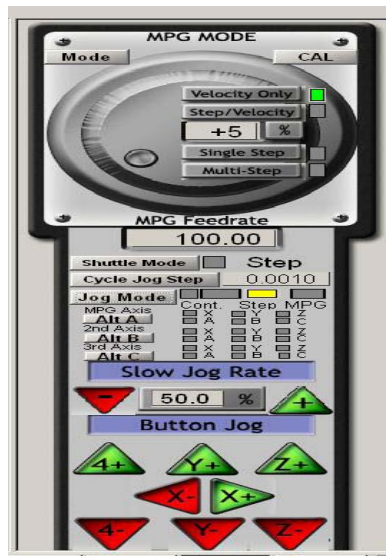


Figure II.2 capture de Jogging

b) Mode MDI

On utilise le mode MDI pour saisir manuellement le script G-code que nous voulons exécuter.

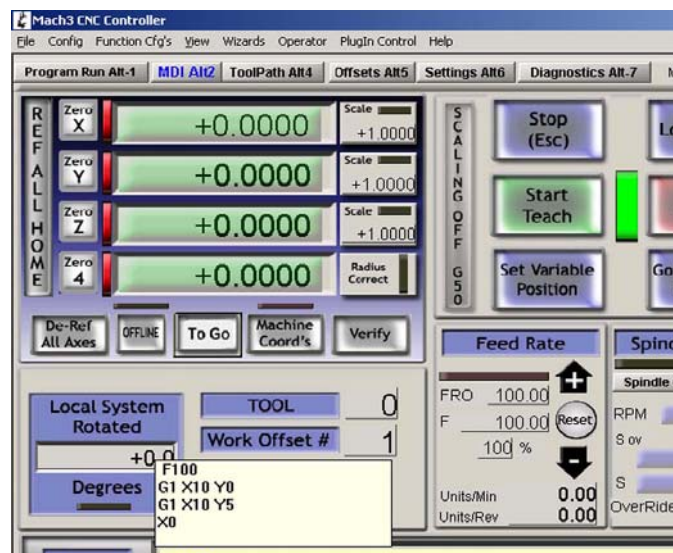


Figure II.3 capture de MDI

c) Mode automatique

Dans ce mode, nous chargeons directement le fichier G-code comportant le programme à exécuter.

II.1.4. Le port parallèle du PC

Le Mach3 utilise le port parallèle pour générer ou recevoir les signaux électriques. Le port parallèle comprend 12 broches en sortie et 5 broches en entrées. La figure (II.4) montre le sens des signaux de ce port.

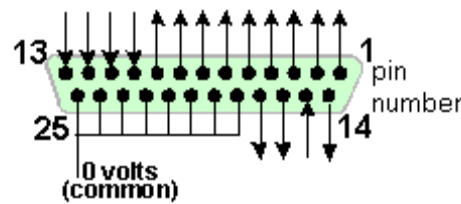


Figure II.4 port parallèle

II.1.5. Configuration de Mach3

II.1.5.1. Configuration initiale

La configuration des ports et des pins se fait de la manière suivante : (*Config>Ports and Pins*)

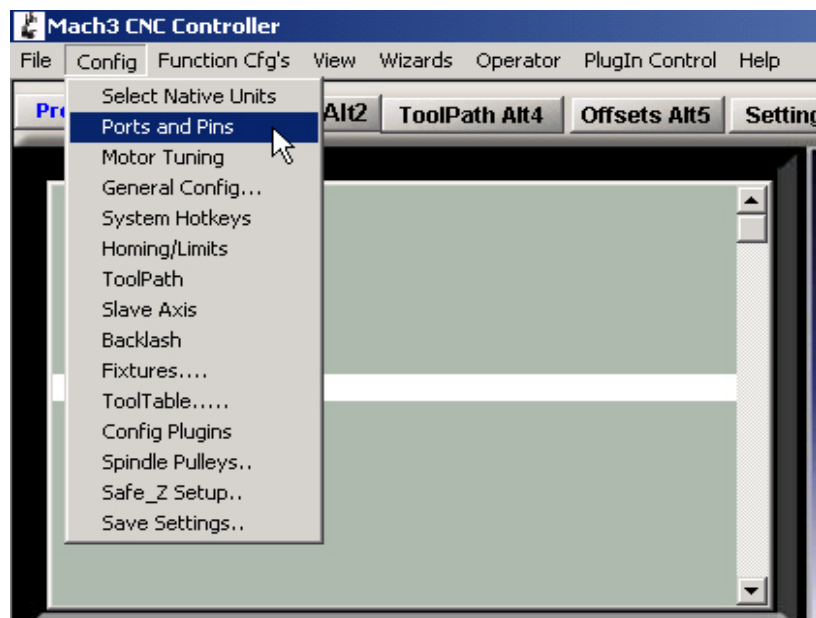


Figure II.5 menu Config.

a) Définition des adresses du Port (s)

Sélectionner *Port Setup and Axis Selection* dans *Ports and Pins*.

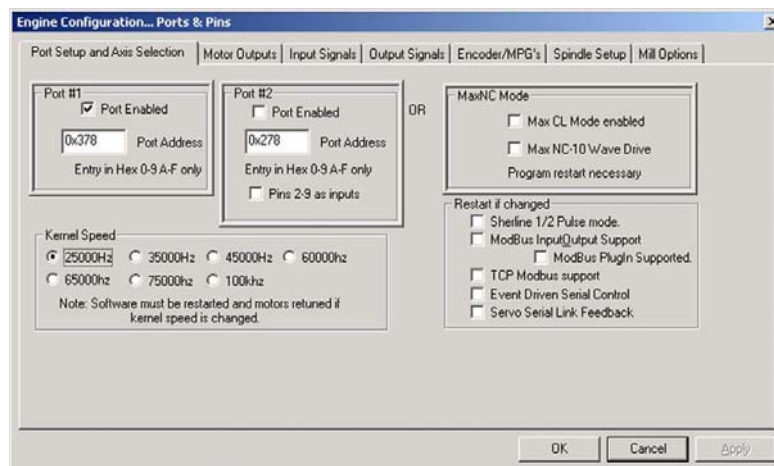


Figure II.6 sélection des ports.

L'adresse par défaut du Port 1 du PC est 0x378. Si nous utilisons un ou plusieurs *add-PCI* sur les cartes de port parallèle, alors nous aurons besoin de déterminer l'adresse relative à chacun des ports.

Choisir la vitesse du noyau

Sur le même onglet on peut sélectionner des fréquences de 25000 Hz à 100000 Hz. Ça dépend de la vitesse du processeur et autres charges lors de l'exécution Mach3. 25.000 Hz sera souvent adapté.

b) Signaux de sorties des axes

Dans l'onglet *Motor Outputs* du dialogue *Ports and Pins*, nous sélectionnons le brochage qui correspond à la machine.

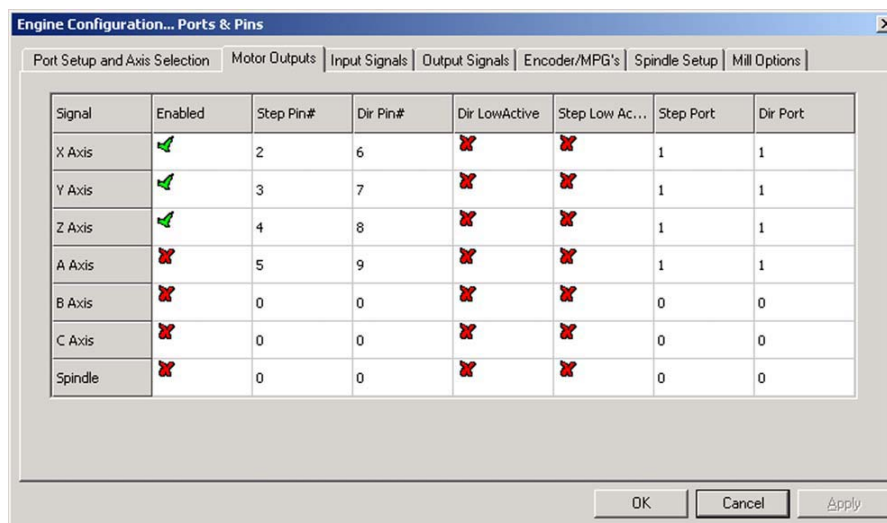


Figure II.7 menu de sorties au moteurs.

c) Signaux d'entrées

De la même manière, nous sélectionnons l'onglet *Input Signals* pour configurer les signaux d'entrées et Output signals pour configurer les signaux de sorties de notre machine comme il est montré sur les Figure II.8 et Figure II.9

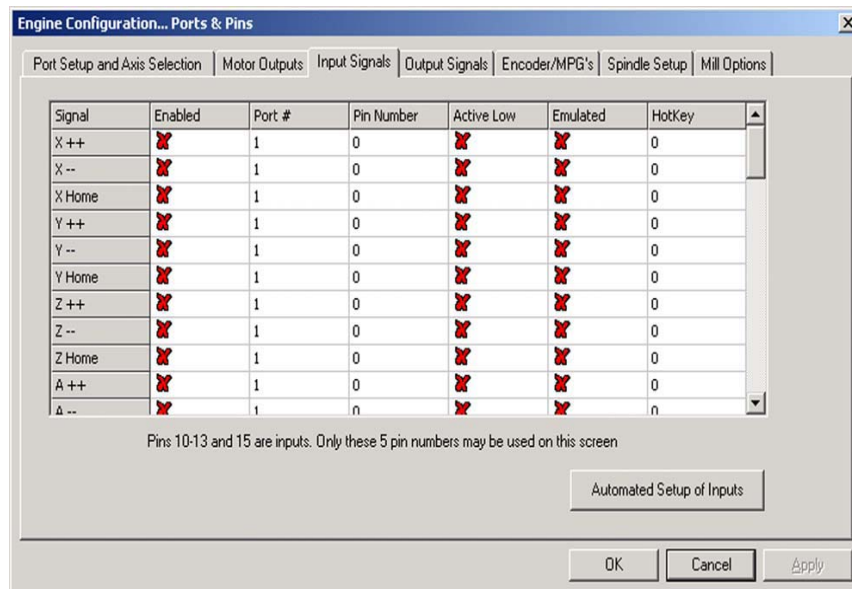


Figure II.8 signaux d'entrées.

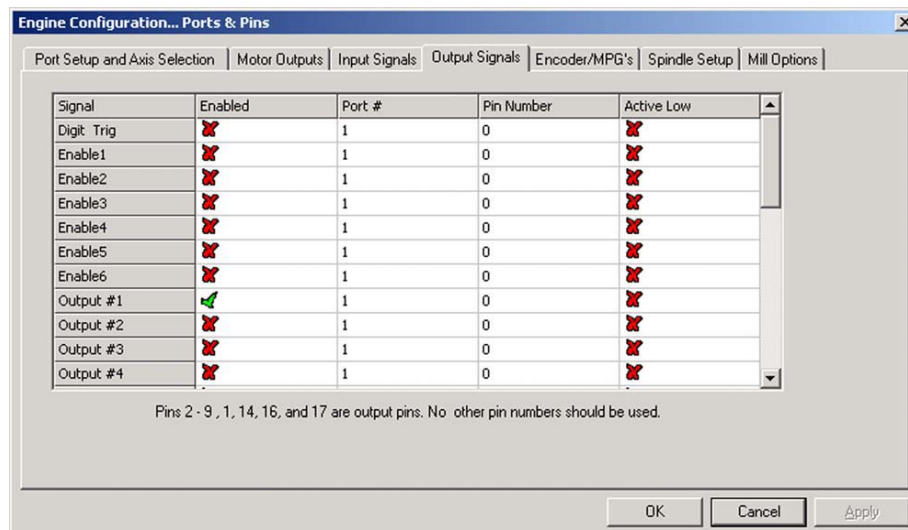


Figure II.9 onglet de signaux de sorties

f) Motor Tuning and Setup:

Dans cette fenêtre, nous indiquons la vitesse, l'accélération et la largeur des impulsions de chaque axe.

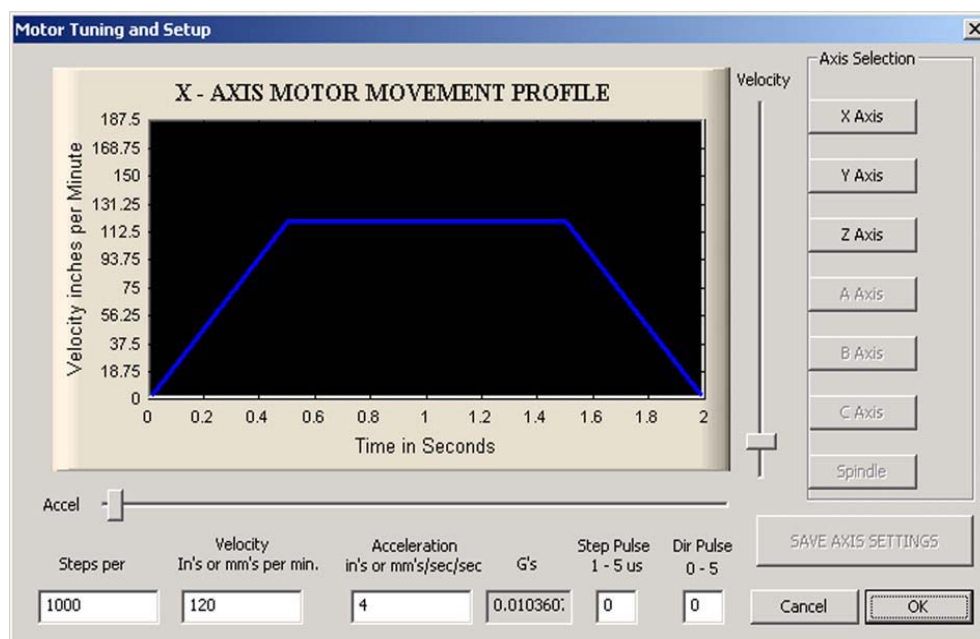


Figure II.10 capture de réglage du moteur

h) Configuration de la broche :

L'onglet *Config>Ports & Pins* est *Spindle Setup* est utilisé pour définir la façon dont la broche et le liquide de refroidissement doivent être contrôlé. Il est montré dans la figure II.11 :

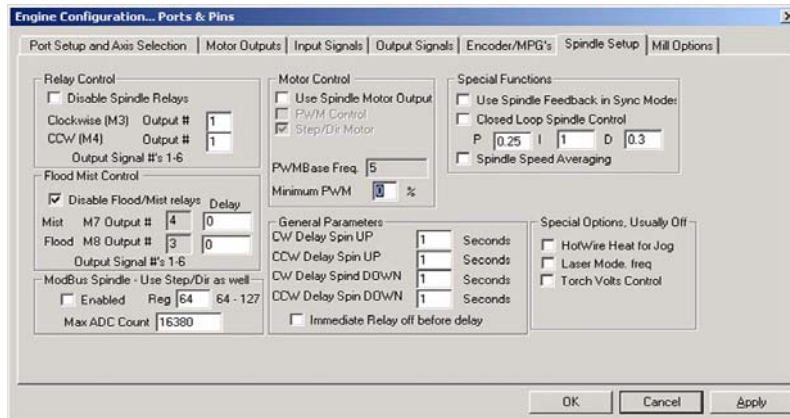


Figure II.11 configuration de la broche

II.2. Logiciel de commande EMC2 [II.3]

II.2.1. Introduction

EMC2 (contrôleur de la machine renforcée) a été développé à l'institut national des standards et des technologies des Etats Unis, le *NIST*. Il a mûri comme un logiciel fonctionnant sur le système d'exploitation Unix. EMC2 utilise le langage G-code. Ce qui signifie que si vous voulez usiner une pièce dessinée dans un programme graphique ou de dessin, vous devez également utiliser un programme capable de convertir le format du fichier de sortie de ce programme en un autre fichier dans le format reconnu et accepté par l'interpréteur d'EMC2 ce qui est fait par les logiciels CAM/CAD. EMC2 ne fonctionne pas sur les systèmes d'exploitation Microsoft (TM). La raison en est qu'EMC2 requiert un environnement temps réel pour son planificateur de mouvements et ses générateurs de pas. Il bénéficie aussi de l'excellente stabilité et des performances nettement meilleures du système d'exploitation Linux.

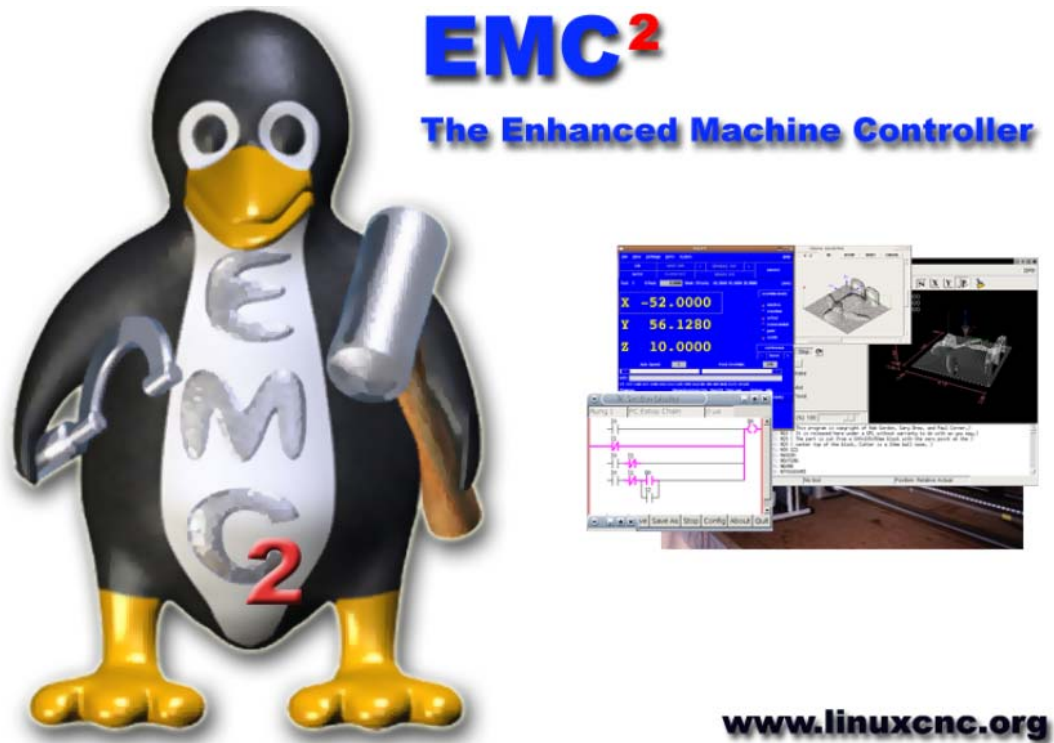


Figure II.12 EMC2.

II.2.2. Comment marche EMC2

EMC2 est capable de contrôler des machines-outils, des robots ou d'autres automatismes. Il est capable de contrôler des servomoteurs, des moteurs pas à pas, des relais ainsi que d'autres mécanismes relatifs aux machines-outils.

Il y a quatre composants dans le logiciel EMC2 :

- Un contrôleur de mouvement (EMCMOT).
- Un contrôleur d'entrées/sorties discrètes (EMCIO).
- Un coordinateur d'exécution des tâches (EMCTASK).
- Une bibliothèque d'interfaces utilisateur, tant graphiques que textuelles.

En plus, il y a une couche appelée HAL (la couche d'abstraction du matériel), qui permet la configuration simple d'EMC2 sans le besoin de le recompiler.

Modes opératoires

Il existe trois différents modes majeurs pour entrer des commandes sur EMC2, nous citons : Le mode Manuel, Automatique et MDI.

En **mode manuel**, chaque commande est entrée séparément. Ces commandes sont normalement contrôlées en pressant un bouton de l'interface graphique avec la souris ou en maintenant appuyée une touche du clavier.

En **mode automatique**, un bouton similaire ou l'appui d'une touche peuvent être utilisés pour charger ou lancer l'exécution complète d'un programme de G-codes stocké dans un fichier.

En **mode d'entrée de données manuelles (MDI)** l'opérateur peut saisir un bloc de codes et dire à la machine de l'exécuter en pressant la touche <Return> ou <Entrée> du clavier.

Certaines commandes de mouvement sont disponibles et produisent les mêmes effets dans tous les modes. Il s'agit des commandes ABORT, ESTOP et FEED RATE OVERRIDE. Ces commandes se dispensent d'explications.

II.2.3. Interface de l'utilisateur :

Il y a plusieurs interfaces utilisateur sur EMC2 comme : AXIS, TkEMC, MINI et KEYSTICK

L'interface graphique AXIS :

AXIS est une interface utilisateur graphique pour EMC2, elle offre un aperçu permanent du parcours de l'outil. Elle utilise *Tk* et *OpenGL* pour l'affichage de l'interface graphique.

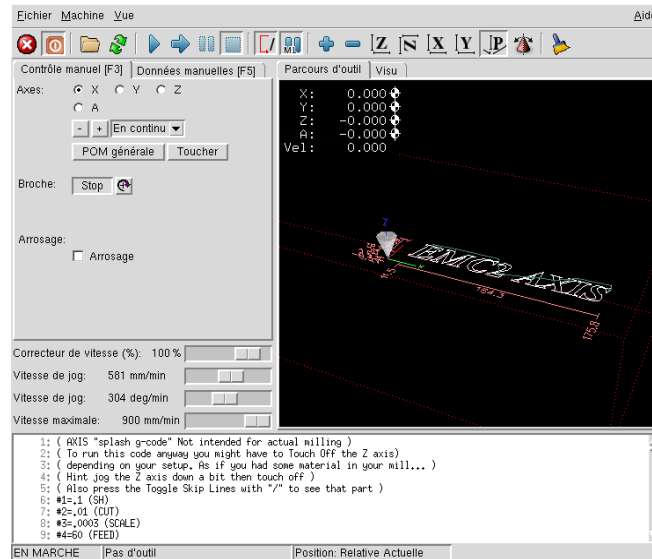


Figure II.13 interface axis

Pour choisir interface graphique AXIS, nous éditons le fichier *ini* et dans la section [DISPLAY] nous mettrons ceci :

DISPLAY = axis.

Puis, nous lançons emc2 et choisissons le fichier ini. La configuration simplifiée sim/axis.ini est déjà configurée pour utiliser AXIS comme interface.

II.2.4. Configuration d'EMC2 [II.4]

Avant de configurer EMC2 il faut savoir que ce logiciel nous offre des exemples préconfiguré que nous pouvons directement utiliser.

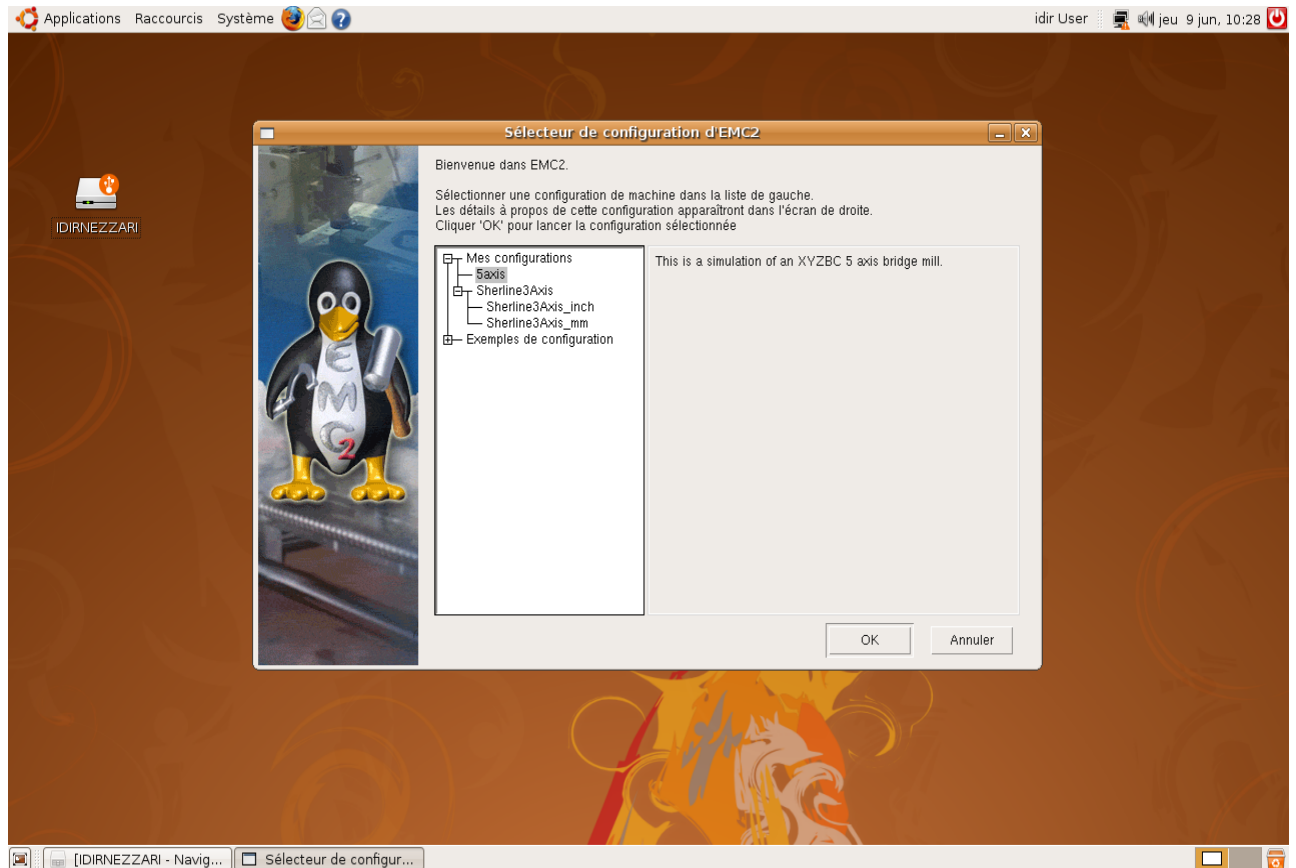


Figure II.14 exemples configuré

a) Test de latence

Ce test est le premier à effectuer sur un l'ordinateur pour savoir si celui-ci est capable de piloter une machine CNC. Le temps de latence est le temps nécessaire à l'ordinateur pour arrêter ce qu'il est en train de faire pour répondre à une requête externe. Dans notre cas, la requête est l'horloge qui sert de référence pour les impulsions de pas. Plus la latence est basse, plus l'horloge pourra être rapide et donc, plus rapides et plus douces seront les impulsions de pas.

b) Adresse des ports

Pour ceux qui construisent leur matériel, il est facile et économique d'augmenter le nombre d'entrées sorties d'un PC en lui ajoutant une carte PCI fournissant un ou deux ports parallèles supplémentaires. Faire suivre ces ports d'une couche d'opto-isolation est utile pour éviter les courts circuits pouvant détruire la carte, voir même toute la carte mère. EMC2 supporte un maximum de 8 ports parallèles.

c) Réglage du port parallèle

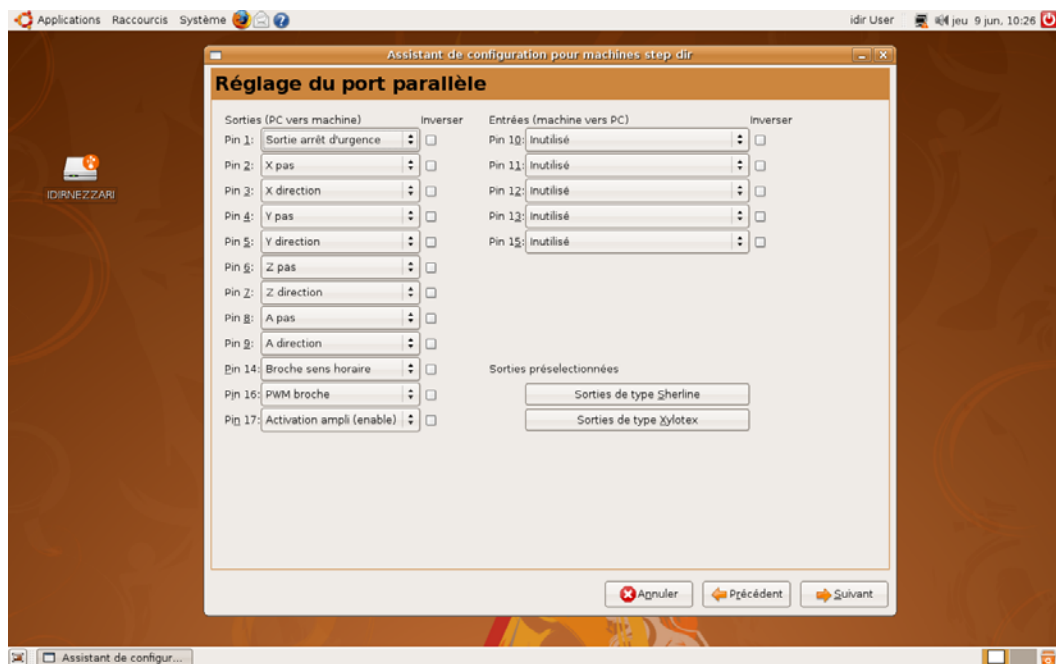


Figure II.15 réglage du port parallèle

Pour chacune des pins, choisir le signal qui correspond au brochage de votre port parallèle. Cochez la case inverser si le signal est en logique négative (0V pour vrai / actif, 5V pour faux / inactif).

Sorties présélectionnées : Réglage automatique des pins 2 à 9 selon le standard *Sherline* (Direction sur les pins 2, 4, 6, 8) ou selon le standard *Xylotex* (Direction sur les pins 3, 5, 7, 9).

d) Configuration des axes

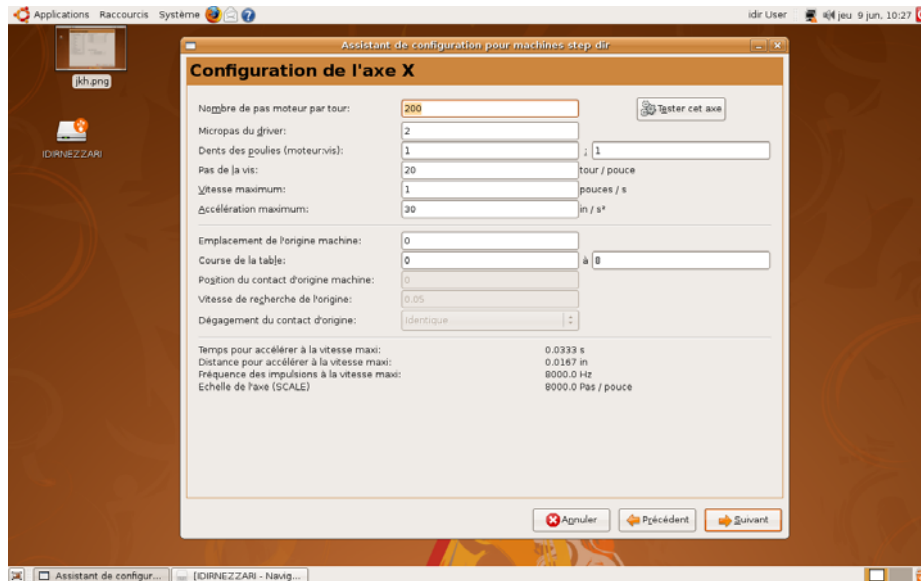


Figure II.16 page de configuration de l'axe X

Dans cette configuration on doit indiquer le nombre de pas du moteur, Micropas du pilote, dents des poulies, si la machine dispose de poulies entre le moteur et la vis, on va entrer ici le nombre de dents de chacune d'elles. Pour un entraînement direct 1:1, Pas de la vis, la vitesse et l'accélération max, et le parcourt.

e) Définir la vitesse et l'accélération d'avance.

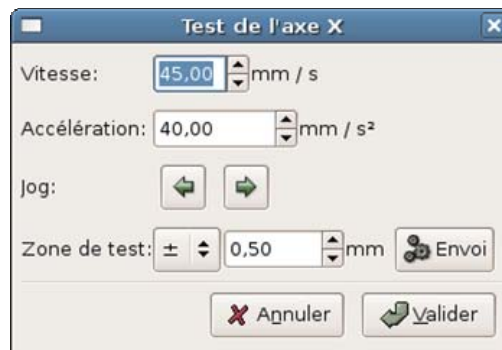


Figure II.17 définir la vitesse et l'accélération max

Avec *Stepconf* il est facile de définir certaines valeurs comme celles de l'accélération et de la vitesse. Il faut d'abord entrer les éléments corrects pour les Pas moteur, les Micropas, les Poulies, et le pas de la vis.

II.3. PID intégré sur EMC2 [II.3]

Le PID fournit une boucle de contrôle Proportionnel/Intégrale/Dérivée . Il est en temps réel. Le nombre maximum des boucles que générer indépendamment EMC2 est 16 boucles. La figure (II.18) donne le schéma fonctionnel d'une simple boucle de PID sur EMC2.

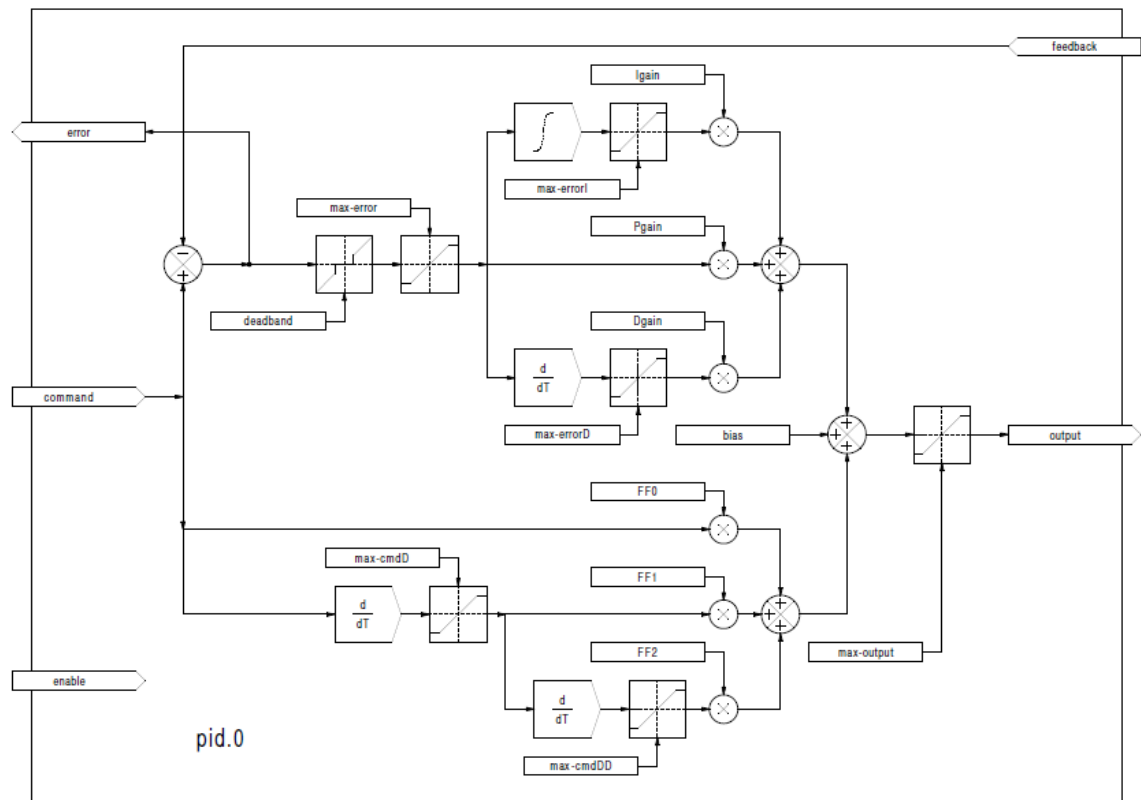


Figure II.18 schéma fonctionnel d'une boucle PID

II.4. La différence entre Mach3 et EMC2 :

Mach3	EMC2
1) système d'exploitation : Windows	1) système d'exploitation : Linux.
2) la version gratuite est limitée de 500 lignes de G-code.	2) toutes les versions sont gratuites et mise à jour disponible sur le net.
3) contrôle PID uniquement pour la broche.	3) le nombre de boucle PID est 16 boucles.
4) il n'y a pas de correction de la vitesse.	4) possibilité de faire la correction de la vitesse.
5) y a pas d'exemples préconfigurés.	5) offrir des exemples préconfigurés.

Tableau II.1 comparaison entre Mach3 et EMC2

II.5. Niveau des signaux logiques

Tous les signaux de sortie et d'entrée de Mach3 et EMC2 sont numériques binaires via le port parallèle. Les niveaux tension entre 0V à 0.8V sont considérées niveau bas 'Lo' et les tensions entre 2.4V à 5V sont des niveaux Haut 'Hi'.

II.5.1. Comment travail les signaux de pas :

Nous avons vu que les logiciels Mach3 et EMC2 génèrent les consignes destinées aux pilotes (cartes de commandes). Ces consignes sont représentées en deux signaux : STEP et DIR.

Chaque impulsion dans le signal STEP indique un déplacement d'un pas de l'outil.

Le sens de ce déplacement est défini par le signal DIR comme suit :

DIR = 0 → Sens 1

DIR = 1 → Sens 2

La figure suivante donne la forme du signal STEP.



Figure II.19 forme du signal STEP

Plus l'écart entre les impulsions n'est petit, plus la vitesse de déplacement des axes augmente.

II.6. Conclusion :

Dans ce chapitre nous avons pu manipuler les deux logiciels Mach3 et EMC2, en les configurant à nos besoins pour la conception de la carte de commande.

CHAPITRE III

Identification et régulation de système

III.1 Identification :

III.1.1 Introduction :

La modélisation a pour objectif d'attribuer un modèle mathématique (fonction de transfert) à un processus ou à un phénomène physique qu'on cherche à étudier ou simuler.

III.1.2 Modélisation et choix du moteur à courant continu :

III.1.2.1. Introduction :

L'évolution des technologies conduit à utiliser des machines nécessitant des mouvements très précises pour l'entraînement d'engins industriels, systèmes robotique ...etc.

Dans notre Project, on s'intéresse aux moteurs à courant continu à aimant permanent, car ils nous donnent plus de puissance et de précision par rapport au moteur pas à pas

III.1.2.2. Description d'un moteur à courant continu :

Un moteur à courant continu est une machine électrique. Il s'agit d'un convertisseur électromécanique permettant la conversion bidirectionnelle d'énergie entre une installation électrique parcourue par un courant continu et un dispositif mécanique, d'où l'énergie électrique est transformée en énergie mécanique. Un moteur électrique à courant continu est constitué :

D'un **stator** qui est à l'origine de la circulation d'un flux magnétique longitudinal fixe créé soit par des enroulements statorique (bobinage) soit par des aimants permanents. à l'arrière du stator, se trouve la partie porte balais et les balais assurant les contacts électriques avec le rotor. Il est aussi appelé inducteur.

D'un **rotor** bobiné relié à un collecteur rotatif inversant la polarité dans chaque enroulement rotorique au moins une fois par tour de façon à faire circuler un flux magnétique transversal en quadrature avec le flux statorique. Les enroulements rotoriques sont aussi appelés enroulements d'induit, ou communément induit, [III.1]

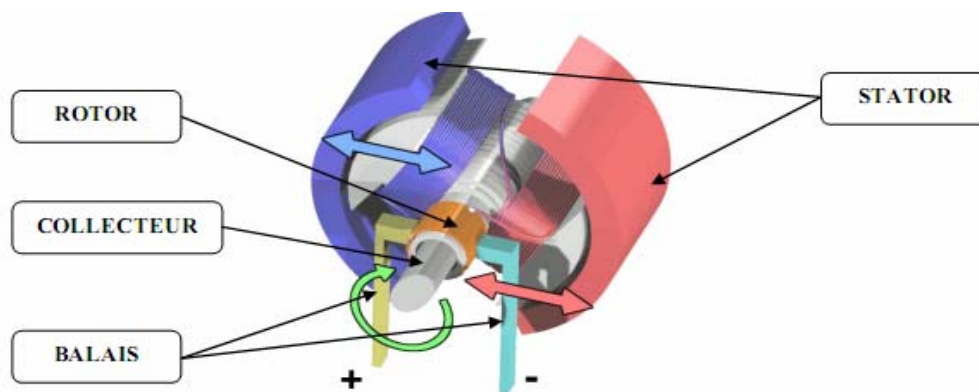


Figure III.1 : Schéma d'un moteur à courant continu

III.1.2.3. Modélisation mathématique :

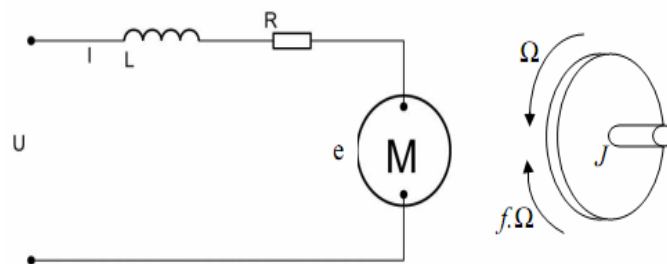


Figure III.2 : Schéma équivalent d'un moteur à courant continu

Selon le schéma de la (Figure III.2), un moteur électrique à courant continu est régi par les équations physiques découlant de ses caractéristiques électriques, mécaniques et magnétiques.

D'après les lois de Newton et de Kirchhoff, On peut écrire les équations différentielles de premiers ordres suivantes :

$$U(t) = R i(t) + L \frac{di(t)}{dt} + E(t) \quad (\text{III.1})$$

$$E(t) = K_e \Omega(t) \quad (\text{III.2})$$

D'après le principe fondamental de la dynamique on trouve :

$$J \frac{dn}{dt} = C_u - C_r$$

(III.3)

$$C_u = K_m i(t) - C_p$$

Où

U(t) : Tension appliquée au moteur

E(t) : Force contre électromotrice

i(t) : Intensité traversant le moteur

Ω(t): Vitesse de rotation du rotor

C_u : Couple moteur généré

C_r : Couple résistant

F : Coefficient de frottement visqueux

J : Moment d'inertie de l'axe du rotor

K_e : Constante de vitesse

K_m : Constante de couple

C_p : Couple de pertes

III.1.2.4. Fonction de transfert du moteur :

Dans le domaine s, on obtient :

$$R i(P) + P L i(P) + E = U(P) \quad (III.4)$$

$$E = K_e \Omega(P) \quad (III.5)$$

$$J P \Omega(P) = C_u - C_r \quad (III.6)$$

L'expression de Ω(p) devient:

$$\Omega(p) = \frac{K_m I(p) - C_p}{s + Jp} \quad (\text{III.7})$$

et

$$U(p) = \Omega(p) \frac{\left[(R + Lp)(F + Jp) \right]}{K_m} + K_e + \frac{(R + Lp)C_p}{s + Jp}$$

On suppose que le moment du couple de pertes (qui est vu comme une perturbation) est négligeable devant le moment du couple électromagnétique ($K_m i(t)$) on peut alors annuler C_p pour simplifier le système.

La fonction de transfert cherchée $F(p)$ est entre la tension entrante dans le moteur $U(p)$ et la vitesse de sortie $\Omega(p)$,

$$F(p) = \frac{\Omega(p)}{U(p)} = \frac{K_m}{(R + Lp)(F + Jp) + K_e K_m} \quad (\text{III.9})$$

La fonction de transfert en position devient alors :

$$H(p) = \frac{\theta(p)}{U(p)} = \frac{K_m}{(R + Lp)(F + Jp) + K_e K_m} \frac{1}{p} \quad (\text{III.10})$$

Le schéma bloc de la fonction de transfert en vitesse du moteur à CC est donné par la figure (figure III.3):

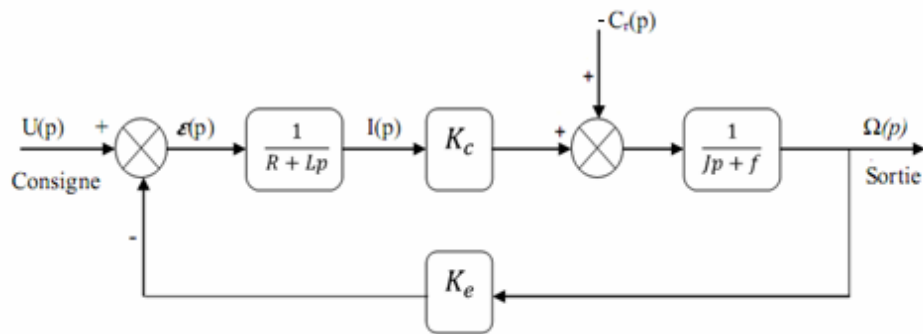


Figure III.3 : Schéma bloc du modèle de moteur électrique en vitesse

Les paramètres R, L, F, J, K_m et K_e sont propres au moteur et à la charge qui est connectée. Leur identification est nécessaire pour la suite de l'étude et au dimensionnement des correcteurs.

Il y a plusieurs méthodes qui permettent le calcul de ces paramètres.

III.1.3. Identification du modèle du moteur à cc :

III.1.3.1. Détermination expérimentale des paramètres du modèle :

Afin de déterminer les différents paramètres figurant dans la modélisation du processus, il est nécessaire de réaliser des essais mécaniques et électriques sur le système. Ces paramètres sont donc. La résistance d'induit R , La constante de fém K_e , ou de couple K_m , l'inductance de l'induit L , le moment d'inertie du rotor J et le coefficient de frottement visqueux f .

Les méthodes proposées ci-dessous s'appliquent essentiellement à une Machine à courant continu à excitation indépendante ou à aimants permanents.

A. Détermination de la résistance d'induit R et De la constante K

[III.2] :

A.1 Détermination de la résistance d'induit R seule :

En vue de simplifier les méthodes, on suppose dans cette partie que le courant d'induit I est parfaitement continu. Pour cela on a deux méthodes :

La première consiste à mesurer la résistance de l'induit à l'aide d'un ohmmètre branché aux bornes de l'induit.

Et la seconde méthode consiste à alimenter la machine sous une tension continue réduite et un courant parfaitement continu (par une alimentation stabilisée par exemple) et de bloquer son rotor $\Omega = 0$. Avec cette technique, et conformément à la relation électrique suivante :

$$U = E + R.I + L \frac{di(t)}{dt} = K.\Omega + R.I + L \frac{di(t)}{dt} \quad , = R.I$$

Une mesure de la tension d'alimentation (U) et du courant (I) permet rapidement de déterminer la valeur de la résistance de l'induit R.

A.2 Détermination de la constante K seule :

Une méthode permettant de déterminer la constante de fém. (K égale à la constante du couple si $E = K\Omega$ et $C_e = KI$) consiste à tracer la caractéristique $U = f(\Omega)$ pour la machine à courant continu à vide. En effet, en supposant un courant d'induit nul (RI

$+L \frac{di(t)}{dt} = 0$) la tension d'alimentation est directement proportionnelle à la vitesse angulaire de rotation

A.3 Détermination conjointe de R et K :

Une méthode précise qui permet de déterminer conjointement les valeurs de R et K consiste à réaliser 2 essais pour deux tensions d'induit (U_1 et U_2), et de mesurer les 2

vitesses de rotation (Ω_1 et Ω_2) ainsi que les 2 courants absorbés (I_1 et I_2). On obtient donc un système d'équations à 2 inconnues (K et R) tel que :

$$U_1 = K.\Omega_1 + R \text{ et } U_2 = K.\Omega_2 + R.I_2$$

Ce système d'équation a pour solution

$$K = U_1.I_2 - U_2.I_1 / I_2.\Omega_1 - I_1.\Omega_2$$

$$R = U_1.\Omega_2 - U_2.\Omega_1 / I_1.\Omega_2 - I_2.\Omega_1$$

B. Détermination de l'inductance d'induit L [III.2] :

La Méthode de la pente du courant d'induit permettant de déterminer L, Cette méthode consiste à alimenter la machine a courant continu par un hacheur série et de mesurer l'ondulation du courant d'induit ΔI . On s'assurera d'être dans le cadre d'une conduction continue du courant d'induit et on négligera l'influence de la résistance d'induit R.

Cette ondulation vérifie en effet la relation suivante :

$$\Delta I = I_{\max} - I_{\min} = U.\alpha \frac{(1 - \alpha)}{L F} \quad \text{Ou } \alpha \text{ est le rapport cyclique}$$

C. Détermination du coefficient de frottement F [III.3] :

Un essai de séparation des pertes permet de déterminer les pertes par frottement P_m , en effet P_m est déterminé en étudiant les variations de la puissance fournie au système en fonction d' E^2 et de E ,

Connaissant P_m et Ω on déduit le coefficient F tel que $P_m = F \Omega$

D. Détermination du moment d'inertie J [III.3] :

Le Constant de temps mécanique ($\tau_m = \frac{J}{C}$) peut être déterminée par un essai de ralentissement. Au ralentissement, la vitesse va prendre l'expression suivante :

$$\Omega(t) = \Omega_0 e^{-\frac{t}{\tau_m}} + \frac{C e}{C} (1 - e^{-\frac{t}{\tau_m}})$$

On définit la fonction $y(t) = \Omega_0 (1 - e^{-\frac{t}{\tau_m}})$, pour deux instants t_1 et t_2
 Tel que $t_2 = 2t_1$ on a :

$$\frac{y(t_1)}{y(t_2)} = \frac{1}{1 + e^{-\frac{t_1}{\tau_m}}}$$

Connaissant expérimentalement $\frac{y(t_1)}{y(t_2)}$ et l'instant t_1 , on introduit la constante de temps mécanique τ_m , le coefficient F étant connu, on détermine alors la constante J .

III.1.3.2. Identification de type « Boîte noire » du moteur à cc :

Dans un modèle de comportement, la détermination des paramètres se fait à partir de l'observation de la sortie, ces méthodes s'appuient sur l'analyse graphique des courbes expérimentales en quelques points particulières et ne tiennent pas compte de l'ensemble de mesure.

Le signal d'excitation le plus courant est l'échelon, l'identification se fait à partir de la réponse indicielle.

Le choix de la structure de modèle se fait à partir de la réponse comme suit :

- La réponse présente elle un retard?
- Le système est il stable ou de type intégrateur ?
- La réponse présente elle un dépassement ?

Suivant la réponse à ces questions, on choisira la structure du modèle et la méthode associée

III.1.3.2.1. système du premier ordre Boucle ouverte

- Pas de retard pur,
- Démarrage franc (à comparer aux systèmes d'ordre 2 ou supérieur).

Quand on applique une tension U_1 au moteur électrique DC de premier ordre, le moteur tourne avec une vitesse angulaire ω que l'on peut convertir en tension de sortie U_0 à l'aide d'un tachymètre, par conséquent la tension U_0 est proportionnelle à ω

En boucle ouverte on pourra représenter le système avec le schéma suivant :

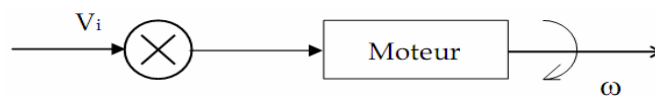


Figure III.4 : Schéma du moteur

D'après l'équation (II.9) et si on met en évidence les constantes de temps on obtiendra

$$\frac{W}{U} = \frac{K_m / R F}{\left(\frac{L}{R} P + 1 \right) \left(\frac{J}{F} P + 1 \right)}$$

Et comme L/R est très petit on peut approximer la fonction de transfert par un système du premier ordre :

$$F(P) = \frac{W}{U} = \frac{k}{(\tau_m P + 1)} \quad K = \frac{K_m}{R F} \quad \text{et} \quad \tau_m = \frac{J}{F}$$

(III.11)

Si on applique une tension échelon au moteur et on enregistre la sortie (la vitesse de l'arbre du moteur sous forme de tension) en fonction du temps ($U_0(t)$) on aura la courbe suivante :

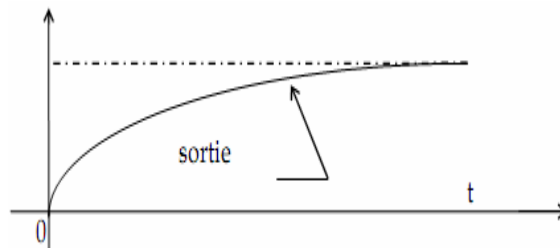


Figure III.5 : Courbe de la sortie $U_o(t)$

III.1.3.2.2. les méthodes de détermination τ et K

Pour déterminer K , il suffit de diviser la valeur finale (U_o) sur la valeur d'échelon appliquée.

Avec la courbe de sortie on peut déterminer la constante de temps du moteur.

- Le temps pour atteindre 63% de la sortie finale U_o en régime établi est la constante de temps τ .
- Une autre méthode pour déterminer la constante de temps avec la courbe de sortie à l'excitation échelon est de tracer la tangente avec la courbe à l'origine, cette tangente coupe la droite de la valeur finale à un point correspondant au temps $t = \tau$.
- la règle de 95% de la valeur finale: $t_r = \tau$
- la règle (Tr 5%) de la valeur finale: $Tr\ 5\% = 3\ \tau$
- $T_m = (t_{90\%} - t_{10\%})$ de la valeur finale $= 2.2\tau$

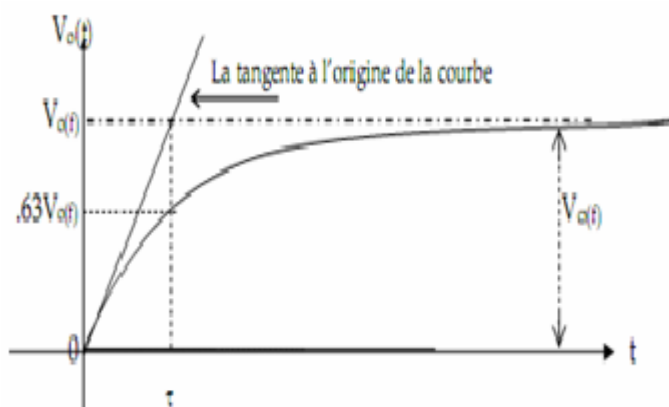


Figure III.6 : Calcul de τ

Remarque

En cas de bruit, la méthode de mesure de τ via l'intersection de la pente de la tangente au démarrage avec la valeur finale est meilleure que le relevé du temps à 63% de la valeur finale.

La méthode que nous avons utilisée pour l'identification de notre système est comme suit:

Quand on applique sur le moteur une tension (a) connue, la vitesse de rotation du moteur soit [III.4]:

$$w(s) = \frac{K}{\tau P + 1} \frac{a}{P} = \frac{K a}{P} - \frac{K a \tau}{\tau P + 1}$$

(III.12)

On applique le transformé inverse de Laplace on obtient:

$$W(P) = K a - K a e^{-\frac{t}{\tau}} \quad (III.13)$$

Pour $t \gg 0$, et comme τ_m et positive, le terme $e^{-\frac{t}{\tau}}$ tend vers zéro quand t tend vers l'infini

$$w(\infty) = K a$$

(III.14)

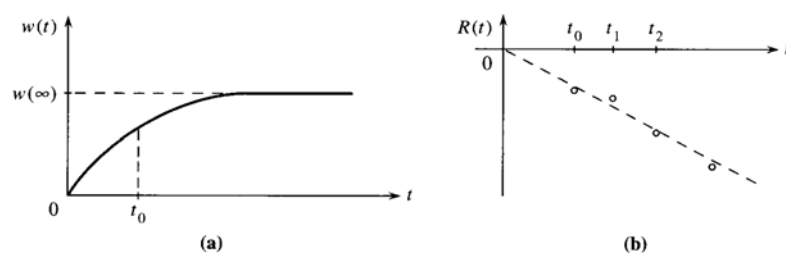


Figure III.7: (a) la réponse indicielle de moteur. (b) estimation de τ_m

Avec ces données on peut déterminer K_m comme suit:

$$K = \frac{W(\infty)}{a} \quad (\text{III.15})$$

Pour déterminer τ_m on applique :

(III.16)

On a depuis (II.15) :

$$e^{-\frac{t}{\tau}} = 1 - \frac{w(t)}{K a} = 1 - \frac{w(t)}{z(\infty)} \quad (\text{III.17})$$

On obtient donc :

$$\tau_m = \quad (\text{III.18})$$

III.1.4. Conclusion

Concernant le moteur, on a choisi un moteur à courant continu à excitation indépendante, car il est disponible avec un très faible coût, en plus il est convenable en grandeur de puissance avec notre machine, ainsi que la simplicité de son utilisation.

III.2 Régulation :

III.2.1 introduction :

Dans la plupart des processus industriels, il est indispensable de maîtriser certains paramètres physiques. En automatique lorsque l'on souhaite atteindre une certaine

vitesse, température, position, angle..., il est souvent très nécessaire d'avoir recours à un asservissement, c'est à dire un système capable d'atteindre et de maintenir une consigne en utilisant une mesure. Il s'agit donc d'un système bouclé, dont il reste à déterminer la fonction permettant de corriger la commande en fonction de la consigne initiale et de l'erreur mesurée.

Le PID (Proportionnel Intégral Dérivé) est une méthode qui a fait ses preuves et qui donne de bons résultats. On peut citer, parmi tant d'autres, quelques exemples concrets d'utilisation : asservissement des servomoteurs, régulation de réaction chimique en chimie pharmaceutique, convoyeurs industriels à navettes indexées, ...etc.

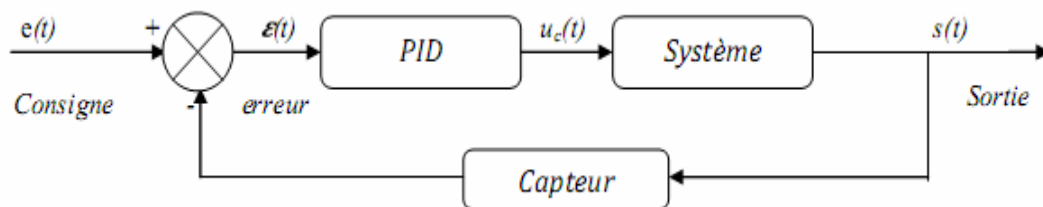


Figure. III 8 Schéma bloc d'un système avec correcteur PID (Boucle fermée)

III.2.2. L'objectif de la commande PID : [III.4]

Le régulateur PID est une simple action de retour d'information (Feedback). Par l'application des PID aux moteurs à courant continu, on cherche à faire coïncider à tout moment l'angle du rotor du moteur à une consigne qui se présente comme une entrée variable par l'utilisateur ou par le logiciel.

III.2.3. Asservissements et rôle des coefficients :

III.2.3.1. Asservissement P :

L'asservissement de type P est le plus simple qui soit. Il s'agit d'appliquer une correction proportionnelle à l'erreur corrigeant de manière instantanée tout écart de la grandeur à régler :

$$U_c(t) = K_p \cdot e(t) \quad \xrightarrow{L} \quad U_c(P) = K_p \cdot e(P)$$

(III. 20)

Son rôle est d'amplifier virtuellement l'erreur pour que le système réagisse plus vivement, comme si l'erreur était plus grande qu'elle ne l'est en réalité.

Il permet de vaincre les grandes inerties du système et diminue le temps de montée en donnant de la puissance au moteur (plus l'erreur est grande, plus on donne de puissance au moteur). Lorsque l'on augmente K_p , le système réagit plus vite et l'erreur statique s'en trouve améliorée, mais en contrepartie le système perd en stabilité. Le dépassement se fait de plus en plus grand, et le système peut même diverger dans le cas d'un K_p démesuré.

III.2.3.2. Asservissement PI :

L'asservissement de type PI est un asservissement de type P auquel on a ajouté un terme intégral

$$U_c(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau) d\tau \quad \xrightarrow{L} \quad U_c(p) = K_p \cdot e(p) + K_i e(p)/p = e(p)[K_p + K_i/p]$$

(III.21)

Le terme intégral complète l'action proportionnelle puisqu'il permet de compenser l'erreur statique et d'augmenter la précision en régime permanent. L'idée est d'intégrer l'erreur depuis le début et d'ajouter cette erreur à la consigne : lorsque l'on se rapproche de la valeur demandée, l'erreur devient de plus en plus faible. Le terme proportionnel n'agit plus mais le terme intégral subsiste et reste stable, ce qui maintient le moteur à la valeur demandée.

L'intégrale agissant comme un filtre sur le signal intégré, elle permet de diminuer l'impact des perturbations (bruit, parasites), et il en résulte alors un système plus stable. Malheureusement, un terme intégral trop important peut lui aussi entraîner un

dépassement de la commande, une stabilisation plus lente, voire même des oscillations divergentes.

III.2.3.3. Asservissement PID continu :

Les termes proportionnel et intégral peuvent amener un dépassement de la commande et des oscillations. Cela implique pour le moteur des inversions de polarité, ce qui est loin d'être idéal. Pour limiter ce phénomène indésirable, on introduit un troisième élément : le terme dérivé. Son action va dépendre du signe et de la vitesse de variation de l'erreur, et sera opposée à l'action proportionnelle. Elle devient prépondérante aux abords de la valeur demandée lorsque l'erreur devient faible, que l'action du terme proportionnel faiblit et que l'intégrale varie peu : elle freine alors le système, limitant le dépassement et diminuant le temps de stabilisation.

$$U_c(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau) d\tau + K_d \cdot \frac{de(t)}{dt}$$

$$U_c(P) = K_p \cdot e(P) + K_i \frac{e(P)}{P} + K_d P e(P) = e(P) \left[K_p + \frac{K_i}{P} + K_d P \right]. \quad (\text{III.23})$$

On obtient alors un asservissement composée d'un terme proportionnel, un terme intégral et un terme dérivé : on parle d'asservissement PID.

$$\text{PID} = K_p + \frac{K_i}{P} + K_d P = K_p \left(1 + \frac{T_i}{P} + T_d P \right) \quad (\text{III.24})$$

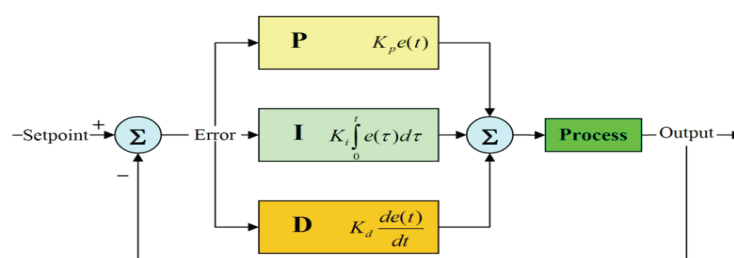


Figure .III.9 Régulateur PID continu

III.2.3.4. Résumé sur l'action des paramètres (coefficients) :

Après avoir expliqué précédemment le rôle des actions proportionnelle, intégrale et dérivé, on représente un résumé sur l'action des paramètres (coefficients) dans le tableau suivant :

Coefficients	Temps de montée	Temps De stabilisation	Dépassement	Erreur Statique
Kp	Diminue	Augmente	Augmente	Diminue
Ki	Diminue	Augmente	Augmente	Elimine
Kd	Chang. faible	Diminue	Diminue	Chang. faible

Tableau. III.1 Récapitulatif des paramètres PID

III.2.4. Réglage des coefficients [III.4] :

Le réglage d'un PID consiste à trouver les meilleurs coefficients Kp, Ki et Kd dans le but d'obtenir une réponse adéquate du procédé et de la régulation. L'objectif est d'être robuste, rapide et précis tout en limitant les dépassements.

III.2.4.1. Méthode de Ziegler-Nichols :

La méthode empirique de Ziegler-Nichols est très répandue dans l'industrie ou chez les automaticiens pour régler les correcteurs de ce type de chaîne. Elle a l'avantage de ne pas nécessiter de modélisation précise du système asservi, mais se contente d'essais expérimentaux, ce qui rend cette méthode très simple.

On fixe K_i et K_d à 0 et on augmente le gain du correcteur proportionnel pur K_p jusqu'à obtenir des oscillations persistantes (pour des moteurs, il s'agira plutôt d'une vibration autour de la position demandée). On mesure alors K_u , le gain du correcteur proportionnel à la juste oscillation, et T_u , la période des oscillations

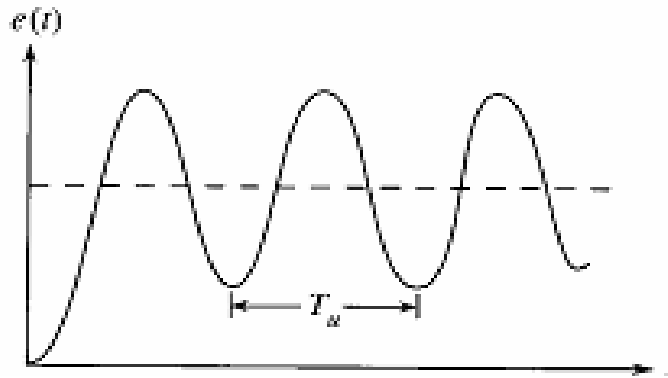


Figure. III.10 estimation de K_u et T_u

La méthode de Ziegler-Nichols indique alors des valeurs des coefficients permettant d'obtenir un bon compromis entre précision, rapidité et stabilité.

Contrôleur	K_p	T_i	T_d
P	$0.5 K_u$		
PI	$0.4 K_u$	$0.8 T_u$	
PID	$0.6 K_u$	$0.5 T_u$	$0.12 T_u$

Tableau III.2 méthode de Ziegler Nichols

Ensuite, on peut légèrement faire varier ces valeurs moyennes selon le compromis précision/rapidité/stabilité recherché. Dans le cas d'un asservissement en position, on aura tendance à monter un peu plus le gain intégral pour avoir une erreur statique faible, quitte à « Accoster » plus lentement à la valeur demandée.

III.2.5. Discrétisation de la commande PID analogique :

La commande PID analogique peut être transformée en numérique, après discrétisation (échantillonnage) avec une période d'échantillonnage T , cette commande est souvent simple à implémentée dans des microprocesseurs ou

microcontrôleurs par un algorithme embarqué (ex. programme en C/C++, assembleur...etc.). En effet la commande PID numérique est la plus utilisée de nos jours dans l'industrie pour des raisons de coût et de rapidité d'implémentation. Sa fonction de transfert en 'Z' est obtenue par discrétisation de la (III.4) du PID analogique.

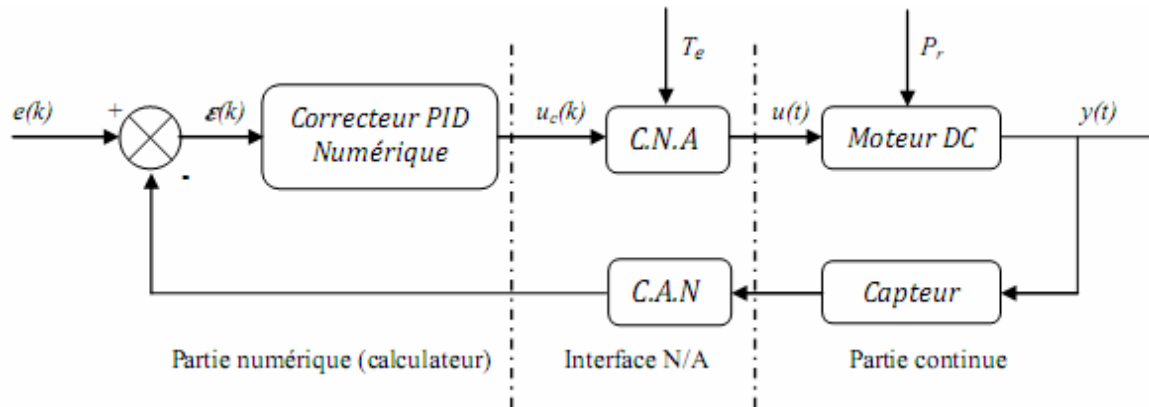


Figure. III.11 Schéma bloc d'un correcteur PID numérique

III.2.5.1 Asservissement en position avec un régulateur PID discret :

[III.4]

L'objectif de notre présente étude est l'implémentation des lois de commande synthétisées sur la machine CNC, cela nous oblige à discrétiser notre modèle et à faire un asservissement échantillonné en position des actionneurs, ce qui nous ramène par la suite à synthétiser un régulateur PID discret.

La fonction de transfert échantillonnée est donnée par la formule :

$$U_c(P) = K_p \cdot e(p) + K_i \frac{e(P)}{P} + K_d p e(p) \xrightarrow{TZ} U_c(Z) = K_P E(Z) + K_I \frac{E(Z)}{(1 - z^{-1})} + K_D E(Z) (1 - z^{-1})$$

(III.25)

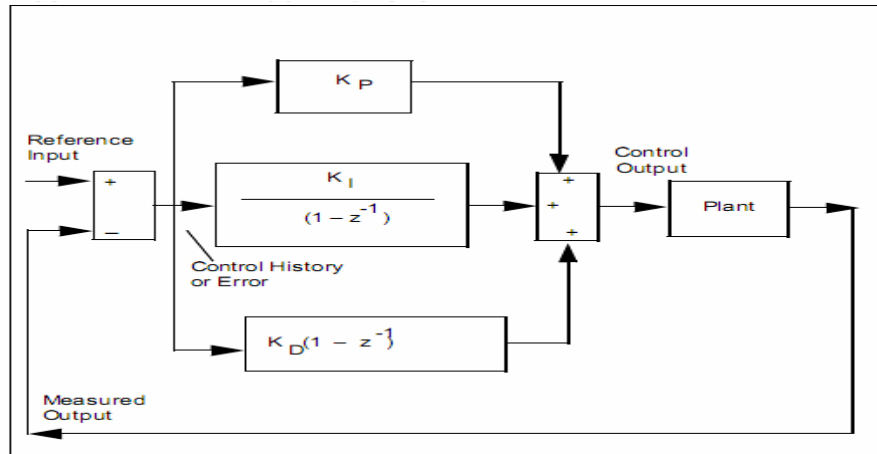


Figure .III.12 Régulateur PID discret

On peut écrire la fonction de transfert en Z du régulateur PID comme suit :

$$PID(Z) = K_P + K_I \frac{Z}{Z-1} + K_D \frac{Z-1}{Z}$$

(III.26) on obtient :

$$PID(Z) = \frac{b_2 Z^2 + b_1 Z + b_0}{Z(Z-1)}$$

(III.27)

$$b_2 = K_P + K_I K_D$$

$$b_1 = -(K_P + 2K_D)$$

$$b_0 = K_D$$

Pour établir l'algorithme à implémenter sur microcontrôleur, on doit élaborer la somme de manière récursive en introduisant une variable auxiliaire $X[K]$ définie comme suit :

$$X[K] = K_I \sum_{i=0}^{K-1} E[i]$$

On obtient ainsi :

$$X[K+1] = X[K] + K_I E[K]$$

À l'aide de cette variable auxiliaire l'expression du régulateur PID devient :

$$U[K] = X[K] + K_{pid} E[K] - K_D E[K-1]$$

(III.28)

Où

$$K_{pid} = K_P + K_I + K_D.$$

III.2.5.2 Synthèse du régulateur PID discret : [III.4]

On peut trouver les paramètres du régulateur PID discret à partir de ceux du PID continu déjà synthétisé en faisant l'approximation suivante de S :

$$S = \frac{1 - Z^{-1}}{T} = \frac{Z - 1}{Tz}$$

(III.29)

Où T est la période d'échantillonnage.

En introduisant cette approximation dans (III.9) et après simplification on trouve l'expression suivante du régulateur PID :

$$U_c(Z) = K_p \left[1 - \frac{T}{T_i} + \frac{T}{T_i} \frac{1}{1 - Z^{-1}} + \frac{T_d (1 - Z^{-1})}{T} \right]$$

(III.30)

Cette fonction de transfert doit correspondre à celle du régulateur PID continu définie dans (III.4). Ainsi, après comparaison des deux expressions on déduit les expressions des paramètres du régulateur PID discret suivantes

$$K_P = K_p \left(1 - \frac{T}{2T_i} \right)$$

$$K_I = \frac{K_p T}{T_i}$$

(III.31)

$$K_D = \frac{K_p T_d}{T}$$

III.3. Conclusion :

Le PID discret qui est sous forme d'un algorithme récursif nécessite une très grande puissance de calcul, cela nous mène à utiliser les circuits dédiés au traitement du signal numérique DSP (Digital Signal Processor) qui possèdent des architectures dédiées aux calculs récursifs intenses. Le chapitre suivant traite et analyse le DSP que nous allons utiliser dans notre projet, c'est le dsPIC30F2010 de Microchip.

CHAPITRE IV

Etude du dsPIC30F2010

IV.1 Introduction

L'élément central dans notre carte de commande est le dsPIC ® DSC [IV.1] [IV.2] qui est le cerveau de la carte de commande. Il dispose plusieurs fonctionnalités, c'est lui qui va exécuter l'algorithme de contrôle PID. Ce chapitre présente une analyse de ce circuit et de quelques de ses périphériques qui vont servir dans ce travail comme les modules PWM et QEI.

IV.2 contrôleur de signaux numériques dsPIC ® DSC :

Quand il s'agit de la mise en œuvre d'une application de commande, on va atteindre les performances requises si on utilise une plate-forme matérielle qui est spécifiquement conçu pour soutenir les tâches que nous devons accomplir.

Heureusement, une nouvelle classe de processeurs connue sous le nom du contrôleur de signaux numériques ou DSC « digital signal controller » a été mise au point récemment qui utilise les performances de traitement mathématique d'un puissant processeur de signal numérique avec le comportement très déterministe de microcontrôleurs standard.

C'est le dsPIC ® DSC de Microchip, qui intègre une énorme quantité de fonctionnalités dans une seule puce, permettant aux concepteurs de créer des solutions robustes de détection et de contrôle dans un paquet très petit. Nous allons utiliser le DSC dsPIC dans notre application pour gérer la commande des moteurs de la machine CNC.

La figure (IV.1) nous montre les périphériques associés de certaines puces dspic30F

Device	Pins	Program Mem. Bytes/Instructions	SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/Std PWM	Motor Control PWM	A/D 10-bit 500 Ksps	Quad Enc	UART	SPI™	I ² C™	CAN
dsPIC30F2010	28	12K/4K	512	1024	3	4	2	5 ch	16 ch	Yes	1	1	1	–
dsPIC30F3010	28	24K/8K	1024	1024	5	4	2	5 ch	16 ch	Yes	1	1	1	–
dsPIC30F4012	28	48K/16K	2048	1024	5	4	2	6 ch	16 ch	Yes	1	1	1	1
dsPIC30F3011	40/44	24K/8K	1024	1024	5	4	4	6 ch	19 ch	Yes	2	1	1	–
dsPIC30F4011	40/44	48K/16K	2048	1024	5	4	4	6 ch	19 ch	Yes	2	1	1	1
dsPIC30F6016	64	66K/22K	2048	1024	6	4	4	8 ch	16 ch	Yes	1	2	1	1
dsPIC30F6010	60	144K/48K	8192	4096	5	8	8	8 ch	16 ch	Yes	2	2	1	2

Figure IV.1 : dsPIC30F famille de la Conversion de la puissance et du contrôle de moteur
Le DSC dsPIC peut être percé et traité de plusieurs façons, mais trois points de vue sont particulièrement utiles :

- Analyse de l'architecture de traitement des données du circuit.
- Etude du jeu d'instructions et des opérations mathématiques que la puce supporte.
- Etude des divers périphériques du circuit.

Nous allons dans ce projet concentrer l'étude sur le circuit nous contons utiliser, qui est le dsPIC30F2010, qui a pratiquement tous les périphériques que nous avons besoin.

28-Pin SDIP and SOIC

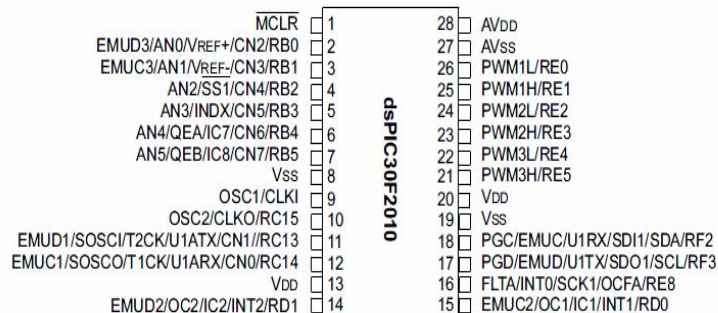


Figure IV.2 : dspic302010

IV.3 L'architecture de traitement des données de dsPIC ® DSC

La famille dsPIC est spécialement conçue pour le traitement très rapide des opérations mathématiques intensives comme les convolutions et les équations récursives aux différences. Ceci revient à la puissance leur architecture qui possède des opérateurs mathématiques niveau bas qui manipulent les nombres fractionnels, en plus elle possède deux modes d'adressages dédiés aux traitements DSP qui sont :

- L'adressage circulaire pour la mise à jour des variables intermédiaires de la récurrence.
- L'adressage en bit-reversed conçu spécialement pour le calcul de la FFT.

IV.3.1 mémoire de DSC dsPIC

Le DSC dsPIC30F2010 emploie l'architecture Harvard modifiée de qui possède une mémoire programme et mémoire données totalement séparées, cela permet au processeur d'être plus souple et plus efficace lors du traitement de l'information.

Cette architecture supporte trois bus:

Un pour aller chercher les instructions (le bus de mémoire-programme) et deux pour aller chercher les valeurs de données associées (souvent appelée le X- et le Y-bus mémoire). Ceci permet un fonctionnement en vrai single cycle, doublant ainsi le débit du système par rapport à une « architecture pure de Harvard » en cours d'exécution à la même vitesse.

Les figures (IV.3, IV.4) illustrent la différence entre une architecture pure de Harvard et une architecture Harvard modifiée.

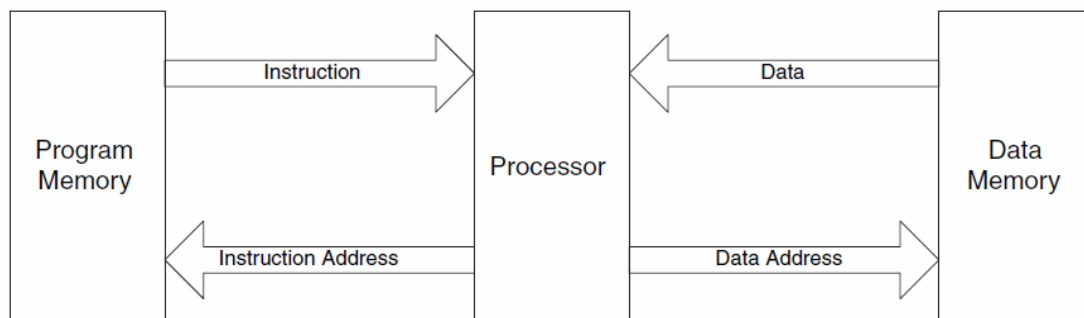


Figure IV.3 : architecture pure de Harvard

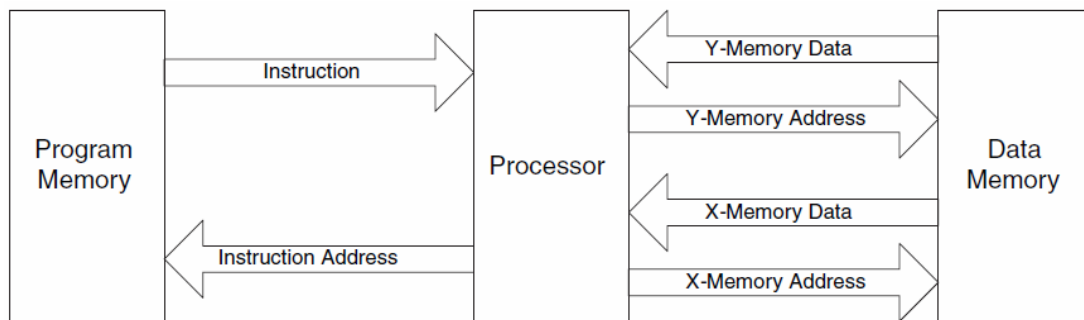


Figure IV.4 : architecture modifiée de Harvard

La mémoire du dsPIC30F2010 se compose de trois grandes catégories:

Registre de fonctionnement spécialisé ou (SFR), la RAM statique (SRAM), et la mémoire-programme qui est mappée dans l'espace mémoire de données.

Comme indiqué précédemment, le DSC dsPIC supporte à la fois un espace du programme et un espace de données, l'espace de données est divisée en deux espaces X-Y et cela pour le support de certaines instructions.

IV.3.1.1 registre de fonctionnement spécialisé «Special Function Registers» (SFR)

Le dsPIC ® DSC emploie des registres mappés en mémoire pour configurer, contrôler et surveiller les différents aspects du fonctionnement de périphériques, Connu sous le nom « **Special Function Registers** », ou SFR, ces registres résident dans les premières 2 Ko de la mémoire de données.

Contrairement aux emplacements de mémoire standard de données qui sont utilisés pour le stockage général, les données dans les SFR sont à mappage binaire, de sorte que l'écriture ou même la lecture de certains endroits permet au dsPIC ® DSC de prendre une action particulière. Par exemple, la transmission d'un octet de données via le port série de la puce exige de configurer les paramètres de fonctionnement du port série en écrivant les données appropriées à SFR du port série dans un ordre spécifique.

De même, l'application permet de vérifier si de nouvelles données ont été reçues en interrogeant le Registre d'état(SFR) du port série.

IV.3.1.2 Statique RAM (SRAM)

Contrairement à SFR, la lecture et l'écriture à la RAM n'affecte pas des aspects du fonctionnement de la puce de dsPIC

Les applications utilisent la section SRAM pour stocker des données qui changent à la demande de l'application en cours d'exécution, par exemple, les variables utilisées pour le filtrage ou des tampons qui contiennent les données reçues de la communication.

IV.3.1.3 Mémoire de programme

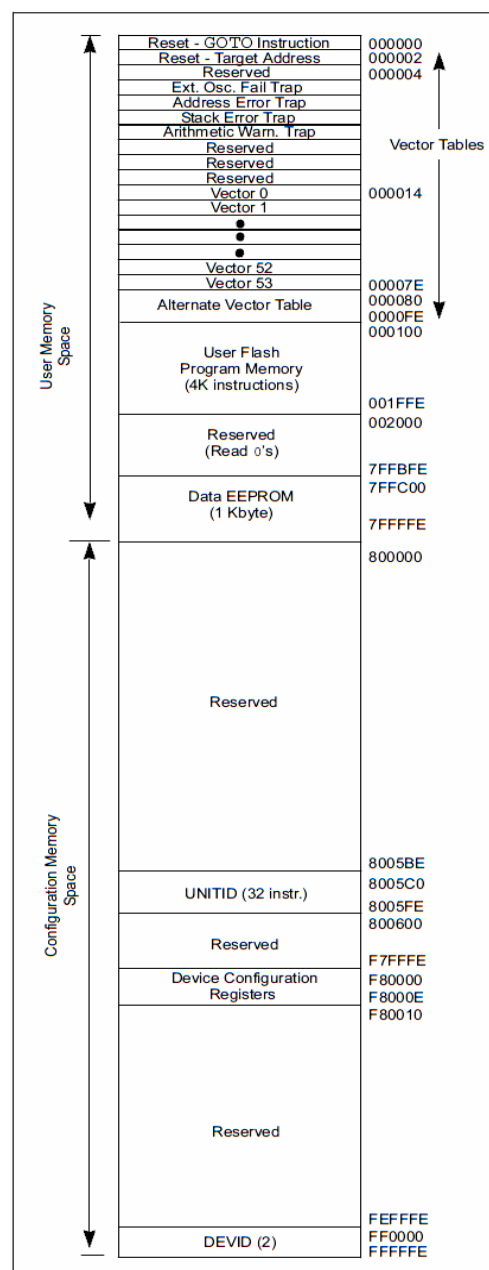
Le mémoire de programme se compose de plus de 4M mots de 24-bits, mais cet espace n'est pas totalement disponible à l'utilisateur.

La mémoire de programme elle-même est divisée en deux sous-espaces « espace mémoire de l'utilisateur » (000000H-7FFFFFFEH) et « espace mémoire de configuration » (800000H-FFFFFFEH) comme le montre la figure (IV.5).

L'application peut accéder l'espace mémoire d'utilisateur uniquement, sauf lorsque on utilise les instructions (TBLRD), (TBLWT) pour lire et écrire l'ID de périphérique, l'ID d'utilisateur, et pour configurer certains bits de la configuration des périphériques dans la mémoire de configuration.

L'espace mémoire de configuration est principalement réservé à utiliser par Microchip pour des buts de test et de stocker certaines informations utiles d'identification des périphériques.

La base de mémoire (adresse 0) contient une instruction GOTO suivi de l'adresse de démarrage de l'application (à l'adresse



2). Cette séquence de deux mots est exécutée chaque fois que le processeur est réinitialisé, et il permet au processeur à sauter au début du code de démarrage pour lancer l'application.

Le tableau des Vecteurs d'interruption contient les adresses des routines à exécuter lorsqu'une interruption survient.

Le bloc de mémoire de programme à partir de 100H-FFEh contient la mémoire flash de l'utilisateur, qui peut aller jusqu'à 4K instructions (12 KB).

Figure IV.5 : mémoire de programme de dspic30F2010

IV.3.2 Modèle du programmeur

le modèle du programmeur est illustré à la figure (IV.6), il se compose de registres de travail 16x16 bits (W0 à W15), 2 accumulateurs de 40 bits (ACCA et ACCB), registre d'état (SR), « Data Table Page register » (TBLPAG), « Program Space Visibility Page register » (PSVPAG), les registres DO et REPEAT (DOSTART, DOEND, BDNB et RCOUNT), et compteur de programme (PC).

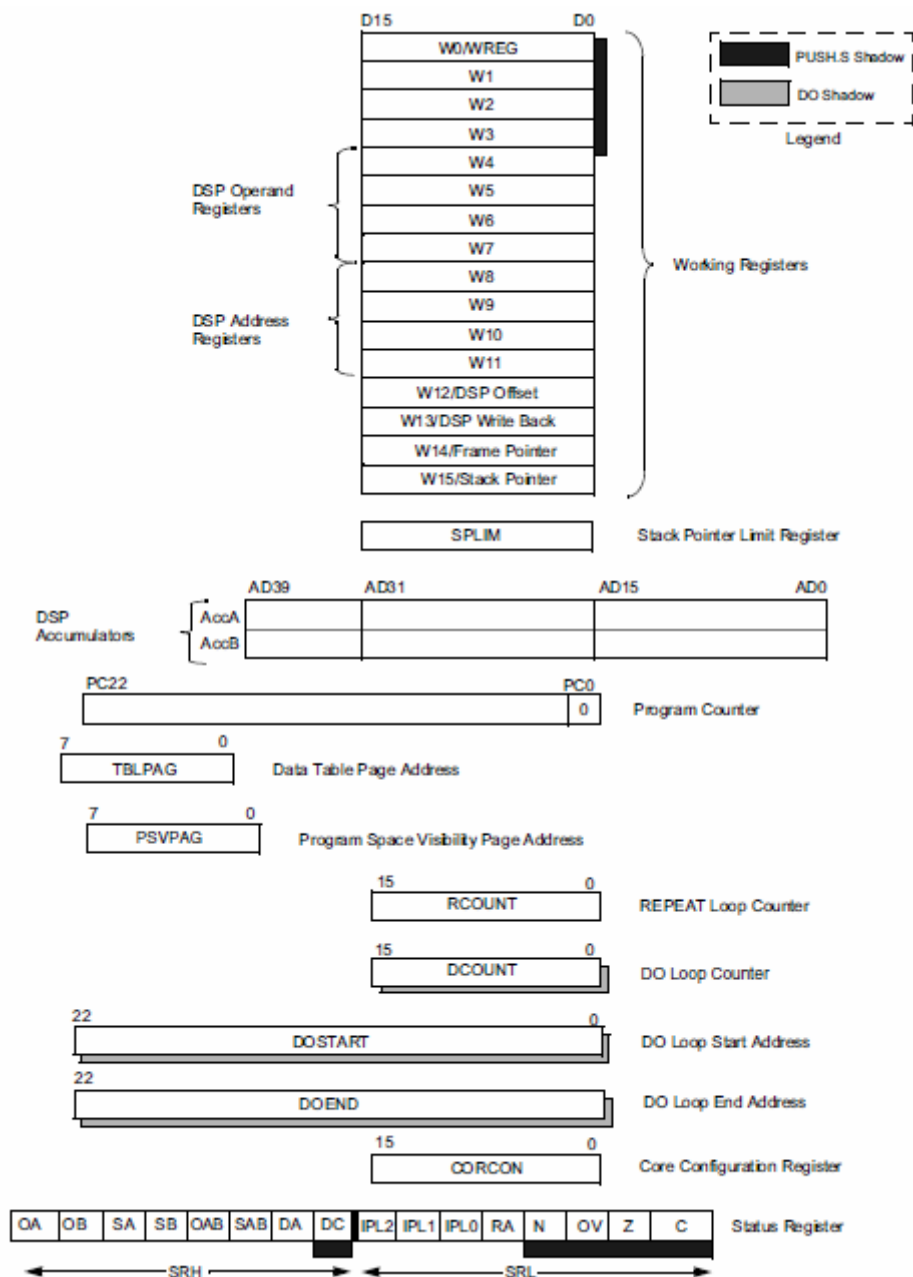


Figure IV.6 : modèle de programmeur de dspic30F2010

Les registres de travail peuvent agir comme des registres de données, d'adresses ou d'offset. Tous les registres sont mappés en mémoire.

Certains de ces registres possèdent des registres d'ombre, comme le montre la Figure (3-6). Le registre d'ombre est utilisé comme un registre tampon provisoire et peut échanger son contenu avec son registre original lors de la survenance d'un événement. Aucun des registres sont accessibles directement à l'ombre.

IV.3.3 Représentation des données numériques

Le DSC dsPIC peut fonctionner soit avec la représentation fractionnelle ou entière. L'entier Signé utilise la représentation en complément à deux dans lequel le bit le plus significatif (MSB) est le bit de signe et chaque bit suivant est une puissance de deux, comme le montre la figure (IV.7).

-2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
-----------	----------	----------	----------	----------	----------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

Figure IV.7 : représentation d'un nombre entier Signé en complément à deux

Un nombre entier de N bits en complément à deux peuvent représenter des valeurs comprises entre (-2^{N-1}) et $(2^{N-1}-1)$.

un nombre entier de 16-bit en complément à deux peut prendre n'importe quelle valeur comprise entre $(-32\,768)$ ($0x8000$) à $32\,767$ ($0x7FFF$)), tandis qu'un certain nombre 32-bit peut varier de (-2147483648) ($0x80000000$) à $2,147,43,647$ ($0x7FFFFFFF$)).

Lorsqu'on travaille avec les données entières en complément à deux, il est important de s'assurer que le bit de signe est manipulé correctement lorsque l'addition ou la soustraction des chiffres avec des largeurs différents, ou des erreurs de calcul peuvent se produire.

Bien que la manipulation des données entières est très rapide, elle souffre de la tendance pour les multiplications à déborder assez facilement, ce qui exige normalement que l'application comporte un logiciel supplémentaire pour vérifier et de gérer le débordement pour éviter des résultats erronés.

En multipliant un nombre entier de M bits par un nombre entier de N bits donne une valeur $(N + M)$ bits. Il ne prend pas trop de multiplications à déborder.

Une façon d'éviter le problème de débordement est à utiliser l'arithmétique fractionnelle, dans laquelle toutes les valeurs sont limitées à la gamme de $(-1 \text{ et } 1)$.

Dans ce cas, les multiplications sont garanties pour rester dans la même gamme, ce qui élimine le problème de débordement.

Parce que cette approche est assez fréquente, le dsPIC ® DSC peut être configuré pour effectuer des opérations arithmétiques avec les nombre fractionnel signé en utilisant le format QN.

Dans le format QN, les données sont représentées en tant que fractionnel en complément à deux dont le MSB est défini comme étant le bit du signe, et la virgule est située immédiatement après le bit de signe.

Le "N" dans la notation QN précise simplement le nombre total de bits dans le nombre. Un certain nombre Q16 utilise 16 bits pour représenter les valeurs, tandis qu'un certain nombre Q32 utilise 32 bits. Ou un autre format (N-1), avec le terme "(N-1)" représente le nombre de bits utilisés pour représenter la mantisse (la partie fractionnaire du nombre). En utilisant cette notation, un certain nombre Q16 serait (1,15), et un certain nombre Q32 serait (1,31).

La figure () ci-dessous montre un exemple de la représentation fractionnelle Q16 ou (1,15).

-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
--------	----------	----------	----------	----------	----------	----------	----------	----------	----------	-----------	-----------	-----------	-----------	-----------	-----------

Figure IV.8 : représentation d'un nombre fractionnel Signé en Q(15)

Une valeur signé QN peut représenter des nombres dans la plage (-1 à 1)

Un nombre fractionnelle Q 16 bits peut représenter des valeurs comprises entre (-1 et 0,999969482421875), tandis que un nombre Q32 peut représenter des valeurs comprises entre (-1 et $(1 - 4,66 \times 10^{-10})$).

Multiplication d'une fraction M.N par une fraction R.S produit un résultat de (M + R). (N + S). Tandis que la multiplication de deux valeurs (1,15), alors, le résultat est un nombre (2,30). Ceci est analogue à la situation de multiplier une fraction décimale avec N chiffres à la droite de la virgule par une deuxième fraction décimale avec S chiffres à la droite de la virgule décimale, le résultat est N + S chiffres à le droit de la virgule décimal.

IV.3.4 L'architecture de CPU

Le CPU a un mot d'instruction de 24-bits. Le compteur de programme (PC) est de largeur de 23 bits avec le moins significatif (LS) peu toujours nul, et le plus significatif (MS) bit est ignoré lors de l'exécution normale du programme, sauf pour certaines instructions spécialisées. Ainsi, le PC peut gérer jusqu'à 4M mots d'instruction de l'espace programme de l'utilisateur.

Un mécanisme de pré-apporter l'instruction est utilisé pour aider à maintenir le débit.

Le tableau de registre de travail est composé de registres de 16x16 bits, dont chacun peut agir en tant que données, d'adresses ou de registres de « offset ».

L'espace de données est de 64 k-octets (32K mots) et est divisé en deux blocs, dénommé X et Y mémoire de données. Chaque bloc a sa propre unité de génération d'adresses indépendantes (AGU).

La plupart des instructions fonctionne uniquement avec l'AGU de la mémoire X, qui donne l'apparence d'un espace unique de données unifiée.

Les instructions de classe multiplication-accumulation (MAC) de double source DSP fonctionnent à la fois par l'X et Y (AGUS),

Chaque mot de données se compose de 2 octets, et la plupart des instructions peuvent adresser les données soit en tant que mots ou octets.

Pour la plupart des instructions, le CPU est capable d'exécuter un ensemble de données (ou données de programme) lecture en mémoire, lire un registre de travail (données), écrire en mémoire de données et lire en mémoire programme (instruction) en un seul cycle d'instruction.

En conséquence, les instructions de 3 opérandes sont pris en charge, permettant d'exécuter des opérations de la forme $C = A + B$ en un seul cycle.

Un moteur DSP a été inclus afin d'améliorer considérablement la capacité de l'arithmétique de base et le débit. Il dispose de 17 bits par 17-bit multiplicateur d'une grande vitesse, un 40-bit ALU, deux accumulateurs de 40-bit et un barillet bidirectionnel de 40-bit.

Les données de l'accumulateur ou de tout registre de travail peuvent être déplacées jusqu'à 15 bits droite ou 16 bits à gauche en un seul cycle.

Les instructions DSP fonctionnent de façon transparente avec toutes les autres instructions et ont été conçus pour une performance optimale en temps réel.

La classe des instructions MAC peut chercher deux opérandes de données en même temps, tout en multipliant les deux registres W.

La plupart des instructions s'exécutent en un seul cycle, avec certaines exceptions.

Le CPU dispose d'une structure de traitement guidé pour les « traps » et les interruptions, avec 62 vecteurs indépendants. Les exceptions comprennent jusqu'à 8 « traps » (dont 4 sont réservés) et 54 interruptions.

Chaque interruption a une priorité associée par l'utilisateur (entre 1 et 7) (1 étant la priorité la plus basse et 7 étant le plus élevé) en conjonction avec un prédéterminée «ordre naturel». Les « traps » ont des priorités fixées, allant de 8 à 15.

IV.3.5 Le coprocesseur DSP

Le dsPIC intègre un puissant moteur DSP pour réaliser des instructions dites (MAC) pour les opérations qui sont la base de nombreux algorithmes de traitement du signal.

Seules certaines instructions peuvent utiliser le moteur DSP, et les exécuter avec des restrictions sur les sources des données qu'ils traitent.

Mais les restrictions sont minimales et permettent au moteur de lire deux opérandes, effectuer une opération MAC sur eux, puis les stocker et Retour à la mémoire, en un seul cycle le plus souvent.

Pour obtenir cette rapidité, le concepteur du dsPIC a mis en place du matériel dédié, y compris:

un 17-bits x 17 bits fractionnaires / multiplicateur entier,

deux accumulateurs de 40-bits,

un barillet 40-étape qui peut monter à 16 bits à gauche ou à droite dans l'instruction unique et double.

unités de génération d'adresse (AGUS) qui permettent de calculer le modulo d'adressage (deux AGUS) et peu-inversé d'adressage (une AGU).une utilisation Correcte de ces sous-systèmes matériels réduit considérablement le traitement requis pour la mise en œuvre des algorithmes complexes de traitement du signal.

IV.4. Les périphériques de dsPIC30F2010

Le dsPIC DSC contient plusieurs périphériques « timer, input capture, output compare, motor control PWM, quadrature encoder interface(QEI), A/D converter,

UART, serial peripheral interface(SPI), inter-integrated circuit(I2C), data converter interface(DCI),... »

IV.4.1 Contrôleur de moteur pwm

IV.4.1.1 Introduction

Le module PWM (MCPWM) simplifie la tâche de génération des sorties synchronisées de PWM.

Le module PWM a les caractéristiques suivantes:

base de temps dédié de résolution TCY / 2 PWM

Deux broches de sortie pour chaque générateur PWM

Fonctionnement complémentaires ou indépendant de chaque paire de broches de sortie des générateurs Hardware.

Temps mort pour le mode complémentaire.

Modes de fonctionnement :

mode du bord aligné « Edge aligned mode »

mode de centre aligné « Centre aligned mode »

mode de centre aligné avec actualisation double « Center aligned mode with double update ».

mode d'évènement singulier « Single event mode ».

Réglage manuel de broches de sortie de PWM

Matériel d'entrée de défaut avec la fonction programmable

Chaque broche de sortie associée avec le PWM peut être activée individuellement

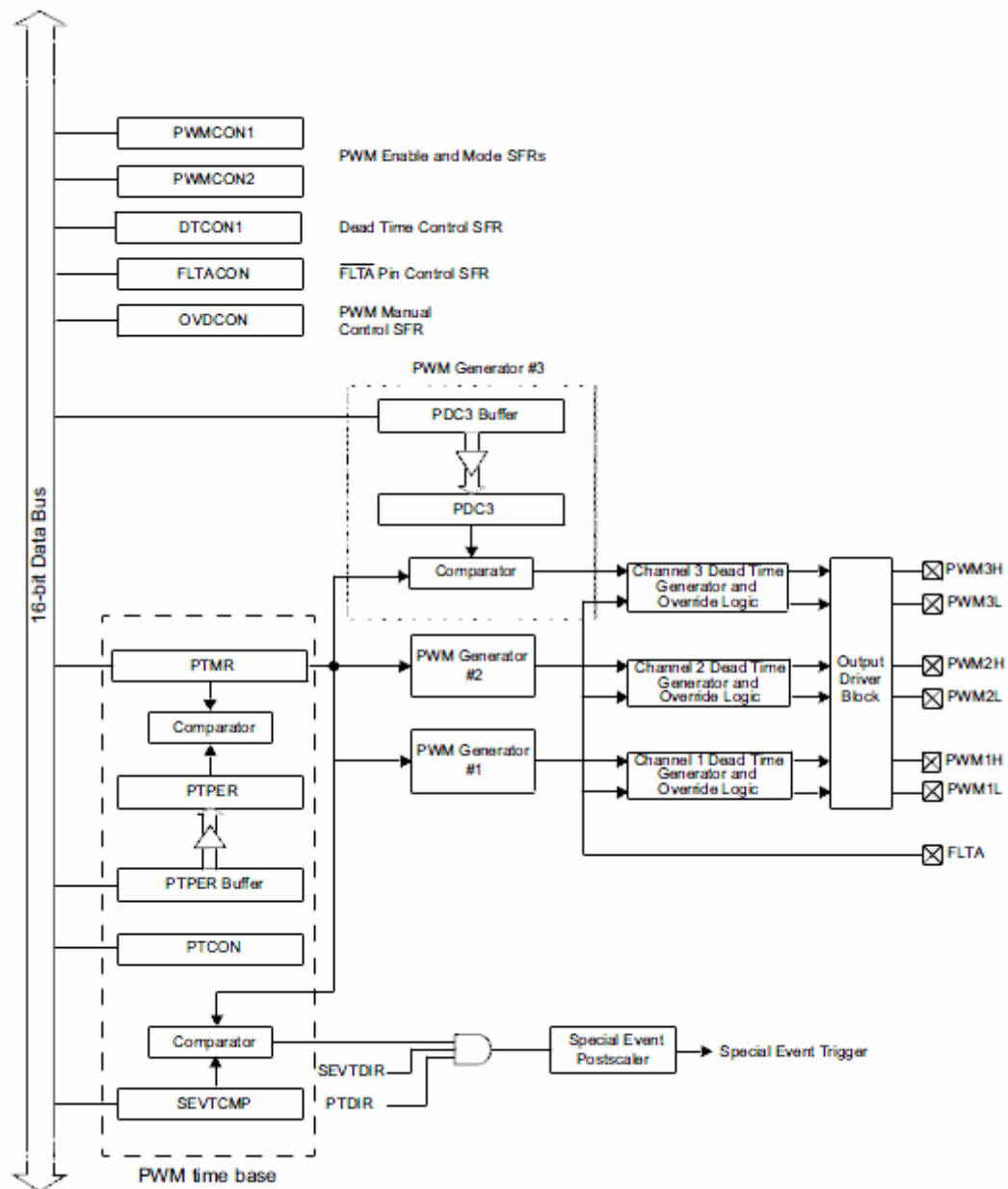


Figure IV.9 : schéma bloc du module PWM

IV.4.1.2 Registres de contrôle (SFR) du module PWM

Les registres suivants contrôlent le fonctionnement du module MCPWM:

- PTMR: PWM registre de base de temps
- PTPER: PWM registre de Période de base de temps
- PDC1: PWM registre de rapport cyclique
- PTCN • SEVTCMP • PWMCON2 • DTCON1 • DTCON2 • FLTACON • FLTBCON.

La base de temps PWM est assurée par un Timer de 15-bit avec un diviseur et « postscaler ». Les 15 bits de la base de temps sont accessibles via le registre PTMR.

La base de temps PWM peut être configurée pour quatre différents modes de fonctionnement:

Mode de fonctionnement libre «Free Running mode ».

Mode d'évènement singulier « Single Event mode ».

Mode du centre alignée «Centre aligned mode».

Mode du centre alignée avec double actualisation «Center aligned Mode with double update».

**** mode de fonctionnement libre**

Dans ce mode, la base de temps (le registre PTMR) comptera vers le haut jusqu'à ce qu'elle arrive à la valeur stockée dans le registre PTPER.

Le registre PTMR est remis à zéro après le prochain front d'horloge et la base de temps pourra continuer de compter vers le haut aussi longtemps que le bit PTEN reste activé.

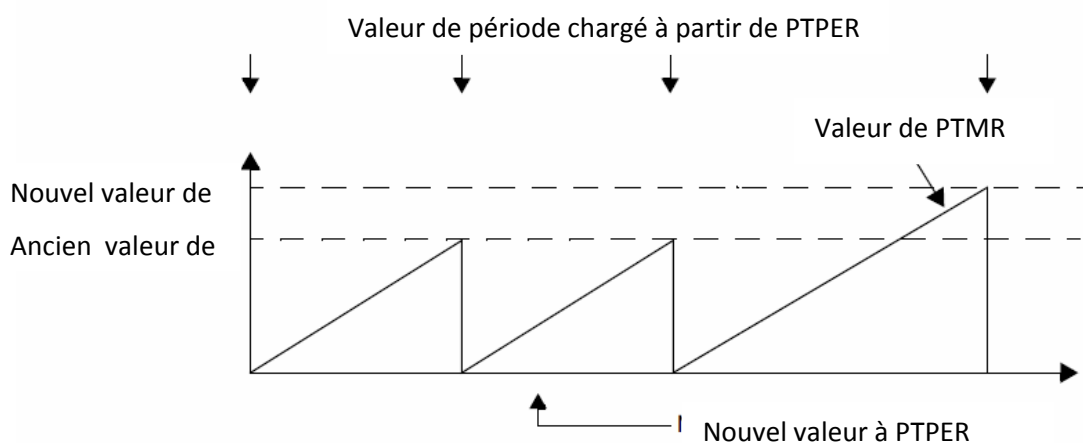


Figure IV.10 : actualisation de la période de PWM en mode de fonctionnement libre

La période de PWM peut être déterminée par la formule suivante :

$$PTPER = \frac{FCY}{F_{PWM} \cdot (PTMR \text{ Prescaler})} - 1$$

Example:

FCY = 20 MHz

F_{PWM} = 20,000 Hz

PTMR Prescaler = 1:1

$$PTPER = \frac{20,000,000}{20,000 \cdot 1} - 1$$

$$= 1000 - 1$$

$$= 999$$

Figure IV.11 : calcul de la période de PWM

Les signaux PWM du bord alignés sont produits par le module lorsque la base de temps de PWM fonctionne en mode (fonctionnement libre).

Le signal de sortie pour un canal PWM donnée dispose d'un délai spécifié par la valeur chargée en PTPER et un cycle de service spécifiée par le registre approprié PDCx.

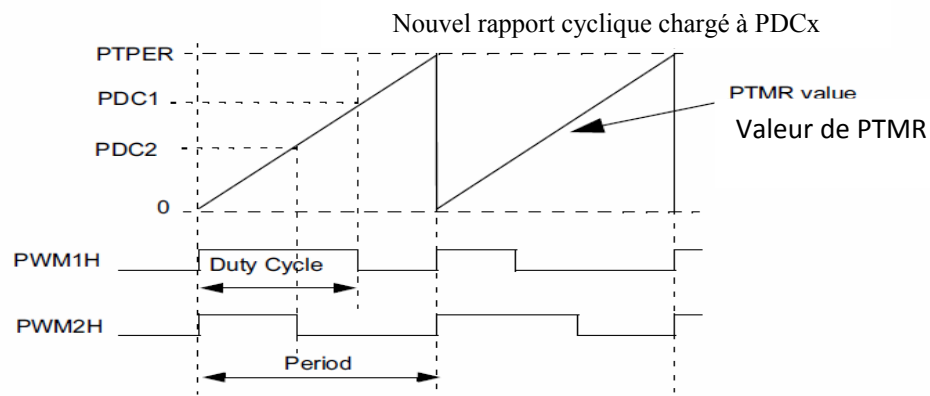


Figure IV.12 : signaux PWM du bord alignés

Le mode complémentaire est sélectionné pour chaque pair de broche E/S PWM.

Les broches de module PWM sont réglées en mode complémentaires par défaut lors de réinitialisation du circuit. Le temps mort est automatiquement activé lorsque l'une des broches paires de PWM I / O opère en mode complémentaire.

IV.4.2 Interface de l'Encodeur en quadrature (QEI)

IV.4.2.1 Introduction

Présentation de fonctionnement des codeurs incrémentaux :

Les codeurs incrémentaux ou autre codeurs optiques sont utilisés dans la détection de la position et de la vitesse des systèmes de mouvement de rotation.

Les codeurs incrémentaux permettent un contrôle en boucle fermée de nombreuses applications de commande des moteurs, comme les moteurs à Réluctance (SR) et les moteurs à induction CA (ACIM).

Un codeur incrémental typique comprend une roue à fentes fixée à l'arbre du moteur et un module émetteur / détecteur pour détecter les fentes de la roue.

En général, l'encodeur délivre trois sorties, appelées: phase A, phase B et INDEX, fournissent des informations qui peuvent être décodés pour fournir des informations sur le mouvement de l'arbre moteur, y compris la distance et la direction. Les deux canaux, la phase A (QEA) et la phase B (QEB), ont une relation unique.

Si la phase A conduit la phase B, donc la direction (du moteur) est considérée comme positif ou vers l'avant. Si la phase A est en retard de la phase B, donc la direction (du moteur) est négative ou inverse. Un troisième canal, appelé impulsion de référence INDEX, a lieu une fois par tour et est utilisé comme référence pour établir une position absolue.

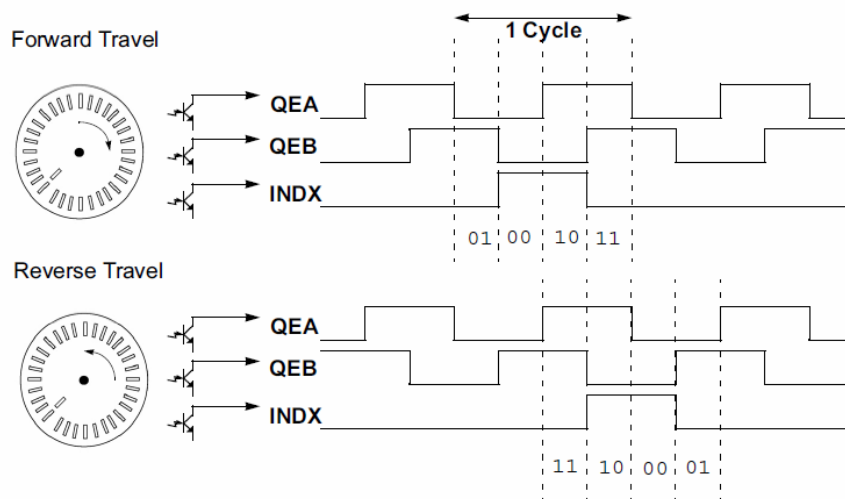


Figure IV.13 : interface de signaux de l'encodeur en quadrature

Les signaux en quadrature produits par le codeur peuvent avoir uniquement quatre états.

Notez que l'ordre des Etats est inversé lorsque la direction d'envoi est changée.

Un décodeur capte les signaux en quadrature QEA et QEB et l'impulsion INDEX et convertit les informations en un compte numérique de position. En général, le nombre augmente dans un sens de rotation de l'arbre et décrémente dans l'autre sens.

Le module (Interface de codeur en quadrature (QEI)) est une interface pour les codeurs incrémentaux. Le QEI se compose de logique de décodage de quadrature pour interpréter la phase A et B et un compteur/décompteur pour accumuler le nombre de pas et un Filtre numérique pour filtrer les signaux d'entrées.

IV.4.2.2 Registres de contrôle du module QEI

- QEICON ce registre permet de contrôler le fonctionnement de QEI et contient les drapeaux d'état indiquant l'état du module.
- DFLTCON ce registre permet de contrôler l'opération d'entrée du filtre numérique.
- POSCNT cet emplacement permet la lecture et l'écriture de la position 16-bit.
- MAXCNT ce registre contient une valeur qui sera comparée à la POSCNT dans certaines opérations.

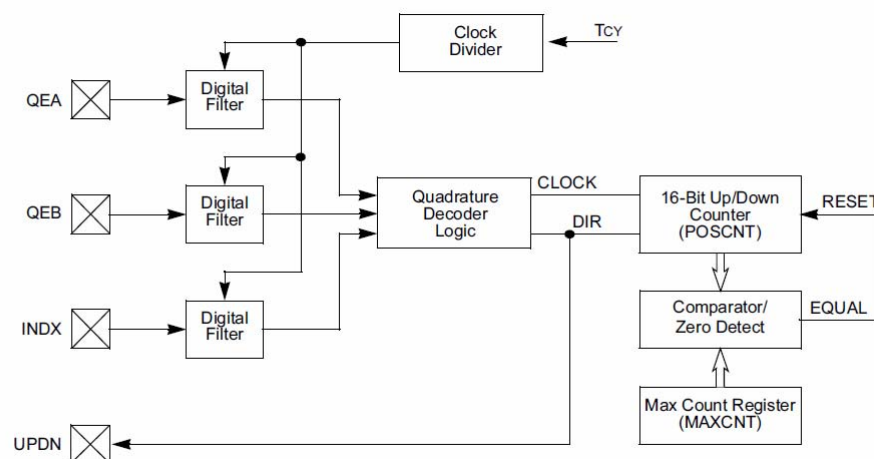


Figure IV14 : schéma bloc de l'interface QEI

IV.4.2.3 Modes de fonctionnement de l'encodeur en Quadrature

mode de mesure "x4" : dans ce mode le comptage de position sera effectuée dans les deux bords des signaux de d'entrée « phase A et B ». Le mode de mesure "x4" fournit des données de plus haute résolution (more position counts) afin de déterminer la position du codeur.

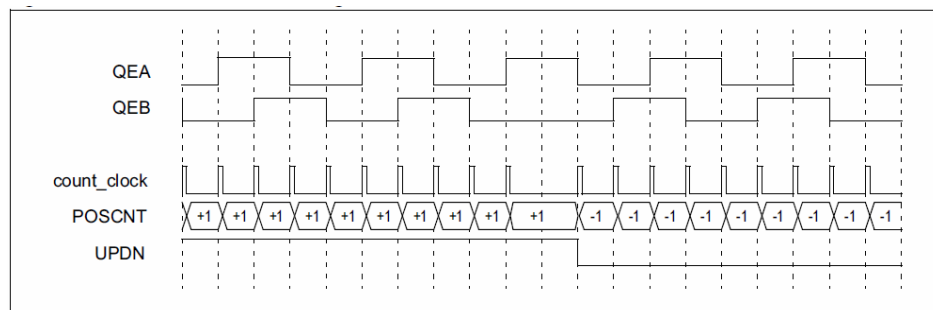


Figure IV.15 : signal de l'encodeur en mode X4

mode de mesure "x2" : dans ce mode le comptage de position sera effectuée seulement dans le front montant et descendant de la phase d'entrée A

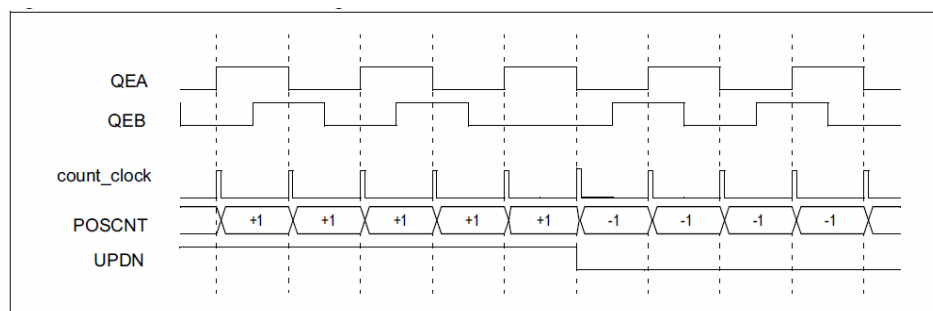


Figure IV.16 : signal de l'encodeur en mode X2

IV.5 Conclusion

Dans ce chapitre nous avons présenté un aperçu sur les dsPIC, reste maintenant à les utiliser pour concevoir la carte de commande et implémenter sur elles l'algorithme PID. Ça va être l'objet du prochain chapitre.

CHAPITRE V

Réalisation de la carte de commande

V.1. Introduction :

Nous allons maintenant décrire les différentes étapes de la réalisation de la carte de commande. Commençons tout d'abord par une description théorique des différents composants utilisés pour la réalisation de notre carte.

La carte de commande est basée sur le dsPIC30F2010. Elle est associée à un étage de puissance, convertisseur DC-DC. Chaque axe possède une carte spécifique.

V.2. Conception et réalisation de la carte de commande :

V.2.1. Schéma fonctionnel :

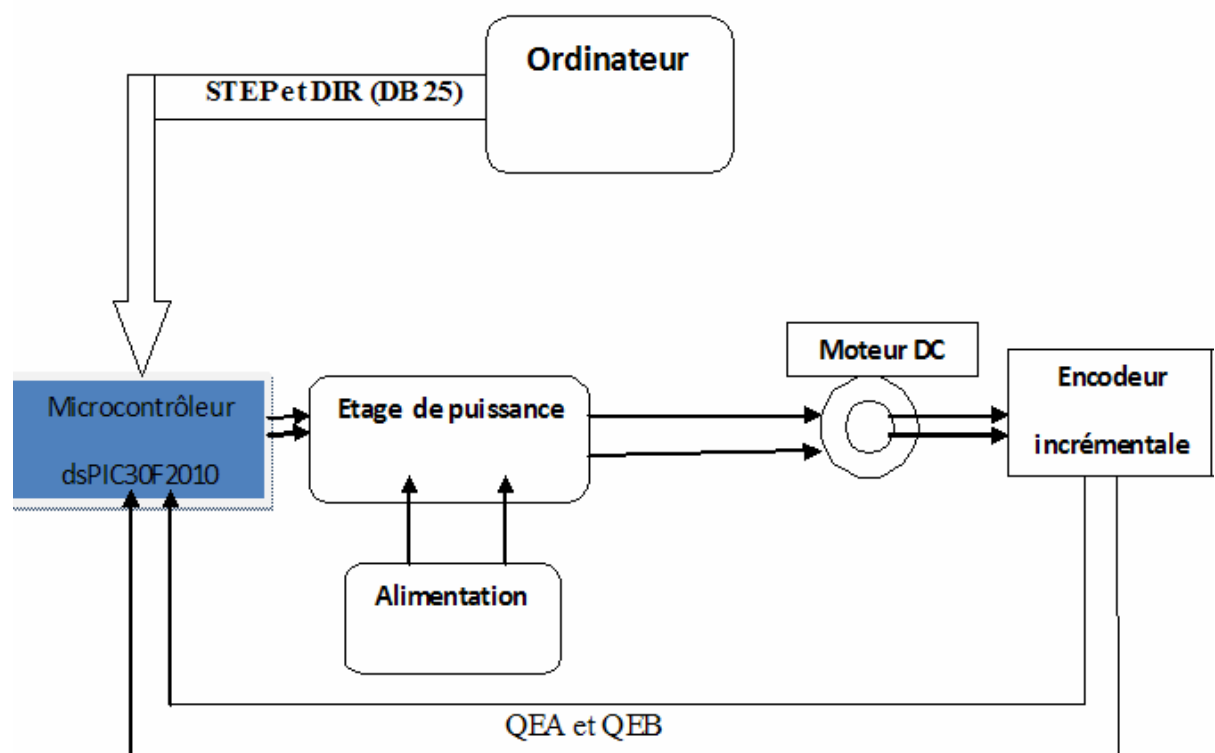


Figure V.1 Schéma fonctionnel

Dans la figure, on remarque que tout le circuit de commande est basé sur le dsPIC30F2010 car c'est lui qui va assurer la vitesse de calcul nécessaire pour la régulation du système.

Le moteur est piloté par le Hacheur alimenté en 35V, ce dernier étant lui-même piloté par le dsPIC30F2010 grâce à ses sorties MLI (PWM). Le signal MLI sortant dépend du type de la commande implémentée (indépendants ou complémentaires) ainsi que de la consigne et de la position.

Comme on l'a déjà dit dans le chapitre I, le retour d'information sur la position du moteur est assurée par l'encodeur incrémental, les signaux qui parviennent par l'encodeur seront injectés dans l'une des entrées QEA et QEB du dsPIC. Le dsPIC possède une horloge interne, mais pour une question de commodité, nous préférons utiliser une horloge externe.

Concernant l'alimentation, notre circuit sera alimenté par un transformateur qui transforme la

tension du réseau public (220 V AC) en une tension de (35V) .

D'autre part nous avons besoin d'une tension de 5V pour alimenter le dsPIC car ce dernier est fabriqué selon la technologie CMOS de 5V, pour cela on utilise un régulateur de tension : le LM7805 qui fournira une tension fixe de 5V.

V.2.2.Fabrication du circuit :

Pour la fabrication du circuit de commande nous avons utilisé la gamme de logiciels nommé «*ALTIUM DESIGNER*» l'ancien Protel.

Ce logiciel permet une assistance totale pour la construction de circuits imprimés, d'autres logiciels existent dans le marché et peuvent être utilisés comme : «NI Circuit Designer».

V.2.3. Notre carte de commande :

V.2.3.1. Alimentation :

Nous alimentons le DsPIC avec une tension de 5V, celle-ci est assurée par un régulateur de tension qui est alimenté avec plus de 5V.

L'alimentation du dsPIC se connecte aux pattes VDD et VSS, la première correspond au niveau logique haut (5V) et la deuxième au niveau logique bas. Des capacités de protection sont employées et reliées à la terre au niveau de la patte VDD (deux capacités en parallèle : 22 μ F et 10 μ F).

En remarque que plusieurs pattes VDD et VSS existent sur Le dsPIC cela a pour but de permettre plusieurs configurations de brochage selon le besoin.

Pour que le dsPIC fonctionne correctement, il faut relier la patte MCLR (*Master Clear*) au 5V par une résistance de 10k Ω et une capacité de 0.1 μ F en parallèle.

V.2.3.2. Circuit de l'horloge externe :

Comme nous l'avons déjà mentionné nous avons choisi d'utiliser une horloge externe.

L'horloge externe est un quartz (matériau piézo-électrique) de 4MHz (Fréquence d'oscillation propre du matériau).

Pour fonctionner correctement, il doit être relié aux entrées OSC1 et OSC2 du dsPIC, et muni de deux condensateurs de 27pF.

Nous avons la possibilité de multiplier la fréquence de l'horloge l'aide d'un module dit de PLL (voir chapitre IV). Nous choisissons de la multiplier par 16 (PLLx16). Or, pour exécuter une instruction, le processeur à besoin de 4 cycles d'horloge ; ce qui divise par 4 la fréquence de fonctionnement. On arrive ainsi à une fréquence de fonctionnement finale de 16MHz, soit 160ns / instruction.

V.2.3.3. Les entrées :

Pas et direction (STEP/DIR):

Ce sont la consigne. Ils sont injectés dans le dsPIC par les pins 15(INT1) et 16(INT0). Avant leurs branchements au dsPIC nous faisons passer par un filtre pass-bas RC de résistance de 100ohm et une capacité de 10nF

QEA et QEB :

Ces deux signaux sont récupérés de l'encodeur, ils sont branchés aux pins 6(QEA) et 7 (QEB). Nous leurs ajoutons également une résistance de 100ohm et une capacité de 10nF.

Port RJ11 :

Réservé pour la programmation du Microcontrôleur. Nous connectons à lui les broches PGC, PGD, VDD, GND et MCLR.

Port série :

C'est un port que nous utilisons pour le débogage, pour l'affichage et pour envoyer des données au PC. Il est relié aux broches 11 (ATX) et 12 (ARX). Et pour assurer la compatibilité avec la norme RS232, nous avons utilisé le circuit MAX232.

V.2.3.4. les sorties :

Les signaux Haut et Bas :

Ce sont des signaux de commande MLI(PWM).

La PWM1H sort de pin 25 qui représente le niveau haut (5v) et la PWM1L sort de pin 26 qui représente le niveau bas, ces deux signaux assurent le pilotage de l'Hacheur.

Nous avons ajouté un afficheur LCD pour l'affichage de la position (encodeur) et la consigne (la référence).

Nous donnons le schéma électrique global de la carte de commande dans la figure suivante :

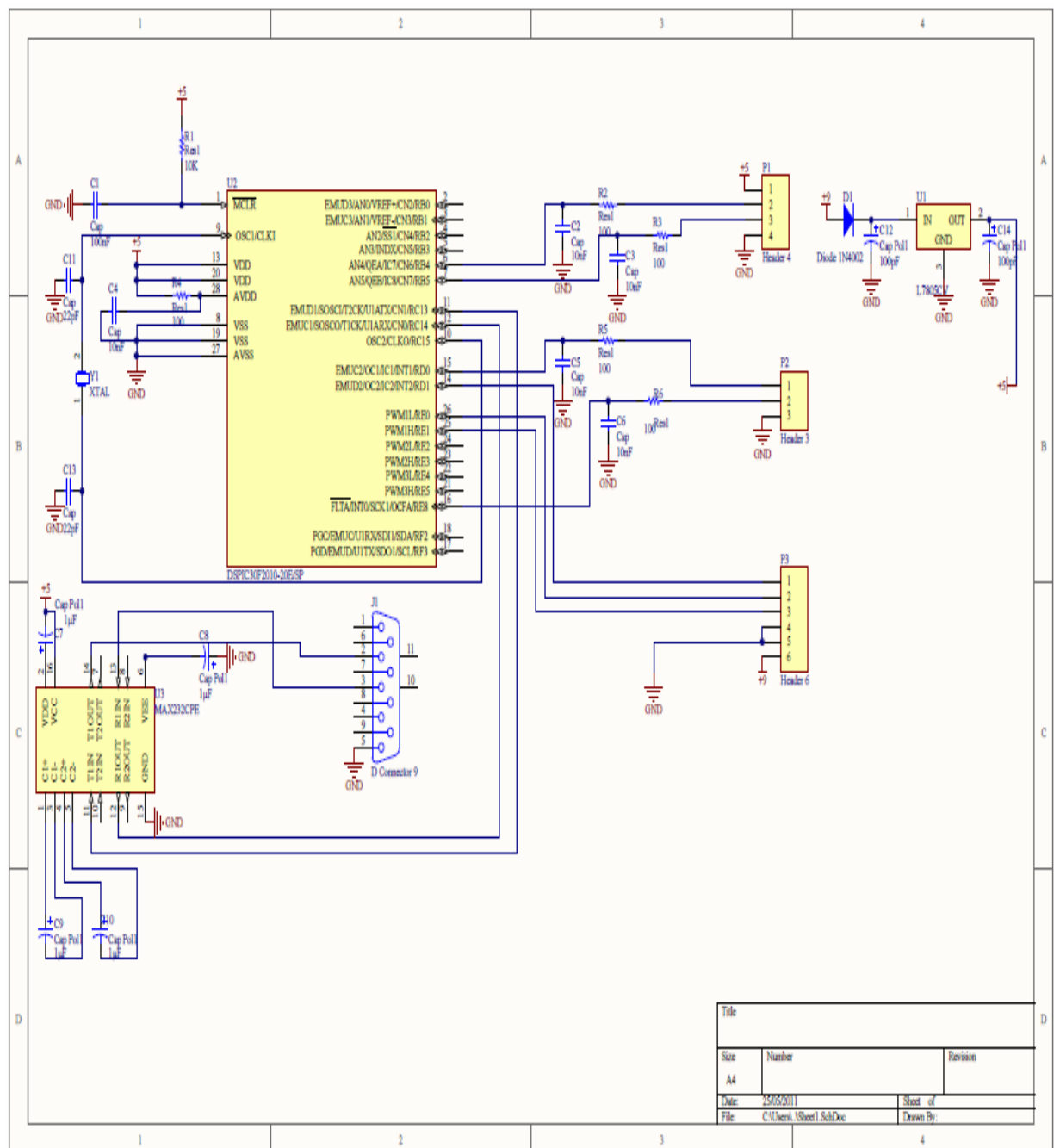


Figure V.2 schéma électrique complet.

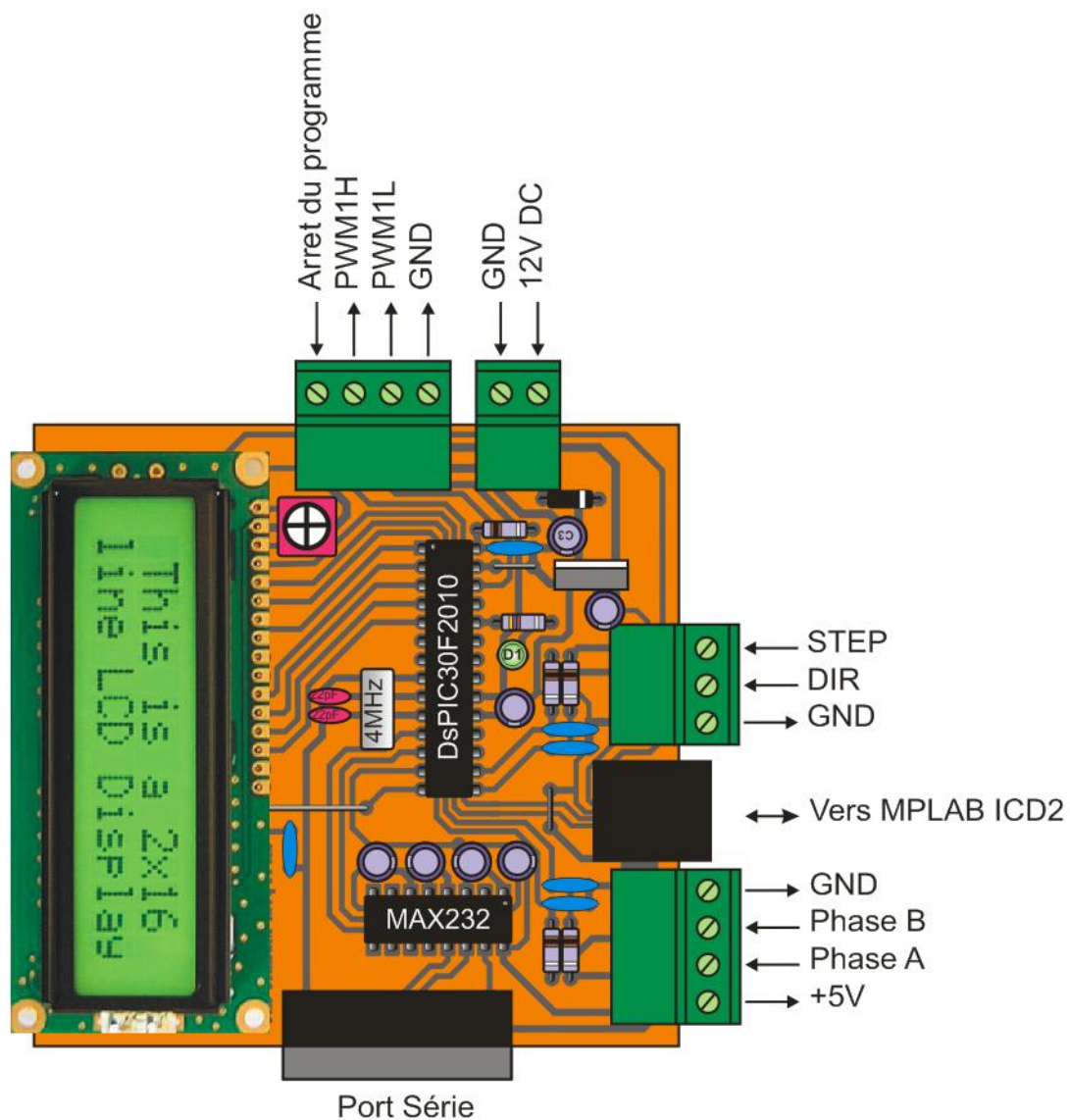


Figure V.3 implémentation des composants de la carte de commande

V.2.4. Schéma PCB sous Altium :

Altium permet de convertir le schéma précédent vers le schéma PCB pour fabriquer le circuit imprimé. Il faut veiller à ce que les pistes du circuit imprimé soient suffisamment larges et que les composants soient espacés pour ne pas trouver de problèmes lors du soudage des composants. Tout de même il ne faut pas les espacer trop pour que la carte de commande ne soit pas trop grande, ce qui augmente les coûts de fabrications.

Altium possède une bibliothèque qui contient les empreintes des composants électronique les plus courants avec leurs dimensions réelles ce qui assure qu'ils seront soudés sur la carte sans difficulté.

L'optimisation des pistes est effectuée automatiquement par le logiciel.

Voici les deux faces dites coté cuivre et coté composants.

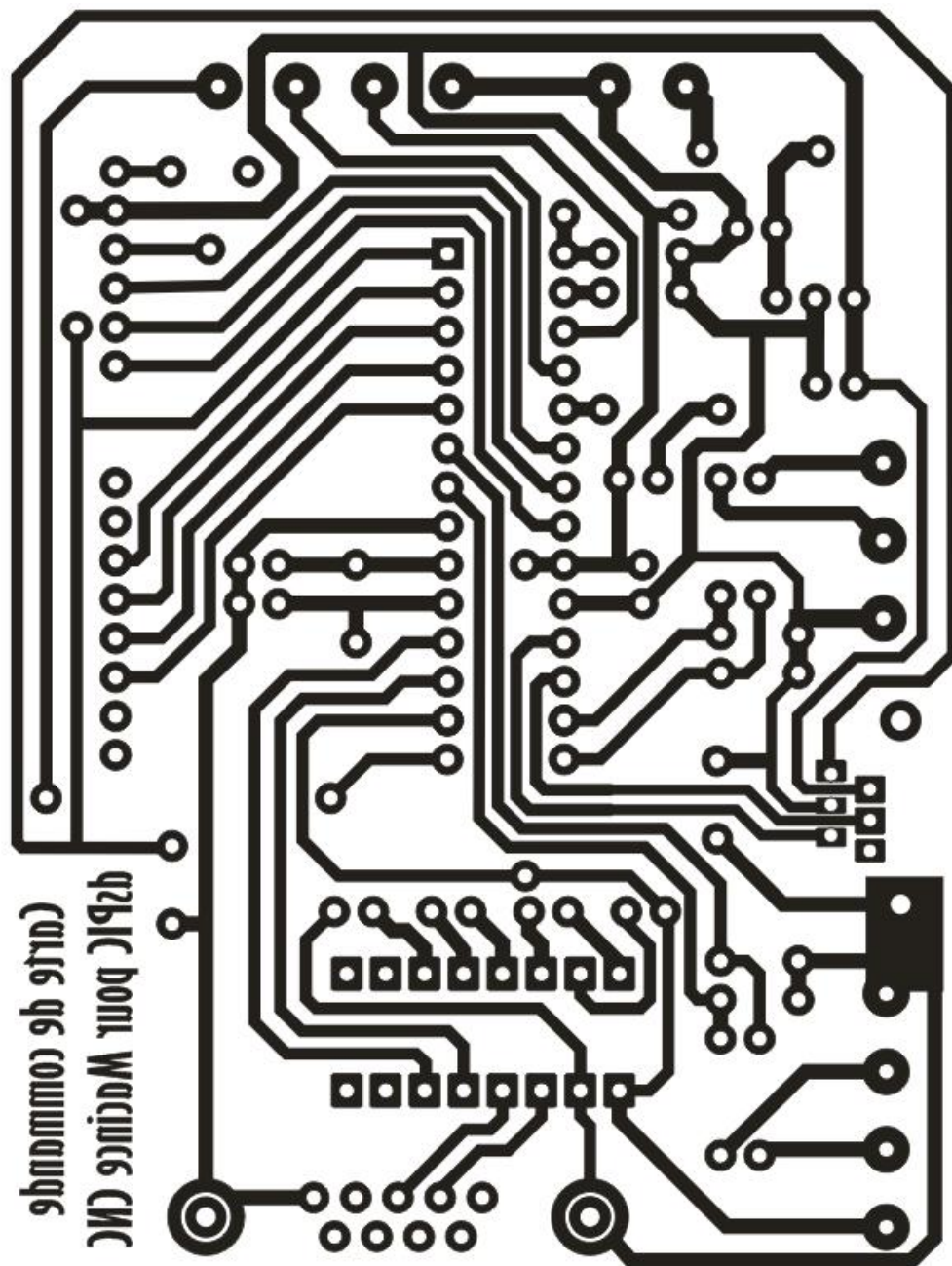


Figure V.4 Schéma du côté cuivre de la carte de commande

Et enfin voici l'image réelle de notre carte :

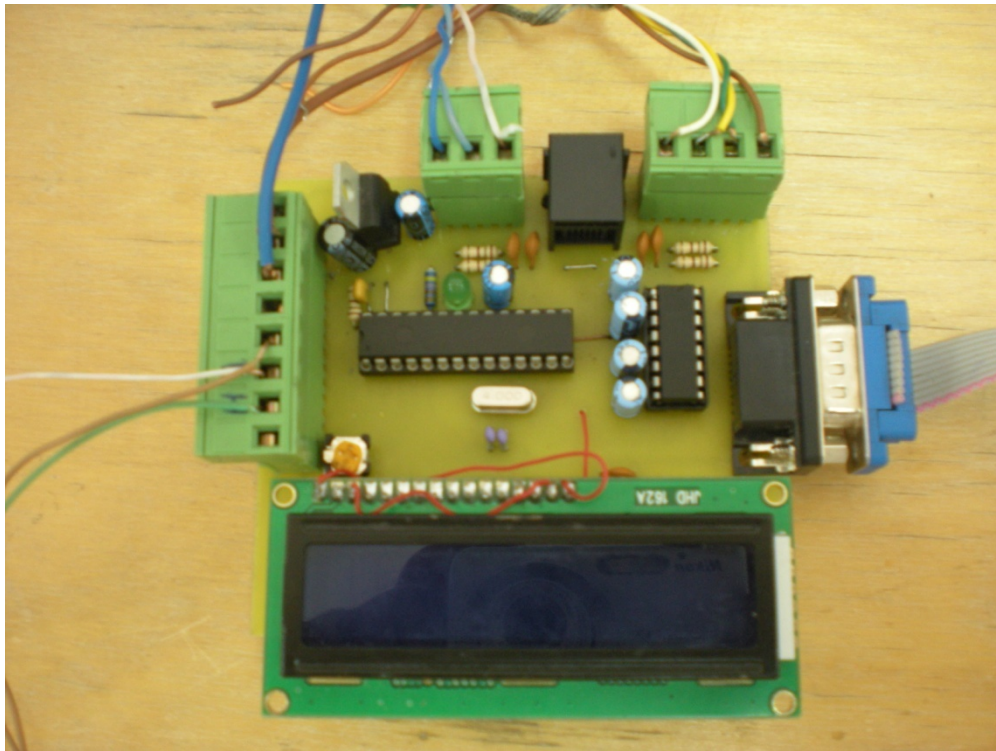


Figure V.5 figure de la carte

V.3. L'étage de puissance (le Hacheur) :

L'étage de puissance est un Hacheur alimenté par une source continue (dans notre cas le redresseur est intégré dans l'étage alors nous n'avons pas besoin d'une source DC) qui est transférée vers le moteur suivant la valeur du rapport cyclique calculé par la carte de commande.

Le hacheur que nous avons réalisé comprend 4 transistors MOSFET canal N associés avec un dissipateur de chaleur.

Le Hacheur est en pont de quatre MOSFET de type IRF740 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 côté 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.

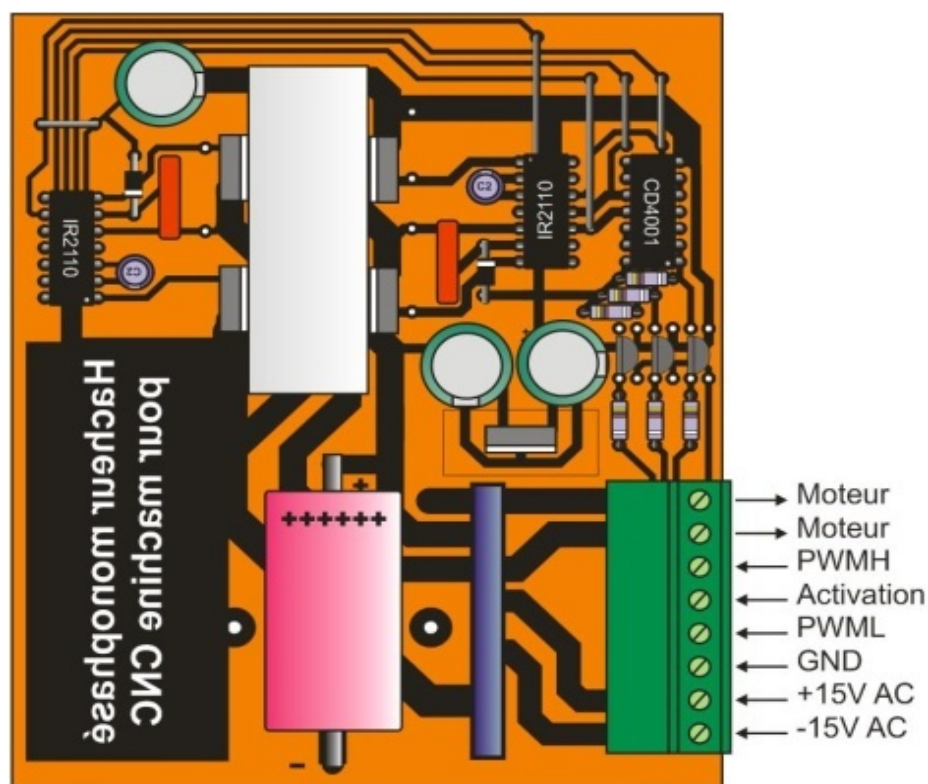


Figure V.6 Implémentation des composants sur notre Hacheur.

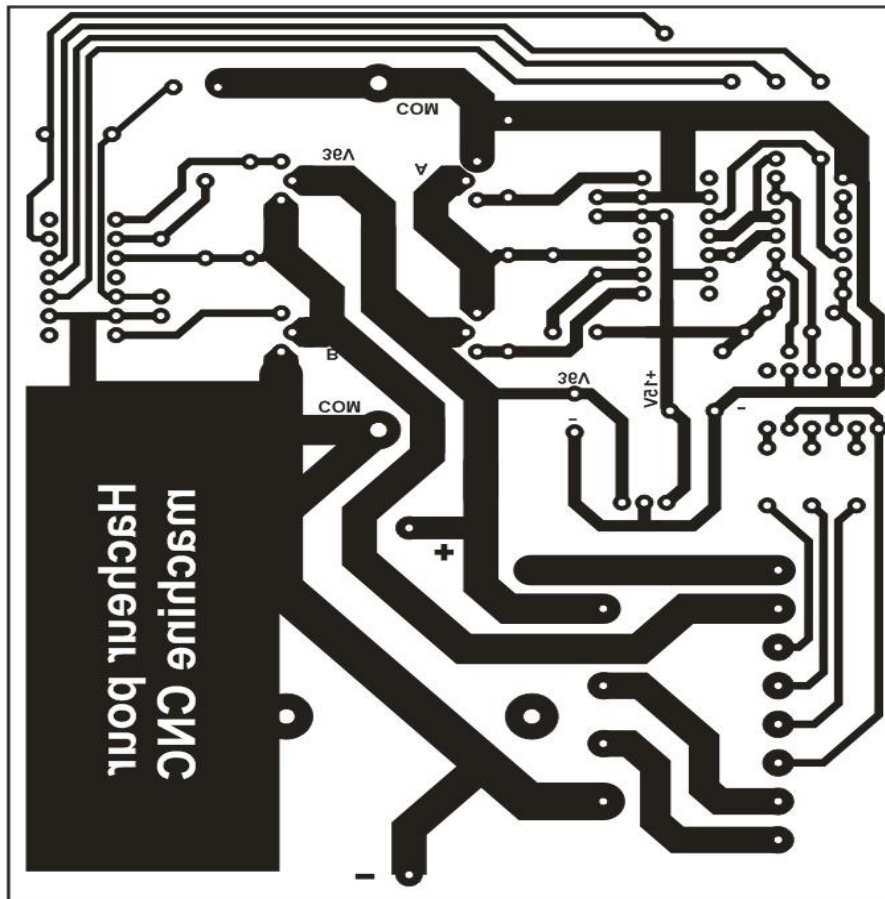


Figure V.7 Typon de l'Hacheur.

V.3.1 L'isolation:

Les drivers de notre hacheur peuvent fonctionner avec une tension de 15V (assurée par le régulateur de tension), donc il est nécessaire d'avoir une isolation pour protéger le microcontrôleur et l'ordinateur afin d'éviter les risques des surtensions.

L'isolation par des optocoupleurs est une bonne solution car elle offre une isolation en tension très élevée, mais dans notre cas l'optocoupleur que nous avons utilisé ne découpe pas correctement la tension, pour cela nous avons utilisé deux transistors bipolaires NPN BC337 pour assurer l'isolation du matériel et une meilleure transmission de la commande. Un troisième transistor BC337 pour les interruptions. Le BC337 inverse l'état logique des signaux, ce qui impose donc d'ajouter un inverseur CD4001 entre les transistors et les drivers des MOSFET.

V.4.1. Extraction de la réponse indicielle de l'axe X :

Pour obtenir la réponse indicielle de notre system on doit suivre les étapes suivantes :

V.4.1.1. Récupération des données de l'encodeur :

Pour la récupération des positions de l'encodeur, Nous avons ce dernier à la carte de commande, Nous avons attaqué le moteur avec une tension de 30V, les positions captés par l'encodeur sont transférées au PC sous forme hexadécimale sur 1 oct.

V.4.1.2. Conversion des donnés :

La conversion des données hexadécimale sur (1Bit) aux valeurs numériques sur un tableau pour obtenir la réponse du système on position sur MATLAB se fait par un programme en C :

```
#include <stdio.h>
#include <stdlib.h>
FILE *in;
FILE *out;
int main (int argc, char *argv [])
{
    short int c, a, b;
    in = fopen("res.txt","rb");
    out = fopen("Courbe.txt","w");
    while (!feof(in))
    {
        fscanf(in,"%X", &b);
        //printf("b = %d\n",b);
        fscanf(in,"%X", &a);
        //printf("a = %d\n",a);
        c = a*256+b;
        fprintf(out,"%d\n", c);
    }
    fclose(in);
    fclose(out);
    system("PAUSE");
    return 0 ;
}
```

V.4.2. Extraction de la réponse en position :

Avec les données précédentes et à l'aide d'un programme MATLAB nous avons obtenu la réponse indicielle en vitesse du système. Elle est tracée sur la figure IV.9.

```
function Conversion
% Ceci est le Help de la commande Conversion
in = fopen('resultats.txt','r');
out = fopen('Courbe.txt','w');
aa = fscanf(in, '%X');
for i=1:2:2765
N((i+1)/2)=aa(i)*256+aa(i+1)
end
```

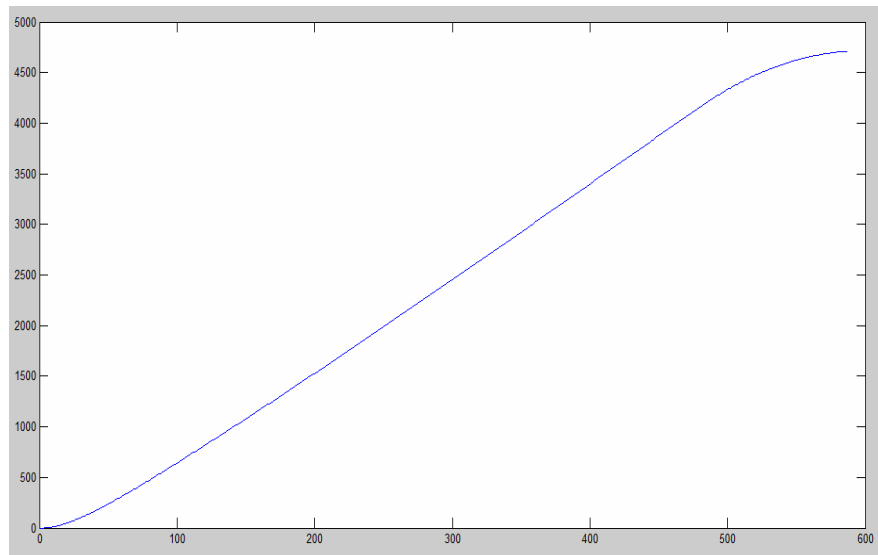


Figure V.8 La réponse indicielle en position (step)

Pour obtenir la réponse en vitesse nous avons dérivé la réponse en position par le programme MATLAB suivant:

```
Vitesse(1)=Courbe(1)/.003;
for i=2:587
Vitesse(i)= (Courbe(i)-Courbe(i-1))/.003;
End
```

Le résultat est affiché sur la figure (V.9)

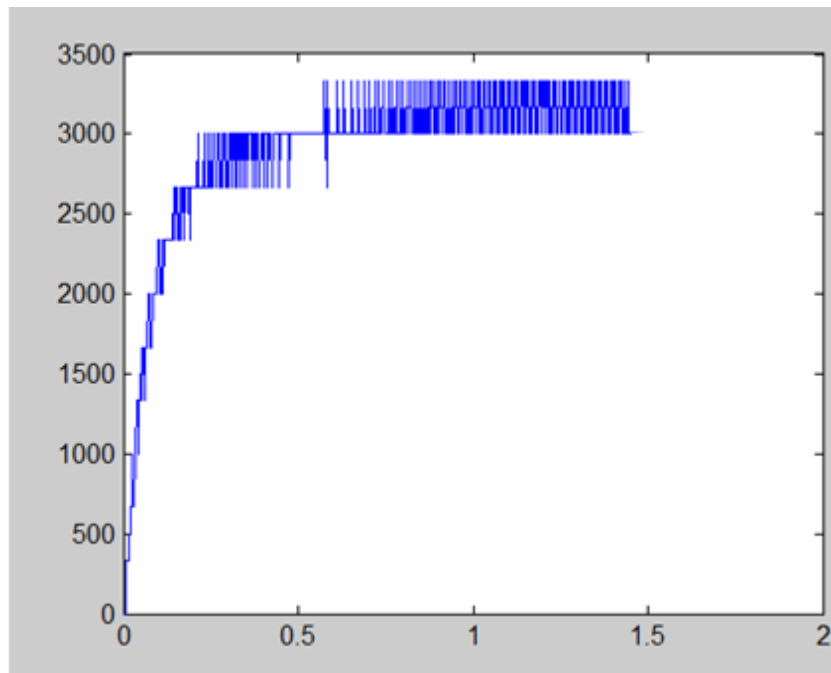


Figure V.9 La dérivée de la réponse du système à un échelon (step/sec)

V.4.3. Calcul de τ_m et K_m et la fonction de transfert :

$W(\infty) = 3000$ pas/sec et comme notre encodeur contient 40 pas par tour implique que

$$\Omega(\infty) = \frac{3000}{40} \cdot 6.28 = 471 \text{ rad/sec}$$

$$K = \frac{\Omega(\infty)}{U} = 13.457$$

A $t = t_0 = 0.1$ sec on a

$$W(t_0) = 2000 \text{ pas/sec}$$

$$\Omega(t_0) = 314 \text{ rad/sec}$$

$$\tau_m =$$

V.4.3.1. Fonction de transfert en vitesse :

$$F(P) = 13. \frac{457}{0.091 P + 1}$$

(V.19)

Après dérivation de la fonction de transfert on obtient :

V.4.3.2. Fonction de transfert en position :

$$H(P) = 13. \frac{457}{P (0.091 P + 1)}$$

(V.20)

V.5. Configuration et Programmation du dsPIC:

Le langage utilisé pour programmer les dsPIC est le MPLAB C30. C'est du C classique auquel viennent se rajouter quelques petites particularité. Le logiciel utilisé pour le développement est le MPLAB IDE, fourni gratuitement par Microchip. Pour le compilateur MPLAB C30 se présente en deux versions, payante : pour les entreprises, gratuite : pour les étudiants et pour usage personnel.

V.5.1. Outils de développement :

Pour le flashage du dsPIC, il ya plusieurs gammes d'outils de développement suggéré par Microchip ou par d'autres firmes.

Microchip propose le programmeur / débogueur ICD2 (In-Circuit Debugger) qui permet de faire le flashage du dsPIC sans avoir être enlevé du circuit imprimé.



Figure V.10 Package du programmeur ICD2

L'environnement de développement intégré (IDE) MPLAB regroupe tous les outils nécessaires à la mise au point d'une application avec un cœur de microcontrôleur dsPIC, entre autres, de Microchip :

- Editeur de texte interactif .
- Compilateur C30 (et assembleur).
- Simulateur.

- Debugger si on dispose de l'équipement nécessaire.

Notre programme consiste à configurer les différents blocs que nous avons utilisé dans notre application et d'implémenter les valeurs des coefficients de régulateur PID.

V.5. 2. Configuration du dsPIC30F2010 :

Nous avons dit que le dsPIC30F2010 de Microchip intègre une énorme quantité de fonctionnalités dans une seule puce, et grâce à son coprocesseur DSP on arrive à effectuer un calcul très fastidieux en un temps très réduit.

Nous donnons dans ce qui va suivre, les configurations apportées sur chaque module utilisé sur le dsPIC.

a) Configuration d'Interface (QEI) :

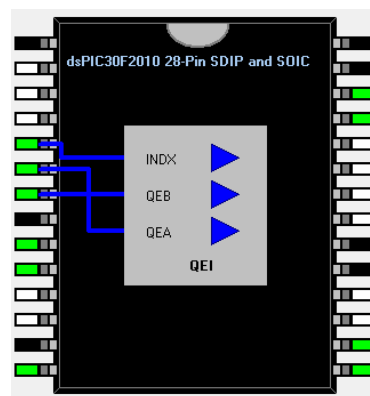


Figure V.11 les pins du module QEI

- Mode utilisé : x4 (4 impulsions pour chaque trou de l'encodeur).
- Activation du filtre numérique pour QEA, QEB avec un préscalaire de 1:256
- Reset du registre POSCNT lorsque il devient supérieur à MAXCNT

b) Configuration du module PWM :

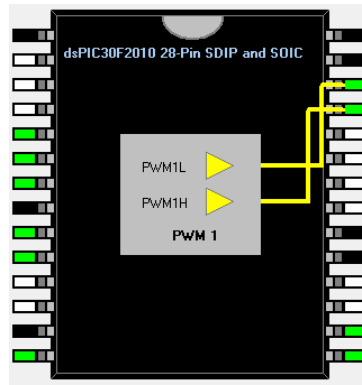


Figure V.12 les sorties des signaux PWM

- Activer les broches PWM1H et PWM1L.
- PWM1H, PWM1L en mode complémentaire.
- Choisir le mode (free runing) comme mode de travail.
- Activer l'interruption PWM en fixant la fréquence à découpage sur 0.2ms.
- Activer le temps mort avec un période de 10 μ s.

d) Configuration du module UART :

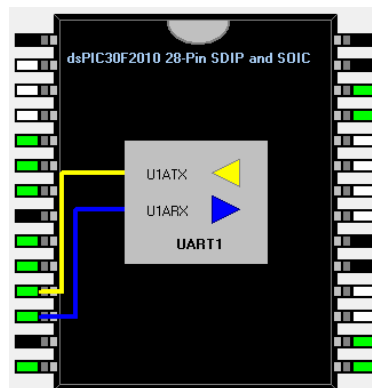


Figure V.13 les pins du module UART

- Activer le module UART au démarrage.
- Vitesse 9600 bps, 8bits de données, 1bit d'arrêt.
- Activer la transmission.
- Activer (les broches alternatives)

e) Configuration de l'Oscillateur :

Nous avons utilisé un quartz de 4MHz et nous avons obtenu la fréquence 16MIPS en activant l'oscillateur primaire en mode XT x16.

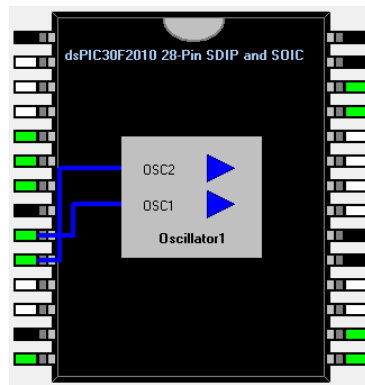
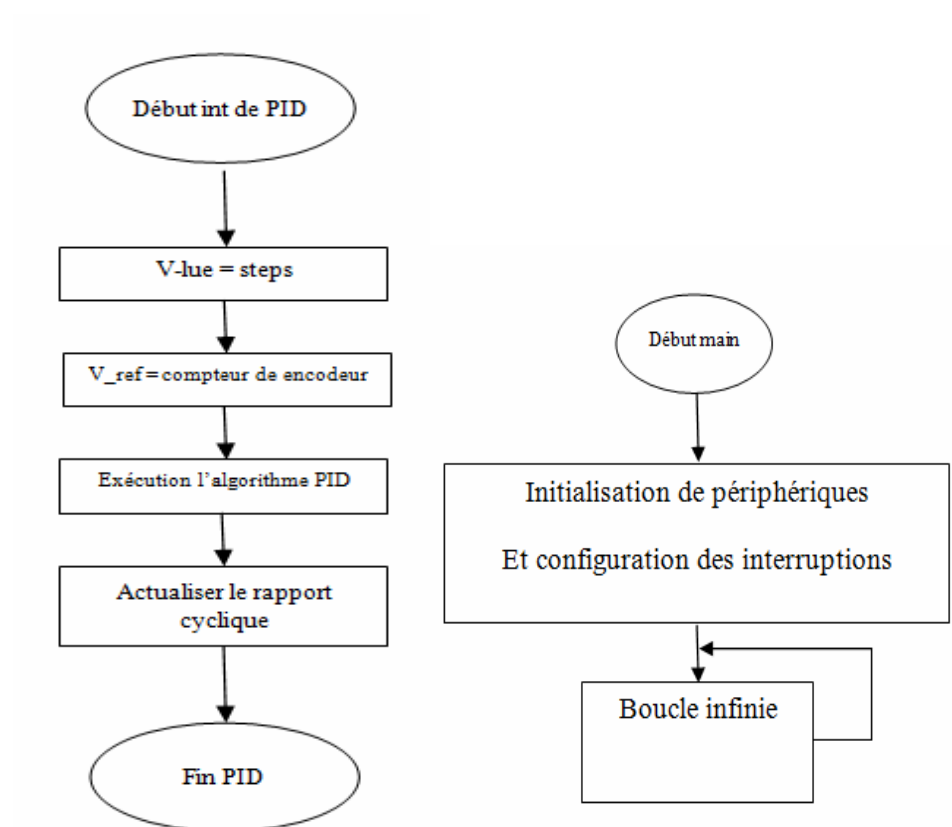


Figure V.14 vers l'oscillateur extérieur

V.6. Organigrammes du programme :



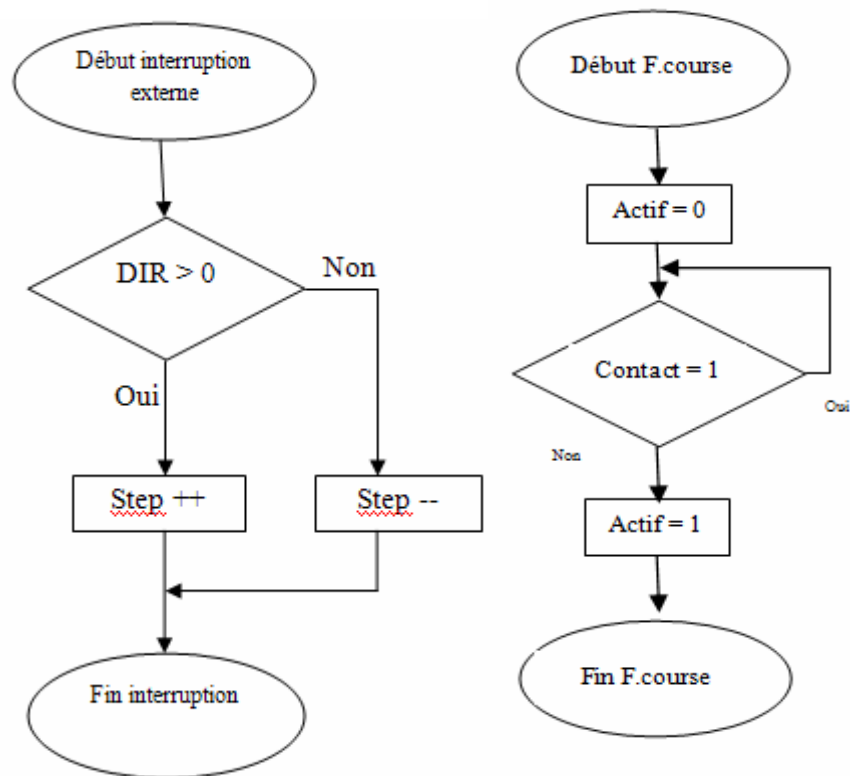


Figure V.15 organigramme du programme implanté dans le dspic30F2010

V.7. Implémentation du contrôleur PID :

L'élément le plus important qui nous a poussé à opter pour les dsPIC est le contrôleur PID.

Le dsPIC doit exécuter un cycle de l'algorithme de régulation PID dans un temps ne dépassant pas un de découpage qui est égal dans notre cas à 200us.

Pour bien profiter du coprocesseur mathématique incorporé dans le dsPIC, nous avons choisi d'utiliser la librairie PID fournie avec le compilateur MPLAB C30 par Microchip.

V.7.1. Fonctions de la bibliothèque PID et structure de données :

La bibliothèque DSP du compilateur MPLAB C30 offre la fonction PID qui programmée en niveau assembleur en utilisant les instructions MAC et le mode d'adressage circulaire.

Voici le schéma en bloc de l'algorithme de calcul de la librairie PID de Microchip.

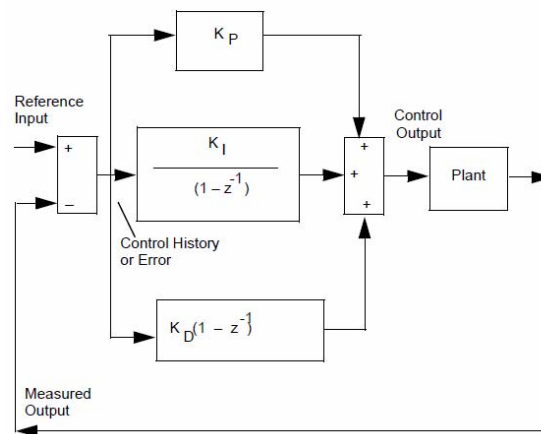


Figure V.16 régulateur PID

Les paramètres de la fonction PID sont organisés en une structure de données définie dans le fichier d'entête « dsp.h » comme suit:

```
typedef struct {
fractional* abcCoefficients;
fractional* controlHistory;
fractional controlOutput;
fractional measuredOutput;
fractional controlReference;
}tPID ;
```

Cette librairie contient les trois fonctions suivantes :

PIDCoeffCalc(fractional* kCoeffs, tPID* controller) elle a deux arguments : le pointeur sur la structure tPID, et un pointeur sur un tableau contenant les coefficients Kp, Ki, Kd. Cette fonction permet le calcul des coefficients A, B et C.

PIDInit(tPID* controller) cette fonction initialise les variables temporelles de l'algorithme PID. Son argument est le pointeur sur la structure tPID.

PID(tPID* controller) : elle exécute l'algorithme PID de la manière suivante :

controlOutput[n] = controlOutput[n - 1]abcCoefficients[0] + controlHistory[n] * abcCoefficients[1] + controlHistory[n] * abcCoefficients[2]

Ou :

$$A = \text{abcCoefficients}[0] = k_p + k_i + k_d$$

$$B = \text{abcCoefficients}[1] = -(k_p + 2 * k_d)$$

$$C = \text{abcCoefficients}[2] = k_d$$

$$\text{Control History}[n] = \text{measuredOutput}[n] - \text{referenceInput}$$

L'exécution de cette fonction prend 30 cycles (y compris le temps perdu lors de l'appel de cette fonction et de son retour).

L'élément `controlHistory` de la structure est un pointeur vers 3 échantillons d'erreurs

($E(k)$, $E(k-1)$, $E(k-2)$) situés dans l'espace de données Y, avec le premier échantillon étant la plus récente (en cours). Ces échantillons constituent l'historique des différences actuelles et passées entre l'entrée de référence et la sortie mesurée du système.

La fonction `PIDInit()` efface les éléments pointés par (`controlHistory`). Elle efface également l'élément (`controlOutput`) dans la structure des données (`rPID`).

V.8.1. signaux de commande :

Le but de tout ce qu'on a parlé dans la « partie commande » est d'avoir les signaux qui commandent les transistors de l'hacheur, on parle de signaux MLI variable qui permettent de contrôler la tension aux bornes du moteur ce qui amène alors à contrôler ses mouvements.



Figure V.17 les deux signaux PWM complémentaires de commande

V.8.2. partie puissance :

Les signaux PWM issues de la partie commande sont emmenés vers la partie puissance de notre carte de contrôle.

Comme nous l'avons déjà expliqué (dans le chapitre 3), l'erreur entre la consigne et la mesure est manipulé par le correcteur PID de sorte que la réponse de notre système soit acceptable, la sortie du PID est en rapport avec le rapport cyclique du signal PWM et sa composante complémentaire, ces signaux contrôlent la durée d'ouverture et de la fermeture des transistors qui travaille en commutation (bloqué, saturé), alors on peut contrôler la tension aux bornes du moteur et son sens de rotation.

V.9. Choix des coefficients du régulateur PID :

L'obtention des paramètres du régulateur numérique à implémenter dans notre dsPIC se fait par des essais expérimentaux sur la machine. Chaque fois nous varions les coefficients KP, KI et KD dans notre programme, et nous donnons une consigne avec le logiciel de contrôle Mach3, en remarquant sur l'afficheur LCD les différents résultats (consigne, mesure et l'erreur),

Comme une première étape nous avons fixé KI et KD à zéro et nous avons fait varier KP et après avoir de bons résultats, nous avons fait varier le KI et enfin le KD. Le tableau suivant donne les valeurs utilisées pour l'asservissement de la machine CNC.

KP	KI	KD	La réponse	L'erreur
0.9	0	0	Grand dépassement	Énorme
0.8	0	0	Grand dépassement	Grande
0.45	0	0	Petit	Plus de 3mm

			dépassement	
0.6	0	0	Pas de dépassement	Moins de 3mm
0.6	0.5	0.001	Convergence	Énorme
0.6	0.0012	0.07	Pas de dépassement	Moins de 3mm
0.6	0.0012	0.09	Pas de dépassement	e = 1mm

Tableau V.1 synthèse des coefficients du PID

KP = 0.6

KI = 0.0012

KD = 0.009

V.10. Test avec le logiciel de commande Mach3 :

Dans ce test, nous avons écrit un petit programme en mode MDI sur Mach3, et nous avons suivi la réponse et l'erreur.

Programme test :

G00 X100

G00 X00

G00 X-100

G00 X-50

G00 X00

G00 X50

G00 X100

G00 X00

M02

Après l'exécution de ce programme Nous avons vu que l'axe fonctionne en souplesse avec un bon démarrage, mais y a toujours une petite marge d'erreurs mais qui sont trop faibles (de l'ordre de dixième de mm). Cela d'abord revient à la mauvaise résolution de l'encodeur utilisé dans l'axe x (40 impulsions par révolution).

V.11. Conclusion :

A la fin de ce chapitre, nous pouvons dire que nous avons pu découvrir les dsPIC et leurs puissance, manipuler les outils de programmation, simulation, compilation du microcontrôleur et le monde des machines CNC. Ces tâches sont nécessaires pour tout travail pratique à réaliser.

Conclusion Générale :

Le but de notre projet était l'implémentation d'un PID numérique sur un dsPIC pour la régulation en position des machines CNC en passant par la réalisation de la carte de commande. Nous avons atteint ce but.

En réalisant ce projet, nous avons constaté au premier lieu la nécessité d'un asservissement précis des moteurs d'avance; et que la grande vitesse de calcul du module DSP a répondu aux besoins du PID discret qui s'exécute dans un temps de calcul inférieur à 200 μ s.

Nous avons pu manipuler deux logiciels de commande Mach3 et EMC2, en les configurant à générer les consignes désirées.

La conception de la carte de commande était la tache la plus importante et qui nous a pris le plus de temps mais en effet, nous avons appris à manipuler des logiciels de la conception des circuits électroniques, comme ALTIUM et l'implantation des composants sur la carte.

Et pour l'identification, nous avons eu l'occasion d'utiliser la méthode expérimentale basée sur les graphes pour extraire les paramètres du modèle mathématique du moteur relatif à l'axe x. ça nous a permis aussi de découvrir d'autres méthodes basées sur la programmation.

Et enfin notre système a fonctionné avec la meilleure façon tout commettant une très légère marge d'erreur.

En réalisant ce travail nous avons pensé d'améliorer la simulation sur MATLAB en utilisant la librairie du dsPIC, et de construire une machine à 5 axes dans le futur.